

# Xojo Documentation

User Guide, Reference Guide and Resources



Thursday, December 03, 2015

Copyright © 2015 Xojo, Inc.. All Rights Reserved.

# Xojo Documentation

Copyright © 2015 Xojo, Inc.

The information contained in this document is subject to change without notice. This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced, or translated to another language without the prior written consent of Xojo, Inc.

# Table of Contents

Getting Started .....	1
Get Started with Xojo .....	1
QuickStart Guides .....	4
iOS QuickStart .....	4
User Guides .....	16
2015 Release 3 Highlights .....	16
64-bit Guidelines .....	18
User Guide .....	23
Welcome! .....	23
Fundamentals .....	27
Fundamentals Overview .....	27
Project Chooser .....	28
The Workspace .....	30
Overview .....	30
Navigator .....	32
Layout Editor .....	40
Library and Inspector .....	48
Code Editor .....	54
Other Editors .....	61
Find/Errors/Messages Panels .....	65
Running Your Apps .....	68
Project Types .....	73
Preferences .....	78
The Xojo Language .....	86
Writing Code .....	86
Variables and Constants .....	88
Using Data Types .....	91
Collections of Data .....	101
Comparing Values .....	108
Controlling Code Flow .....	112
Repeating Code .....	120
Properties .....	127
Methods .....	130
Enumerations .....	137
Modules .....	138
External Methods and Structures .....	143
Object-Oriented Programming .....	146
Object-Oriented Programming .....	146
Classes .....	147
OOP Design Concepts .....	153
Properties, Methods and Events .....	158
Constructors and Destructors .....	161
Interfaces .....	163
Subclassing Examples .....	167
Advanced Features .....	169

App Structure .....	174
App Structure Overview .....	174
iOS .....	175
iOS Apps .....	175
Launch Images and App Icons .....	179
Retina Images .....	184
Using the iOS Simulator .....	185
Copying Files to the Device .....	187
Deploying iOS Apps .....	190
Ad-Hoc Device Deployment for Testing .....	191
Submitting to the App Store .....	196
Free Provisioning Profile .....	198
Desktop Apps .....	204
Web Apps .....	214
Console Apps .....	219
Compilation Constants .....	223
Debugging and Profiling .....	224
About Debugging .....	224
Using the Debugger .....	227
Exception Handling .....	232
Remote Debugging .....	236
Performance Tuning .....	239
Build Automation and IDE Scripting .....	245
Build Automation .....	245
IDE Scripting .....	247
IDE Scripting Commands .....	249
Building Commands .....	250
DoCommand .....	255
Input/Output Commands .....	259
Notification Commands .....	261
Project Commands .....	263
String Commands .....	271
User Interface .....	273
iOS .....	273
Overview .....	273
Screens .....	274
Views .....	277
Auto-Layout .....	280
Buttons .....	286
Pickers .....	291
Inputs .....	296
Decorations .....	300
Indicators .....	305
Viewers .....	308
Toolbars .....	319
iOS Guide .....	322
iOS Guide .....	322

iOS Overview .....	323
The New Xojo Framework .....	328
Migrating to the New Framework .....	331
Desktop .....	337
App Icons .....	337
Web .....	338
Deployment Overview .....	338
Linux Deployment .....	341
Drag and Drop .....	344
Security .....	346
Sample App .....	348
Add Web Apps to iOS Home Screen .....	349
Secure Login Screens .....	350
Run Standalone Web App in the Background .....	353
Xojo Web App Security .....	355
Raspberry Pi .....	357
Overview .....	357
GPIO .....	361
Blinking LED Tutorial .....	362
Button LED Tutorial .....	370
Desktop GPIO App .....	375
Technical Information .....	376
Creating Crash Dumps .....	376
Internal Plugins .....	377
Keyboard Shortcuts .....	379
Uniform Type Identifiers .....	383
App Deployment .....	385
Desktop Deployment .....	385
Web Deployment .....	390
IIS Deployment .....	397
Migrating from Other Languages .....	407
Migrating from Visual Basic .....	407
Migrating from Microsoft Access .....	411
Migrating from FileMaker .....	413
Migrating from Visual FoxPro .....	420
Eddie's Electronics Web Service .....	422
Windows OS .....	425
InnoSetup .....	425
Elevating User Access Control .....	427
Reference .....	428
Overview .....	428
Language .....	429
Overview .....	429
Commands .....	430
Overview .....	430
AddHandler .....	433
App .....	435

Array .....	436
As .....	438
Assigns .....	439
ByRef .....	440
ByVal .....	441
Break .....	442
Call .....	443
Case .....	444
Catch .....	445
Const .....	446
Continue .....	448
CurrentMethodName .....	449
Declare .....	450
iOS Declare Example Projects .....	454
OS X Declare Samples .....	456
Windows Declare Samples .....	458
Dim .....	459
Do .....	461
Do...Loop .....	462
DownTo .....	465
Each .....	466
Else .....	467
Elseif .....	468
End .....	469
Exit .....	470
Extends .....	472
False .....	474
Finally .....	475
For .....	476
For...Next .....	477
For Each...Next .....	483
Global .....	486
If .....	488
If...Then...Else .....	489
In .....	491
Lib .....	492
Line Continuation .....	493
Loop .....	494
Me .....	495
Next .....	496
Nil .....	497
Optional .....	499
ParamArray .....	500
Raise .....	501
RaiseEvent .....	502
ReDim .....	503
Rem .....	504

RemoveHandler .....	505
Return .....	507
Select .....	509
Select...Case .....	510
Self .....	512
Soft .....	513
Static .....	514
Step .....	516
Super .....	517
Then .....	518
To .....	519
True .....	520
Try .....	521
Try...Catch .....	522
Ubound .....	524
Until .....	525
Using .....	526
Wend .....	528
While .....	529
While...Wend .....	530
Compiler Constants .....	532
Conditional Compilation .....	534
Data Types .....	535
Overview .....	535
Arrays .....	536
Auto .....	543
Boolean .....	545
Color .....	546
Currency .....	551
Double .....	553
Integer .....	556
Text .....	561
Advanced Data Types .....	578
CFStringRef .....	579
CString .....	580
Delegate .....	581
Integer (size-specific) .....	583
OSType .....	585
Ptr .....	586
Single .....	590
Structure .....	592
WString .....	593
Literals .....	594
Operators .....	597
Overview .....	597
Addition (+) .....	599
AddressOf .....	600

And	601
CType	603
Division (/)	605
Division, Integer ()	606
Equals (=)	608
Not Equals	610
Exponentiation (^)	612
If	613
Greater Than	614
Is	616
IsA	617
Less Than	619
Mod	621
Multiplication (*)	623
New	624
Not	626
Operator Overloading	628
Or	636
Subtraction (-)	638
Xor	640
WeakAddressOf	642
Pragma Directives	643
Reserved Words	645
Classes	647
Constructor	647
Destructor	648
Enumeration	649
Xojo Framework	650
Xojo Framework Overview	650
Core	653
Core Overview	653
Date	655
DateInterval	663
Dictionary	665
DictionaryEntry	670
Iterable	671
Iterator	672
Locale	674
MemoryBlock	677
MutableMemoryBlock	685
Point	692
Rect	696
Size	700
StackFrame	702
TextEncoding	703
Timer	707
TimeZone	711

WeakRef .....	712
Exceptions .....	714
BadDataException .....	714
ErrorException .....	715
InvalidArgumentException .....	717
IteratorException .....	718
LogicException .....	719
UnsupportedOperationException .....	720
Crypto .....	721
Crypto Overview .....	721
CryptoException .....	727
Data .....	728
Data Overview .....	728
InvalidJSONException .....	731
Introspection .....	732
Introspection Overview .....	732
AttributeInfo .....	733
ConstructorInfo .....	735
MemberInfo .....	736
MethodInfo .....	738
ParameterInfo .....	740
PropertyInfo .....	741
TypeInfo .....	743
IO .....	747
IO Overview .....	747
BinaryStream .....	748
FolderItem .....	755
IOException .....	762
SpecialFolder .....	763
TextInputStream .....	766
TextOutputStream .....	770
Math .....	774
Math Overview .....	774
Net .....	783
Net Overview .....	783
HTTPSocket .....	784
NetException .....	790
SSLSettings .....	791
TCPSocket .....	793
System .....	797
System Overview .....	797
Threading .....	799
Threading Overview .....	799
CriticalSection .....	800
IllegalLockingException .....	801
Semaphore .....	802
Thread .....	803

iOS Framework .....	807
iOS Framework Overview .....	807
iOSApplication .....	809
iOSBitmap .....	811
iOSBlock .....	813
iOSButton .....	815
iOSCanvas .....	817
iOSContainerControl .....	820
iOSControl .....	821
iOSDatePicker .....	825
IOSEventInfo .....	829
iOSFont .....	831
iOSGraphics .....	835
iOSHTMLViewer .....	847
iOSImage .....	849
iOS Image Formats .....	852
UIImageView .....	854
UILabel .....	856
iOSLayoutConstraint .....	859
iOSLine .....	864
iOSMessageBox .....	866
iOSSOval .....	868
iOSPath .....	869
iOSProgressBar .....	874
iOSProgressWheel .....	877
iOSRectangle .....	879
iOSScreen .....	881
iOSScreenContent .....	882
iOSSegmentedControl .....	883
iOSSegmentedControlItem .....	886
iOSSeparator .....	889
iOSSound .....	890
iOSSlider .....	892
iOSSplitContent .....	895
iOSSplitView .....	896
iOSSwitch .....	898
iOSTabBar .....	900
iOSTabContent .....	902
iOSTable .....	903
iOSTableCellData .....	912
iOSTableDataSource .....	915
iOSTextArea .....	917
iOSTextField .....	920
iOSToolbar .....	924
iOSToolButton .....	927
iOSUserControl .....	934
iOSView .....	935

iOSViewController .....	943
SQLiteDatabase .....	944
Example Projects .....	950
SQLiteDatabaseField .....	951
SQLiteException .....	954
SQLiteRecordSet .....	955
Exceptions .....	958
Overview .....	958
FunctionNotFoundException .....	959
KeyNotFoundException .....	960
NilObjectException .....	961
ObjCException .....	962
OutOfBoundsException .....	963
OutOfMemoryException .....	964
RuntimeException .....	965
StackOverflowException .....	967
TypeMismatchException .....	968
Classic Framework .....	969
Overview .....	969
Console .....	970
Overview .....	970
Desktop .....	975
Overview .....	975
Web .....	978
Overview .....	978
WebDragItem .....	980
Resources .....	981
System Requirements .....	981
System Requirements (2015r2.x) .....	985
Release Notes .....	988
2015r3.1 .....	989
2015r3 .....	990
2015r2.4 .....	1011
2015r2.3 .....	1012
2015r2.2 .....	1013
2015r2.1 .....	1014
2015r2 .....	1015
2015r1 .....	1022
Examples .....	1031
Videos .....	1045
Fun Stuff .....	1045
Xojo Wars 2015 Community Battle .....	1045
iOS .....	1046
Introduction to Xojo iOS .....	1046
Raspberry Pi .....	1047
Creating an LED Circuit .....	1047
Button LED Circuit .....	1048

---

Community .....	1049
Overview .....	1049
Blogs .....	1050
Books, Magazines, Training and Tutorials .....	1051
Discussion .....	1052
Free Source Code and Tools .....	1053
Open-Source Projects .....	1054
XojoTalk Podcast .....	1058
International Resources .....	1060
General .....	1060
Español .....	1061
Deutsch .....	1063
Italiano .....	1065
Français .....	1066
Japanese .....	1067
Português .....	1068
Nederlands .....	1069
Deprecations .....	1070
Feedback .....	1080
Recipes .....	1084
How do I work with files? .....	1084
How do I use databases? .....	1086

# How to Get Started with Xojo

If you are brand-new to Xojo, there are a variety of ways you can quickly get started.

Xojo is free to use for developing, testing and debugging. To get started:



1. [Download Xojo](#)
2. Launch Xojo and Sign In using your Xojo ID (that's the username and password you created at Xojo.com)

If you have any questions while using these resources, send an email (be sure to include your Xojo ID) to [hello@xojo.com](mailto:hello@xojo.com).

## Guides

If you prefer reading, you should work through the QuickStart guides which will give you a quick overview of the Xojo user interface and how to create a simple, working app. You should be able to work through each of the QuickStarts in 30 minutes or less. Even if you are familiar with other development tools, you are encouraged to take the time to go through at least one QuickStart to help you get familiar with how Xojo works.

- [QuickStart for Desktop apps](#)
- [QuickStart for Web apps](#)
- [QuickStart for iOS apps](#)

After completing these, you can move on to the Tutorial, which shows you how to create a slightly more sophisticated app with more code. Each Tutorial should take you less than an hour to complete.

- [Tutorial for Desktop apps](#)
- [Tutorial for Web apps](#)

Both the QuickStarts and Tutorials are also available in [multiple languages](#).

You should then move on to the [Introduction to Programming with Xojo](#) textbook. Full of examples, completing the textbook will give you a working knowledge of the fundamentals of programming. While you will be using Xojo, the concepts introduced in the textbook are applicable to any language you may choose to learn.

You can finish your Xojo voyage of discovery by reading the four books in the User Guide:

- [User Guide Book 1: Fundamentals](#)
  - Covers the Xojo IDE, its editors, programming language and object-oriented programming concepts
- [User Guide Book 2: User Interface](#)
  - Covers the user interface controls used by desktop and web projects.
- [User Guide Book 3: Framework](#)
  - Covers the "Classic" framework, including topics such as files, text, graphics, database, networking and

more. In addition, there are topics on debugging and profiling.

- [User Guide Book 4: Development](#)

- Covers app deployment, platform-specific features, source control, migrating from other tools (such as FileMaker, Visual Basic and FoxPro) and much more.

To learn about iOS development, check out the [iOS Guide](#).

Curious about Raspberry Pi? We also have a [Guide](#) for that.

## Videos

If you are more of a hands-on, visual learner, you can watch our many introductory webinars in the [Getting Started](#) section. Most webinars are about 45 to 60 minutes long. Some highlights include:

- [Creating a Desktop App Tutorial](#)
- [Creating a Web App Tutorial](#)
- [QuickStart for iOS](#)
- [Introduction to Programming 101](#)
- [Introduction to Programming 201](#)
- [Introduction to Object-Oriented Programming 101](#)
- [Introduction to Object-Oriented Programming 201](#)

## What if I have questions?

The best place to ask your questions is the [Xojo Forum](#). This forum is not like other forums you may have used. At Xojo we truly believe that our community is our greatest resource. The Xojo Forum is full of very active and helpful users, plus our own engineers who answer questions all the time. A quick search usually reveals a lot of common questions have already been answered, so be sure to take a look around before you post your questions as it might save you some time!

Log in to the forum using your Xojo ID and post your question in the appropriate "Channel" (aka topic).

## How can I learn more?

Use the Search of the Xojo Dev Center to find more about your topics of interest. You may also want to search the [Xojo Wiki](#), for information about the Classic framework that is used by Desktop and Web apps.

[Register for our regular Xojo webinars](#) to stay on top of your learning. And be sure to take a look at the over 60 webinar recordings that are available in the [Videos](#) section. Have a suggestion for a webinar topic? Be sure to [submit](#) it to us.

The [Xojo Blog](#) is a great place to get great tips and information about Xojo.

You can also subscribe to [xDev Magazine](#), which publishes six issues a year, each with about 80 pages of Xojo content. xDev has been publishing since 2002 and a large collection of back issues are also available.

For a more thorough understanding of programming concepts, you may want to take the [Stanford Programming Methodology course on iTunes U](#). This is a free, university-level, 21-hour course on many programming concepts, including: object-oriented design, decomposition, encapsulation, abstraction and testing.

## I want to build my app. Now what?

You'll need to purchase a license in order to build your apps to share with others. Licenses start at just \$99, which you can purchase in the [Xojo Store](#). A Xojo license gives you access to new Xojo releases for a 12 month period. After your Xojo license expires you may continue to build with those releases distributed during that 12 month period, indefinitely. You are never required to renew your Xojo license and can do so anytime before or after its expiration.

# iOS QuickStart



## About the QuickStart

Welcome to Xojo, the easiest tool for creating multi-platform desktop, web and iOS apps. Xojo is made up of a rich set of graphical user interface objects, a modern object-oriented language, an integrated debugger, and a multi-platform compiler. Together they make up the Xojo Integrated Development Environment or IDE.

With the IDE, you can build your app's interface simply by dragging and dropping interface objects into the app's layout views. In this QuickStart, you will see how easy it is. Xojo provides you with all the tools you need to build virtually any app you can imagine.

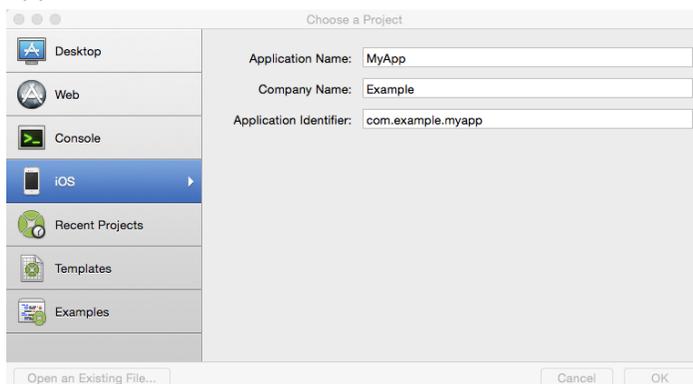
This QuickStart is for people who are new to programming and new to Xojo. It will give you a gentle introduction to the Xojo development environment and lead you through the development of a working iOS app (a simple web browser). It should take you about 15 minutes to complete this QuickStart. If you are short on time, you can also watch the 5-minute video or download the finished project:

- [iOS QuickStart Video](#)
- [Download SimpleBrowser project](#)

## Getting Started

If you haven't done so already, now is the time to start Xojo.

1. Double-click the Xojo application icon to start Xojo. After it finishes loading, the Project Chooser window appears.

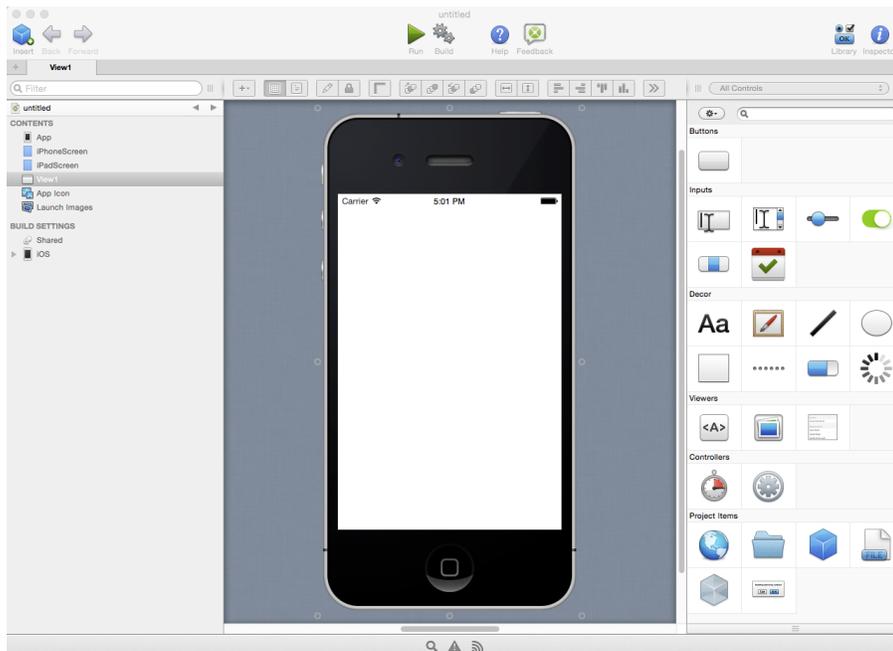


2. Xojo lets you build several different types of applications (Desktop, Web, Console and iOS). For this QuickStart, you are building an iOS app, so click on iOS.
3. You should now see three fields that need values: Application Name, Company Name and Application Identifier.
  - **Application Name** is the name of your app. This will be the filename of the actual app file that gets created.

- **Company Name** is the name of your company. You may choose to leave this blank.
  - **Application Identifier** is a unique identifier for this app. It will automatically populate using what you enter for the Application and Company Names, but you can also change it to whatever you want.
4. Enter "SimpleBrowser" as the Application Name. You can leave Company Name as it is or change it.
  5. Click OK to open the main Xojo window (called the Workspace), where you will begin designing your app.

## Workspace

Xojo opens the Workspace with the default view for your app automatically selected in the **Navigator** and displayed in the **Layout Editor**.



**Navigator:** The area on the top left shows you all the items in your project. By default you can see View1 (which is selected), the App object and the iPhone and iPad screen items. You use the Navigator to navigate to the various items within your project.

**Layout Editor:** The center area is the Layout Editor. You use the Layout Editor to design the user interface for the views in your app. It shows the view and previews how it looks when the app runs. In this illustration, the view is blank because you haven't yet added any user interface controls from the Library.

**Library:** The area on the right is the Library and shows the controls and interface elements that you can add to a view or to the project. You design the view by dragging controls from the Library to the view. You can also add a control to the view by double-clicking it. You can change how the controls display in the Library by clicking the small gear icon and choosing a different setting.

**Note:** If the Library is not visible, click the Library button on the toolbar to show it.

**Inspector:** Not shown in the above illustration is the Inspector, which allows you to see and change the properties for the selected control. This area of the Workspace window is shared with the Library. You can show the Inspector by clicking the Inspector button on the toolbar. The Inspector shows information about the selected item in the

Navigator or Editor. The contents of the Inspector changes as you click on different items. You change an Inspector value by entering a new value in the field to the right of the field label.

# Making the Simple Browser App

## Overview



The best way to quickly learn how to use Xojo is to create an app. For this QuickStart, you will create a simple web browser.

A Xojo app consists of a collection of objects, called classes. Nearly everything in Xojo is a class, including your views and its controls. In the SimpleBrowser project, you use the default View class to create your view and you add controls (user interface classes) to the view to create the design.

The app uses three controls:

- **Text Field:** A Text Field control is used to enter text. In this project, the URL to display is typed into a Text Field at the top of the view.
- **Button:** A Button is used to trigger an action. The user clicks the button to load the web page at the URL into the HTML Viewer.
- **HTML Viewer:** An HTML Viewer is used to display HTML (a web page). In this project, it is what displays the web site at the URL.

The next sections walk you through creating the user interface and adding the necessary code to make the app

work.

## Building the User Interface

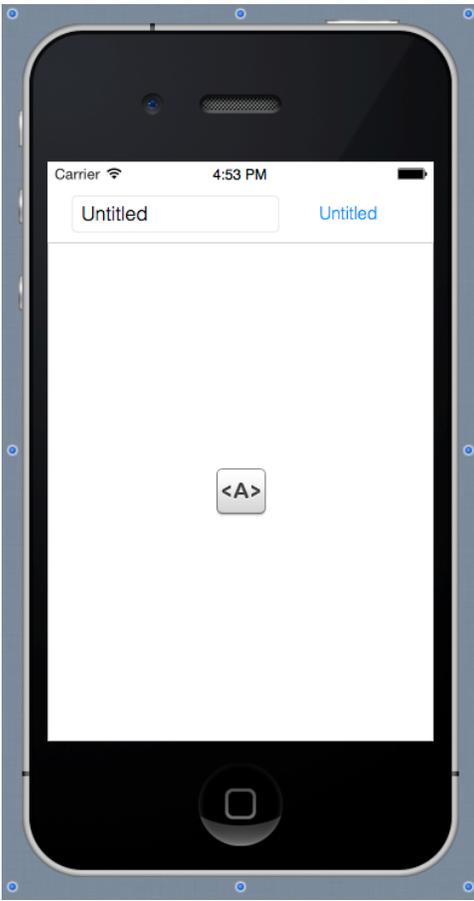
You should have Xojo running and View1 open in the Layout Editor. Now you are ready to start adding



controls to the view.

1. Starting with **Text Field**: In the Library, click on the Text Field icon and drag it to the top-left corner of the view in the Layout Editor. As you get close to the edges of the view, you will see alignment indicators that help you position the control.
2. Next, add the **Button**: In the Library, click on the Button icon and drag it to the top-right corner of the view.
3. The final control is the **HTML Viewer**: Drag the HTML Viewer icon to the remaining empty area on the view. Resize this control (using the selection handles so that it fills most of the view below the Text Field and Button) and use the locking guides that appear to help align it to the edges.
4. The final step is to resize the Text Field so that it is larger. Click on it to show the selection handles. Click the center-right handle and drag it to the right until the alignment guides tell you it is close enough to the Button.

Your finished view layout should look like this:



## Setting the Properties

### What is a Property?

A property is a value of a class. Changing property values allows you to change the behavior of the class. For this project, you want to change various properties for the view and the controls you added. Some of the things you need to do are:

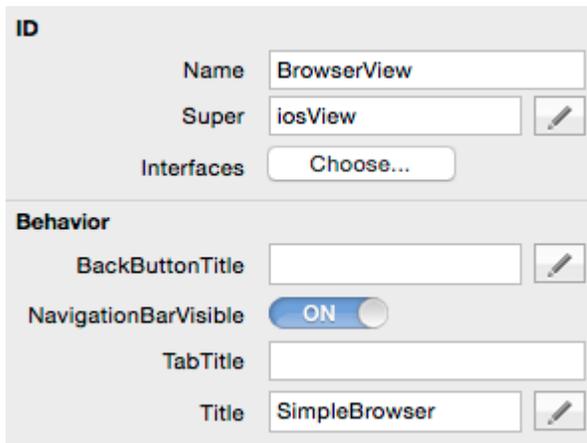
- Rename all controls (and the view) so that they describe their behavior and are easy to refer to in code.
- Add a Caption to the Button.
- Set Auto-Layout properties so that the controls resize properly for different sized iOS devices.

### Inspector

The Inspector is used to change view and control properties. It shares the same area on the right of the Workspace as the Library. In order to show the Inspector, click the Inspector button on the toolbar or press `⌘-I` (Ctrl+I on Windows and Linux).

### Setting the Properties for the View

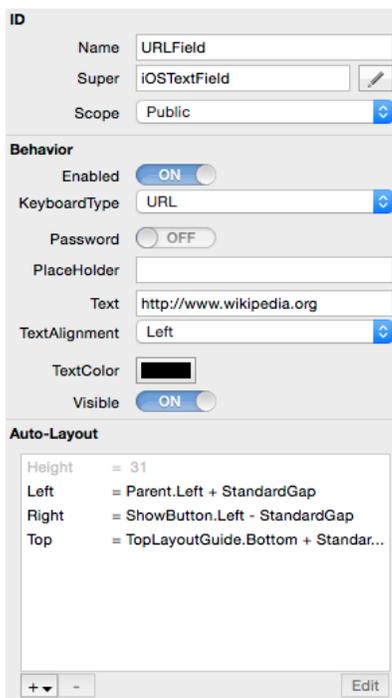
If you haven't already, display the Inspector by clicking the Inspector button on the toolbar and select View1 in the Navigator. You need to change the Name, `NavigationBarVisible` and `Title` properties.



- First, in the **Layout Editor**, click on the iPhone border of the view to select it. The Inspector now shows the properties for the view.
- In the **Name** field (located in the ID group), change the name from "View1" to "BrowserView". Press Return to see the name change in the Navigator.
- In the **NavigationBarVisible** field, set the switch to ON. This displays the Navigation Bar for the view. When you do this, the Text Field and Button may disappear as they are covered by the Navigation Bar. You can fix this in the next sections.
- In the **Title** field, change the name to "SimpleBrowser". Press Return and you will see the name change in the Navigation Bar.

## Setting the Properties for the Text Field

The Text Field is where your user enters the URL they want to see in the browser. You want to change the following properties in the Inspector: Name, KeyboardType, Placeholder, Text and Auto-Layout.

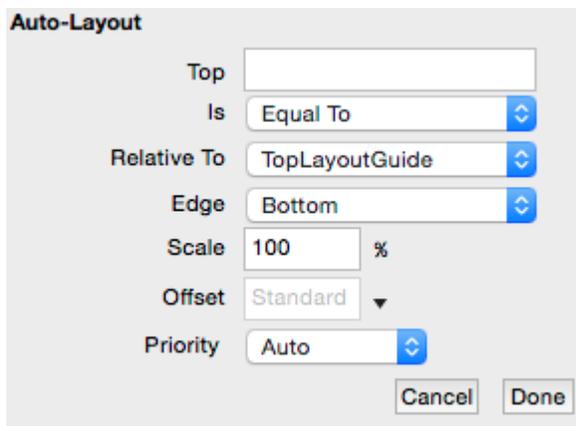


1. In the **Navigator**, select the TextField1 control on BrowserView. The **Inspector** changes to show the Text Field properties.

2. In the **Name** field, change the name from "TextField1" to "URLField". Press Return to see the name change in the **Navigator**.
3. In the **KeyboardType** field, select the value "URL" from the popup menu. This displays the special URL keyboard on the iOS device when the user taps in the field.
4. In the **Text** field, change the text from "Untitled" to "http://www.wikipedia.org".

If you did not align the Text Field using the layout guides, you may need to make changes to Auto-Layout so that the Text Field gets larger or smaller as the size of the view changes when different devices are used.

1. In the **Auto-Layout** section you will see a list of rules that control how the Text Field appears on the view. In particular, you want to change the Top rule so that it indicates to put the Text Field at a standard gap below the **TopLayoutGuide**. To do this, click on the "Top" rule and then click the Edit button.
2. Set the values for the Auto-Layout rule as below and click Done:



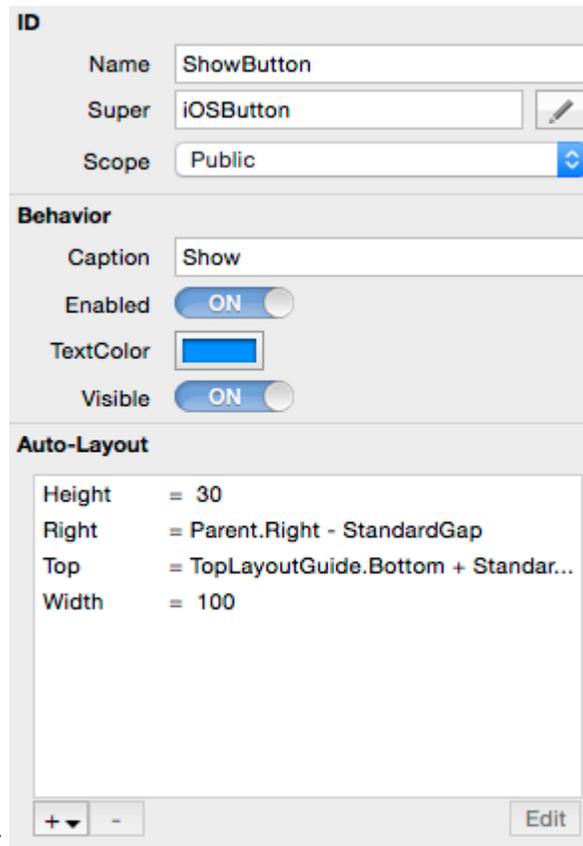
The screenshot shows the 'Auto-Layout' settings panel for a text field. The settings are as follows:

Top	<input type="text"/>
Is	Equal To
Relative To	TopLayoutGuide
Edge	Bottom
Scale	100 %
Offset	Standard
Priority	Auto

Buttons: Cancel, Done

## Setting the Properties for the Button

When running the app, clicking the button to display the web page. You need to change these properties: Name,

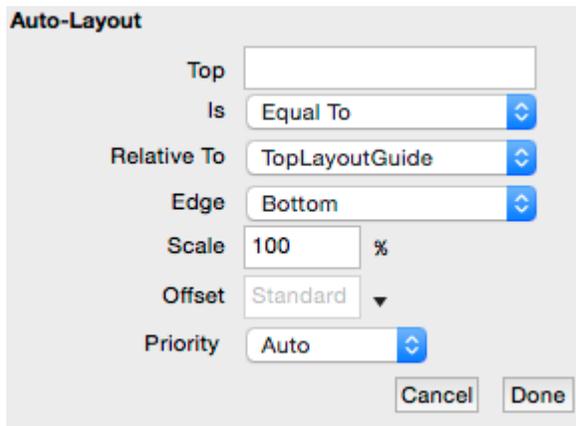


Caption and Auto-Layout.

1. On BrowserView, select the Button1 control. The Inspector changes to show the Button properties.
2. In the **Name** field, change the name from "Button1" to "ShowButton". Press Return to see the name change in the Navigator.
3. Give your button a caption by changing the **Caption** field from "Untitled" to "Show".

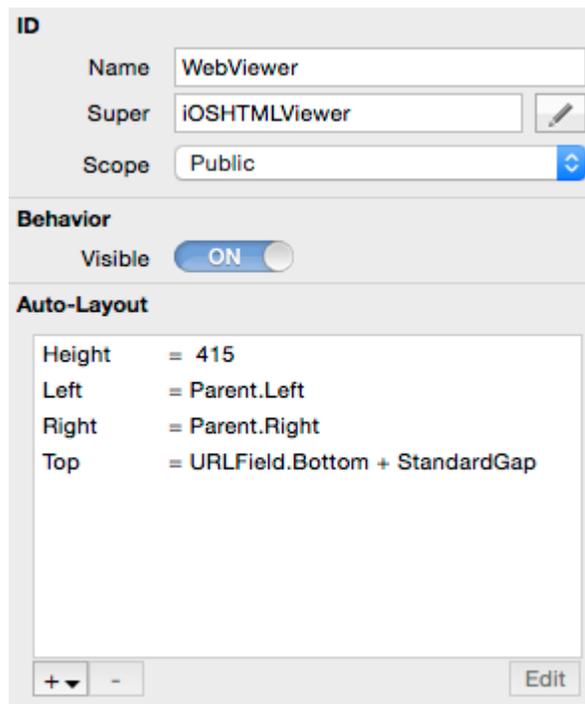
If you did not align the Button using the layout guides, you may need to make changes to Auto-Layout so that the Button stays below the navigation bar and stays attached to the right side of the view when the device size changes.

1. In the Auto-Layout section you will see a list of rules that control how the Button appears on the view. In particular, you want to change the Top rule so that it aligns the button with the top of the Text Field and the Right rule to ensure the Button stays on the right side of the view.
2. First click on the "Top" rule and then click the Edit button. If the Top rule is not displayed, you can add it using the "+" button at the bottom of the Auto-Layout area. Set the values for the Auto-Layout rule as below and click Done:
3. Lastly, click on the Right rule, click Edit, set its values as below and click Done:



## Setting the Properties for the HTML Viewer

The last user interface change you need to make is for the HTML Viewer. Here you need to change these properties: Name and Auto-Layout.



1. On BrowserView, select the HTMLViewer1 control. The Inspector changes to show the HTML Viewer properties.
2. In the **Name** field, change the name from "HTMLViewer1" to "WebView". Press Return to see the name change in the Navigator.

Finally, you may need to change Auto-Layout so that the HTML Viewer displays using the rest of the area in the View. Specifically you will change these Auto-Layout rules: Top, Left and Right (You may not have to change all rules if you used the layout guides to position the HTML Viewer).

1. Select the Top rule, click Edit and set its values as below:

**Auto-Layout**

Top

Is

Relative To

Edge

Scale  %

Offset

Priority

2. Select the Left rule, click Edit and set its values as below:

**Auto-Layout**

Left

Is

Relative To

Edge

Scale  %

Offset

Priority

3. Select the Right rule, click Edit and set its values as below:

**Auto-Layout**

Right

Is

Relative To

Edge

Scale  %

Offset

Priority

Remember, if any rule you want to edit is not in the list of rules, you can add it by using the "+" button at the bottom of the Auto-Layout section of the Inspector.

## Adding Code

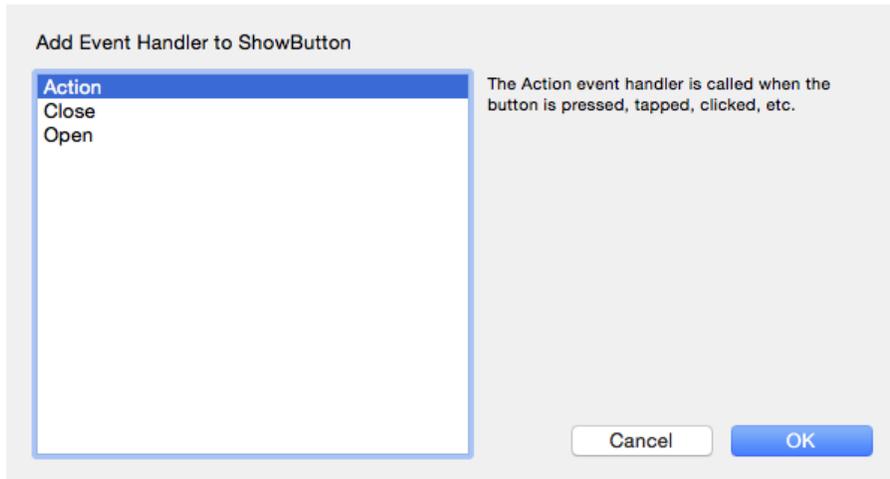
Your app is almost complete. Now it is time to add the code that will tell the HTML Viewer (called WebView) the web page to display. Xojo uses an object-oriented programming language which is quite easy to learn. You need only one line of code to finish your project!

The steps you need to do are:

1. Figure out when your user clicks the ShowButton, called "Show" on the view.
2. Get the URL that your user entered in URLField.
3. Have WebViewer display the URL.

Follow these steps to add the code:

1. On BrowserView, double-click the ShowButton control, labelled "Show".



2. The Add Event Handler window appears. Event Handlers occur when the user initiates an action. In this case, when a user clicks on a Button, your app runs any code in its Action event handler. This means you want to add your code to the Action event handler, so select Action from the Event Handler list and click OK. Notice the Navigator updates to show the Action event underneath the ShowButton control and the **Code Editor** displays. This step solves your first problem of when to know when the user clicks on the ShowButton.
3. Now you need to get the URL that the user typed. The value that a user types into a Text Field is stored in the Text property of the Text Field. You access the Text property like this  
`URLField.Text</code>`
4. The last step is to have WebViewer display the web page. This is done by calling the LoadURL method of the HTML Viewer control and sending it the URL that the user typed. Your code now looks like this:  
`WebViewer.LoadURL(URLField.Text)</code>`
5. Go ahead and add this code to the Code Editor. Start by clicking in the white space below the Action() event name and then type this code (do type it rather than copy and pasting it):  
`WebViewer.LoadURL(URLField.Text)</code>`

That's it! Your first app is complete.

## Running the App

Before you go any further, you should save your work:

1. Save the project by choosing File → Save As.
2. In the File Dialog, name the project "SimpleBrowser" and click Save.

## Running Your Project

Now you can test your finished app:

1. In order to run an iOS project, you first need to download and install Xcode in order to get the iOS Simulator app that is used to run iOS apps on OS X. [You can download Xcode for free from the Mac App Store.](#) After you have downloaded and installed Xcode, you need to run it one time to accept its License Agreement. After doing so, you can quite Xcode as you will not need it.
2. Click the Run button in Xojo to run the app in the iOS Simulator.
3. Type a URL of your choice (or use the default) and click the "Show" button.
4. You will see the web page.
5. When you are finished experimenting with the Simple Browser app, you can quit the iOS Simulator to return to Xojo.

## What's Next

This QuickStart has introduced you to Xojo and showed you how to make a simple app.

Next, you should view the the [iOS Guide](#) and explorer the [Reference Guide](#).

## 2015 Release 3 Highlights

This is one of the largest release of Xojo ever, with hundreds of fixes, changes and new features. Here are some highlights.

### iOSContainerControl

Like the `ContainerControl` for desktop projects and the `WebContainer` for web projects, the [`iOSContainerControl`](#) allows you to combine multiple control into a single control that you can easily re-use on your iOS views.

### File Type Sets

The [`File Type Set Editor`](#) has been complete recreated to better support Uniform Type Identifiers. You use this editor specify the types of files that your app can open and create, including custom icons.

### Collect Project Items

The [`Collect Project Items`](#) feature (File->Collect Project Items) puts all external project items (such as pictures) alongside the main project file for easier distribution.

### iOSLabel Wrapping

You now have more control over how the text in your iOSLabels wraps when it is too wide to fit. Refer to the [`LineBreakMode`](#) property and the [`iOSLineBreakMode`](#) enumeration.

### Code Bookmarks

Need to jump around to specific parts of your code? Now you can do it easily by setting a [`bookmark`](#) on a line of code.

## Web Drag & Drop

This highly requested feature allows controls on a WebPage to be dragged and dropped onto other controls on the page.

## Raspberry Pi

Xojo can now create desktop, web and console apps that run on Raspberry Pi! Read more about this in the [Raspberry Pi Overview](#) and take a look at our [LED Circuit tutorial](#).

## 64-bit

The signature feature of this release is that you can now create 64-bit version of your desktop, web and console apps for Windows, OS X and Linux. This gives you access to more memory, improves performance and makes deployment easier. Read the [64-Bit Guidelines](#) for more information.

## 64-bit Guidelines



Because 64-bit support involves new frameworks with a new, optimizing compiler, 64-bit is considered Beta for its initial release. Please read the [known issues](#) at the end of this page. For most projects, 64-bit builds will work just fine and we encourage you to try 64-bit builds for your projects. Should you find issues, be sure to report them to use using [Feedback](#).

Starting with 2015 Release 3, you can now create 64-bit app for OS X, Windows and Linux. In most cases, you'll be able to build a 64-bit version of your existing projects without having to make any changes.

With 64-bit, you apps gain access to much more memory than before (32-bit apps had a 3-4GB limit) and math-based code can see significant speed increases due to the new optimizing compiler. In addition, 64-bit operating systems can run 64-bit apps more efficiently than 32-bit apps since they do not have to load an extra compatibility layer.

## Building a 64-bit App

The Inspector for each build target (Windows, OS X and Linux) has a new property called Architecture where you can select the type of architecture you want to build. By default it is "x86 32-bit". To create a 64-bit app, check the box next to the target OS, choose "x86 64-bit" for the Architecture and then press the Build button.



You cannot Run (Debug mode) when the Architecture is set to "x86 64-bit". In order to debug, change the Architecture to "x86 32-bit".

64-bit builds using the LLVM compiler which can give you significant performance improvements for math-based code.

You may notice that it takes longer to build a 64-bit app than it does to build a 32-bit app. Because the new compiler is optimizing your app build, the compilation speed will be somewhat slower than the non-optimizing 32-bit compiler. At no point is the compiler "hung" or "stuck", so just wait for it to finish. Over time expect the 64-bit build times to be reduced as the compiler gets fine-tuned.



To make better use of hardware resources, the LLVM compiler uses multiple CPU cores when compiling.

## Considerations

### Compiler Constants

In a 32-bit app, `Target32Bit` returns True and `Target64Bit` returns False.

In a 64-bit app, `Target64Bit` returns True and `Target32Bit` returns False.

Use these two compiler constants to determine the type of app.

### Integers

in a 64-bit app, the `Integer` data type is an alias for `Int64`. In a 32-bit app, the `Integer` data type is an alias for `Int32`.

The type of app you need to create varies based on the operating systems you need to support.

### OS X

All versions of OS X supported by Xojo (10.7 or later) are already 64-bit. If your app is working properly as a 64-bit app, then you have no reason to continue creating 32-bit apps. Simply create a 64-bit app and make that available to your users.

### Windows

Windows is available in both 32-bit and 64-bit versions. The 32-bit version of Windows cannot run 64-bit apps, but the 64-bit version of Windows can run both 32-bit and 64-bit apps.

This means you will likely need to create both 32-bit and 64-bit versions of your Windows apps so that your users can choose the appropriate one for their version of Windows.

Keep in mind that the `TargetWin32` compiler constant returns True on Windows, regardless of whether the app is 32-bit or 64-bit. This constant is indicating that the Win32 system is being used and not an indicator of the type of system. To check if the app is 64-bit, use the `Target64bit` compiler constant.

## Build Files

For 64-bit builds, the compiler places framework DLLs next to the executable and not into the "MyApplication Libs" (or just "Libs") folder like it did for 32-bit builds.

A little history: For 32-bit Windows builds, the compiler doesn't actually spit out a native PE32 file. What it does is writes out a stub executable and then tacks all of the program data onto the end of it. The stub is responsible for loading that data into memory and performing all of the relocations and import binding that the OS loader would normally do. The stub doesn't have to link directly against any of the libraries referenced by the program (including the framework), so Xojo has the freedom to put the DLLs required by the program anywhere. However, this approach occasionally gets apps flagged as malware by antivirus programs and prevents some features Windows users have wanted for ages (like the ability to edit the app manifest).

For 64-bit Windows executables, the compiler is using a native linker to generate actual PE32+ files. These executables link directly against whatever DLLs it needs and are therefore bound to the operating system's search path. In order for the loader to find the DLLs the executable need to launch, they have to be in the same folder as the executable itself.

Attempting to the make the Windows build structure look more like OS X's bundles is unlikely to happen, primarily due to the DLL issue but also because in Windows, the directory is the application bundle . Microsoft expects programs to use installers and doesn't really care how 'messy' your program's folder is.

That said, the change in directory structure does not impose a requirement that installers are used or that it has to be on the system disk. The only change is that there will be more stuff in the folder with the executable.

## Linux

Linux distributions are also available in both 32-bit and 64-bit versions, although 64-bit versions are far more common. In fact, some Linux distributions are no longer making 32-bit versions available.

32-bit versions of Linux cannot run 64-bit apps. 64-bit versions of Linux can run 32-bit versions of app, but only if the appropriate 32-bit compatibility libraries have been installed. These libraries are not typically installed by default and are not always easy to install.

This means you will likely want to create both 32-bit and 64-bit versions of your Linux apps so that your users can choose the appropriate one for their version of Linux.

## Declares

In general, you'll want to review any [Declares](#) that are used in your project. In particular, if a Declare is using a specific Integer type, such as `Int32`, it may be possible that the library requires you to use `Int64` in a 64-bit app. To simplify this, use the Integer data type which is an alias to `Int32` in 32-bit apps and `Int64` in 64-bit apps.

On OS X, some Cocoa declare require a floating-point number. The 32-bit libraries use `Single`, but the 64-bit libraries use `Double`. There is no data type that functions as an alias for these, so you will have to create two separate Declare statements in these situations, separated with compiler constants.

A 64-bit app can only use 64-bit libraries. Declaring to a 32-bit library from a 64-bit app will fail. Similarly, a 32-bit app cannot use 64-bit libraries.

## Plugins

A 64-bit app requires plugins that support 64-bit. If your app uses plugins, you should check with your plugin vendor to see if they have updated versions that support 64-bit.

## IDE Script for Building

To make it easier to build all the different versions of your app in one step, you can use the following IDE Script:

```
// Build all target platforms
Const kOSX32 = 7
Const kOSX64 = 16
Const kWin32 = 3
Const kWin64 = 19
Const kLin32 = 4
Const kLin64 = 17
Const kLinARM = 18

Dim path As String
path = BuildApp(kOSX32)
path = BuildApp(kOSX64)
path = BuildApp(kWin32)
path = BuildApp(kWin64)
path = BuildApp(kLin32)
path = BuildApp(kLin64)
```

```
path = BuildApp(kLinARM)

Dim result As String
result = ShowDialog("Build All", _
    "Finished building.", _
    "OK", "", "", -1)
```

This script creates Windows, OS X and Linux builds for both 32-bit and 64-bit plus a Raspberry Pi build (ARM 32-bit) all in one step. You can comment out the parts you do not need to build.

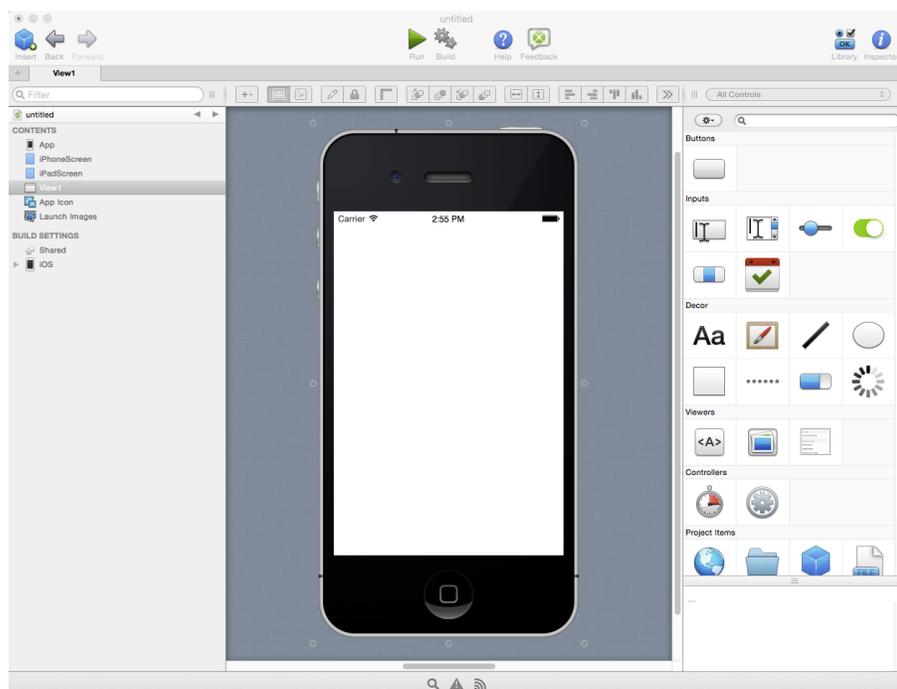
## Current Known Issues

These limitations currently exist with 64-bit apps. They will be resolved in future releases.

- You cannot run a 64-bit app in Debug Mode from the IDE. This means you cannot Run as a 64-bit app. To create a 64-bit app, you must Build.
- 64-bit builds can result in larger app sizes.
- For some projects, 64-bit builds can take a long time to finish. Be sure to let Xojo finish building.
  - To reduce build times, ensure you do not have really long methods that can be refactored and simplified. Project items (classes, Windows, etc.) with large amounts of code can also increase build times. Break them into separate classes to reduce build times.
- Windows 64-bit apps do not have version and icon information.
- XojoScript is not yet supported for any 64-bit app.
- You cannot build 64-bit OS X apps from Windows or Linux IDEs.
- The Tooltips class and tooltips on ListBox do not appear on 64-bit OS X apps.
- Long Split/Join operations on String make have issues. If you do not need multibyte encodings, you can use SplitB. Otherwise, convert the String to Text and then use the Text methods to Split or Join.

# Welcome to Xojo!

**Welcome to Xojo**, the easiest way to make desktop, web and iOS apps. With Xojo you can create desktop apps for Windows, OS X and Linux, web apps for a wide variety of web browsers and iOS apps for iPhones, iPads and other iOS devices.



Xojo makes it easy to build powerful, multi-platform desktop, web and iOS applications quickly and easily. If you are new to programming, you will find Xojo's object-oriented programming language easy to learn. If you are an experienced programmer, you will find the language to be powerful and robust. In either case, you will find you can accomplish quite a bit in a short amount of time.

Xojo is a rapid application development (RAD) tool with a user interface ("UI") builder that lets you create your app's user interface with little to no programming required. If you know how to drag and drop, you can build the UI using the wide variety of built-in controls.

Xojo's programming language is compiled, strongly-typed, and object-oriented. It uses the same type of modern architecture that most programming languages (like C++, Objective-C, Swift, and Java) are using today. Object-oriented programming languages make it easier to write and debug because the code is written as individual objects that are similar to objects in the real world.

Xojo makes application development faster and easier than traditional tools by removing the need to learn how to use the complex Application Programming Interface ("API") for the operating system. Instead, you use the Xojo Framework which provides a simpler and faster way to create your apps.

## Installing Xojo

You can download Xojo for your operating system (Windows, OS X or Linux) from the [Xojo Download page](#). You'll have to first create an account (your Xojo ID), but after that Xojo is free to use. You can purchase a license when

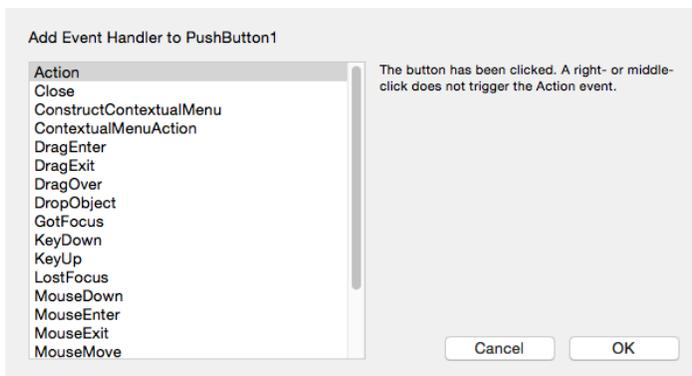
you are ready to create stand-alone apps.

After the download has completed, you can run the installer (or copy the files) for your OS. If you like, you can take a moment to review the [Xojo System Requirements](#) while Xojo installs. Once the installation is finished, you can start Xojo to begin creating your first app.

## Starting Xojo

After starting Xojo, the Project Chooser window appears. In order for you to quickly dive right into Xojo, follow these steps to create your first (admittedly simple) desktop app.

1. In the Project Chooser, choose "Desktop" and fill in the App Name as "MyFirstApp" and click OK. This displays the IDE (or Integrated Development Environment) with the Workspace for the project. This is where you work on your Xojo projects, creating UI layouts and writing code. Don't worry about everything that you see there, it will be covered later in the User Guide.
2. Now, drag a Button from the Control [Library](#) on the right onto the Window layout in the center. You can position the button anywhere you like within the window.
3. With the button selected, press Return to change its caption to "Click Me".
4. Double-click the button on the layout to add an Event Handler. This is where you'll put a single line of code to display a message when the button is clicked.
5. After double-clicking the button, the Add Event Handler window appears.

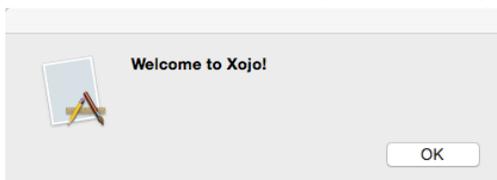


Select Action and click OK to add the Action event handler. This event handler is called when the button is clicked.

6. The Code Editor now appears. Add this simple line of code to display a message:

```
MsgBox("Welcome to Xojo!")</code>
```

7. Click the Run button on the toolbar to run your app.
8. Click the button to display the message.



That's it. You've now created an app. Although Xojo makes it easy to create simple apps, it can also create much larger and more powerful apps just as easily. In fact, Xojo is actually written in Xojo itself!

## About the User Guide

Xojo has a lot of features and capabilities. The User Guide is divided into several parts that each focus on a specific area of Xojo. The parts are: Fundamentals, User Interface, Framework and Advanced Topics.

In Fundamentals you'll learn about the [Xojo Integrated Development Environment \(IDE\)](#) and then move on to the Xojo Programming Language and object-oriented development concepts. It closes with chapters on App Structure, Debugging and Build Automation.

User Interface covers the Controls and Classes used to create the UI for Desktop, Web and iOS applications.

Framework book builds on what you learned in User Interface and Fundamentals. It covers the major framework areas in Xojo, including the Classic Framework (used by Desktop, Web and Console apps) and the Xojo Framework (used by Desktop, Web, Console and iOS apps). There are topics on files, text, graphics, databases, networking and much more.

As you might expect, Advanced Topics covers topics that are best used once you've master the material in the other parts. Topics include app deployment, platform-specific features, code management and more.

To grab the User Guide (and the entire Dev Center) as a PDF, click on the Print menu at the top of the Dev Center and select "Download Dev Center as PDF". You can also view the older User Guide PDF (which is in the process of being updated and moved to the Dev Center) here: [Classic User Guide](#).

## Getting Help

Learning a new tool and language takes some time. Even the most experienced developers may occasionally get stumped or have a question. When that happens, head on over to the [Xojo Dev Center](#) to find out more information. The Xojo Dev Center contains all documentation about Xojo, including the Reference Guide, this User Guide, over 70 video tutorials and links to many community resources. The Xojo Dev Center is online at <http://developer.xojo.com>. You can also display it in a window within the Xojo IDE by clicking the Help button on the toolbar.

A development tool is only as good as its community and Xojo has a wonderful community. One of the best places to learn about Xojo and ask questions is the [Xojo Forum](#) where you can interact with the larger Xojo community. Unlike

other forums you may have used, the Xojo Forum is a friendly and inviting place. The Xojo Community is known for helping answer questions and the Forum is a great place to hang out.

And if you like to learn by looking at existing code, be sure to check out the over [300 example projects](#) that are included by clicking on Examples in the Project Chooser.

# Fundamentals Overview

The Fundamentals part of the User Guide describes the [Xojo IDE](#) (Integrated Development Environment), the [Xojo Language](#), [Object-Oriented Programming](#), [App Structure](#), [Debugging](#) and [Build Automation](#).

You should start with this section if you are new to Xojo so that you can understand how the IDE is organized and the features it has. If you are new to programming the chapters on the Xojo Language and Object-Oriented Programming should be especially helpful. If you are not new to programming you may want to scan those chapters and then jump into the Reference Guide to see all the commands that Xojo has.

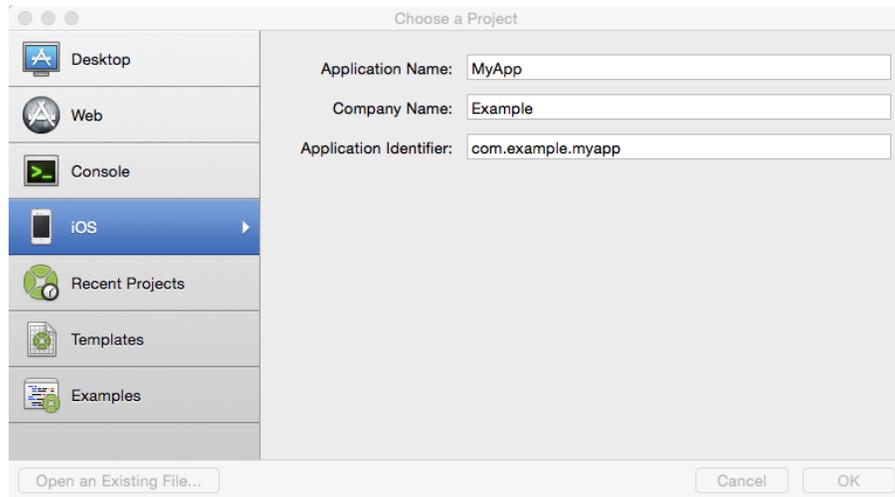
In the App Structure chapter you will learn how desktop, web, iOS and console apps are structured and built.

Following that is information on how to debug and profile your apps, for testing and performance optimization.

Lastly, the Build Automation chapter shows you how you can create scripts to automate your builds and control the IDE.

# The Project Chooser

The Project Chooser window displays each time you start Xojo and when you select File → New Project. With the Project Chooser window, you can choose to create a new project, work on a recent project, open any project file.



## New Projects

The Project Chooser lets you quickly decide what project you want to work with. You can choose to create a new blank Desktop, Web, Console or iOS project. Fill in the Application Name and Company Name to automatically create the Application Identifier (which is used by OS X, iOS and web apps).

## Recent Projects

You can view the most recently used projects and choose to open one of them. If the recent project is no longer available, clicking on it will prompt you to remove it.

## Templates

Templates are Xojo projects that are saved in the Templates folder alongside the Xojo application.

If you have a set of commonly used modules, classes or anything else, you can put them all in a project and then save it to the Templates folder. When you open the template, you get a new project with all your project items already included. Xojo includes templates for creating Service Applications, CGI Applications and Event-Driven Console Applications.

You can read more about Templates and how to create your own in the Project Types section.

## Examples

Select Examples to view the example projects included with Xojo. The examples demonstrate how to use specific features and functions of Xojo. When you choose an example, it is opened in a new project for you to run, edit or save where you like.

## Open an Existing File

You can open an existing Xojo project by clicking the Open an Existing File button. A file selector dialog will appear allowing you to choose the file to open. You can also drag a Xojo project onto the Xojo application icon (in the Dock or on the Desktop) or double-click a Xojo project to open it.

Once you have chosen the project to work with, the Xojo Workspace window is displayed.

# Project Workspace Overview

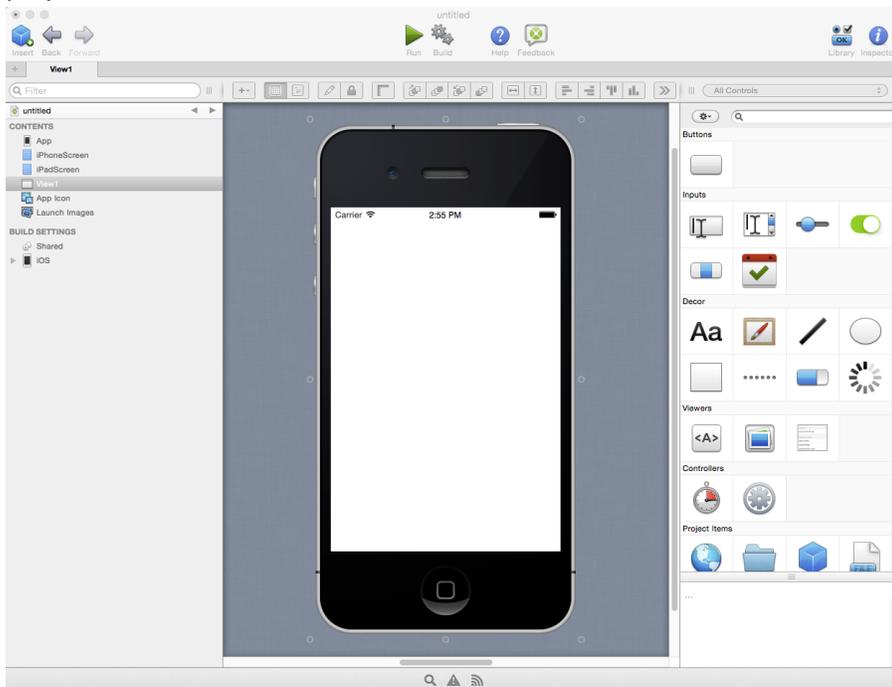
## Table of Contents

- [IDE](#)
  - [Toolbar](#)
  - [Full Screen Mode](#)

## IDE

As mentioned earlier, Xojo is an Integrated Development Environment (IDE). This means that all its components (layout editor, code editor, compiler, and debugger) are all integrated into one package. In traditional programming languages, these items would each be a separate application.

When you open Xojo and choose a project (new or existing), a single window, called the **Workspace** displays. Within this window you can navigate among project items by clicking on their names. You can also choose to open project items in their own tabs.



You have more than one project open at the same time; each will be shown in its own Workspace window.

 You can open multiple workspace windows for a single project by using **File → New Workspace**,

 which can be handy when working with multiple displays.

The Workspace consists of these areas:

- Toolbar
- [Navigator](#)
- Editor Area
- [Library/Inspector](#)
- [Find/Errors/Messages Panels](#)

## Toolbar

The toolbar at the top of the main window is called the Workspace toolbar. It has these buttons:

Insert	Adds an item to the project or to the selected project item.
Back	Moves to the previously used item in the tab.
Forward	Moves to the next used item in the tab.
Run	Runs your project using the debugger.
Build	Creates stand-alone applications.
Deploy	Builds and uploads your app to Xojo Cloud. This is only visible for web projects.
Help	Displays the Xojo Help window.
Feedback	Launches the Xojo Feedback application.
Library	Toggles the visibility of the Library pane.
Inspector	Toggles the visibility of the Inspector pane.

You can hide the toolbar by selecting View → Hide Toolbar.

## Full Screen Mode

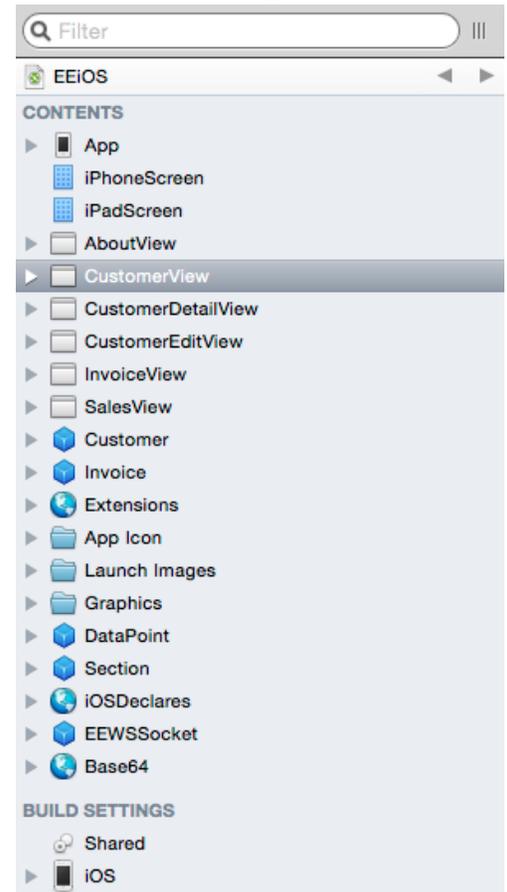
On OS X you can also click the “full screen” button in the window to put the Workspace into full screen mode. When in full screen mode, the window title bar and the main menu bar are hidden. To see the main menu bar, move the mouse cursor to the top of the screen; the menu bar will slide down. To exit full screen mode, click the full screen icon in the menu bar.

# Navigating Your Project

## Table of Contents

- [Filter](#)
- [Jump Bar](#)
- [Contents](#)
- [Run](#)
- [Profiles](#)
- [Build Settings](#)
- [Tabs](#)
- [Adding Project Items](#)
- [Go To Location](#)
- [Printing](#)
- [Working with Project Items in the Navigator](#)

The **Navigator** is the area on the left sidebar that allows you to move around (and organize) the items your project. The Navigator consists of the Filter and Jump Bar, plus the Contents, Run, Profiles and Build Settings sections.



The [Filter](#) is used to only show specific items in the Navigator's Contents section.

The Jump Bar is used to control the level of detail you see for an item in the project.

The Contents section contains the items in your project that were added using the Insert button or menu, such as windows, web pages, classes, menus and folders. You can click on project items in the Contents section to view them or edit them. When an item is selected, you can use the arrow keys to move the selection between items.

The Run section is only visible in the Debugger tab when you are running your project.

The Profiles section only appears when you have enabled the Profiler and have run your apps in debug mode to get profiler data.

The Builds Settings section contains all build-related information, including the build targets (and their settings) and active build steps used by Build Automation.

## Filter

The Navigator has a Filter field at the top that can be used to filter what is displayed in the Contents section. Use the Filter to quickly show specific project items based on your criteria. For example, if you know you have a method that is called “AddInvoice”, but you don’t recall what class or window it is on, you can just type “Add” in the Filter field. The Navigator will then display any project item that have something containing “Add” in its name (e.g. a method, control name, constant or property). You can then click on the project item to see it or edit it.

## Jump Bar

The Navigator displays items using a hierarchical list. The scope of what is displayed is controlled by the Jump Bar. By default, your entire project is in scope, so the Jump Bar displays your project name.

When you double-click on an item that is a parent, the Jump Bar changes to show the parent (this is referred to as “drilling into” the project item). Now only the items that are its children are displayed in the Navigator.

The “Double click opens item in new tab” setting in Preferences alters the above behavior. When  that setting is checked, double-click opens the item in a new tab, so use Option+⌘ when double-clicking on a project item to drill into it (on Windows and Linux, use Shift-Control).

Click the name in the Jump Bar to see the entire history and to jump quickly to a specific item in the history. The Jump Bar is incredibly powerful when used with tabs because it allows you to focus the tab on just the specific item

you are working with.

## Contents

The Contents section shows the items in your project. You can click on project items in the Contents section to view them or edit them. When an item is selected, you can use the arrow keys to move the selection between items. You can do the usual things such as deleting, copying or pasting project items. You can also drag project items to reorganize them or to move them between project items.

For example, you can drag a method from one class to another class to move it. In addition, objects can be multi-selected. For example, you can select a Class and a Window, and drag them both into a Folder.

## Run

The Run section appears in a separate tab when you run your project, which causes the App name to appear and displays the Debugger. If you navigate to another section of your project, clicking on this tab while a project is running displays the debugger.

## Profiles

If you run your app with Profiling enabled, the Profiles section appears when the app quits. You can review the Profile results in this section.

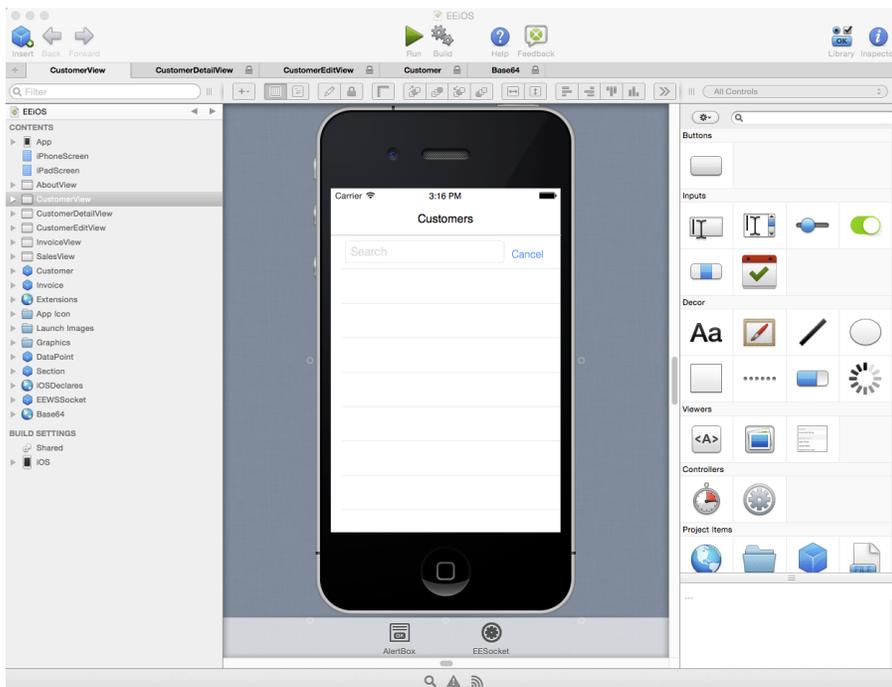
## Build Settings

The Build Settings section shows the various build OS targets available to you and is used to view and change the information needed to build and run your project. The item “This Computer” is checked by default and contains the settings for the platform you are currently using. Check the check box next to other targets to create a build for it the next time you build the project. Click on a build target name to change its settings using the Inspector.

The Build section also is where you manage your Build Steps, which are described in Advanced Topics.

# Tabs

A tab is simply another view into your project. When you create a new tab, you get new Navigator and Editor areas. You can navigate anywhere you want within a tab, just as you can when there are no tabs. Everything works exactly the same; you just now have multiple views into your project, each of which can show different information. Tabs can be a great way to keep frequently used items available, particularly when used in conjunction with the Jump Bar.



To open a project item in a tab, you can use the contextual menu and select “Open in New Tab”. You can also use Option+⌘ when double-clicking on a project item to open it in a new tab (on Windows and Linux, use Shift-Control). There is also a preference to have project items open in a new tab when double-clicked.

The “Double click opens item in new tab” setting in Preferences alters the above behavior. When that setting is checked, double-click opens the item in a new tab. Using Option+⌘ when double-clicking on a project item allows you to drill into it (on Windows and Linux, use Shift-Control).

Tabs can be locked or unlocked. A locked tab will not have its contents changed when you click on Filter or Search results, nor will it be changed when you use Go To Location. In those cases, a new tab (or the next available unlocked tab) are used. Click the small “x” in the tab to close it. Hold down Option (on OS X) or Alt (on Windows or Linux) when clicking the “x” to close all tabs but the left-most one.

If you open more tabs than can be displayed in the tab bar, an “overflow” icon appears for you to click to get a drop-

down list of the remain tabs. Select a tab from the list replaces the currently selected tab. You can switch between tabs (wrapping around at the beginning or end) using  $\text{⌘}+\text{Shift}+[$  or  $\text{⌘}+\text{Shift}+]$ .

## Adding Project Items

Use the Insert button on the toolbar or the Insert menu to add new project items (which appear in the Contents area).

You can also add new project items by dragging a control directly from the Library to the Navigator Content area. This creates a subclass of the control. Learn about subclasses in the [Object-Oriented Programming](#) chapter.

 Project items cannot be named “Xojo”.

These project items can be added to any type of project:

Build Step	Build Steps are used to <a href="#">automate the build process</a> .
Class	A <a href="#">class</a> is a container for code and is the fundamental building block for <a href="#">object-oriented programming</a> .
Class Interface	A <a href="#">class interface</a> is a contract that defines methods that specific classes must implement.
Database	Used to add a database object to your project.
Folder	A folder is a collection of project items and is used for organizational purposes only.
Module	A <a href="#">module</a> can contain members (methods, properties, etc.) that are global to your project. It can also contain project items, such as classes and even other modules, allowing you to control access and scope of your project items.

These project items are specific to desktop projects:

Container Control	Used to combine multiple controls to create custom or reusable controls.
File Type Set	Manages the files that your app uses.
Menu Bar	By default, MainMenuBar is added to all new projects. You can add additional menu bars as well, but this is not common.
Report	Create a report layout.
Toolbar	Design a toolbar that can be displayed on a Window.

Window	Desktop apps consist of Windows, which contain the controls that define your user interface.
--------	--

These project items are specific to web projects:

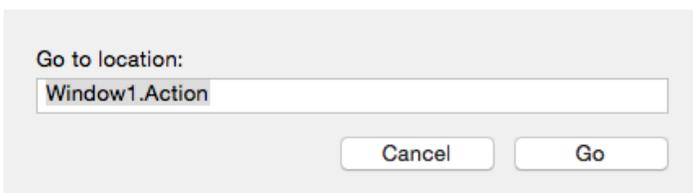
Container Control	Used to combine multiple controls to create custom or reusable controls.
Web Page	Web apps consist of Web Pages, which contain the controls that define your user interface.
Web Dialog	Define a dialog that can be displayed on a web page.
Web Style	Create a style that can be used to alter the look and feel of your user interface.

These project items are specific to iOS projects:

Container Control	Used to combine multiple controls to create custom or reusable controls.
Icon	Adds an Icon Set where you can specify an icon in various sizes to handle the different iOS screen sizes.
Image	Add an Image Set where you can specify an image in various sizes to handle the different iOS screen sizes.
View	iOS apps consist of Views, which contain the controls that define your user interface.
Screen	A screen defines the primary layout of the iOS screen and the views that are used.

## Go To Location

If you know its name, you can jump to a specific project item using the Go To Location feature. Select Project → Go To Location to display the Go To Location window. Enter the name of the project item you want to jump to and press Return (or click Go). The Navigator will select the item.



## Printing

You can print your source code using File → Print from the menu:

- When you have project items selected, only the selected items print.
- If you do not have anything selected, the entire project prints.

```
Project: Welcome

Date: Thursday, September 10, 2015 10:02:06 AM

Window1.Controls.PushButton1.Action

Sub Action()
    MsgBox("Welcome to Xojo!")
End Sub
```

## Working with Project Items in the Navigator

You can right-click (Control-Click on OS X) on any project item in the Navigator to display the contextual menu. From the contextual menu you have these options:

- **Add to**  
Use this command to add code items such as event handlers, methods, properties, etc. to the project item.
- **Inspect**  
Displays the Inspector for the project item.
- **Cut/Copy**  
Use to cut or copy the project item to the clipboard.
- **Paste**  
Pastes a project item in the clipboard into the Navigator, adding it to your project.
- **Delete**  
Deletes the project item. To put it in the clipboard, use Cut.
- **Duplicate**  
Creates a copy of the project item in the Navigator.
- **Make External/Internal**  
External items can be used to shared project items between your projects.
- **Encrypt/Decrypt**  
Refer to [Encrypting Project Items](#) for information on encrypting and decrypting project items.
- **Export**  
Use to export a project item to a file. This is a useful way to share a project item with someone else.
- **Print**  
Prints all the source code for the project item (see [Printing](#)).
- **New Subclass**

Creates a new class in the Navigator that uses the project item as its superclass. Learn about classes in [Object-Oriented Programming](#).

- **Implement Interface**

Lets you select an interface that contains methods to implement for the class. When you select the interface (or interfaces) to add, the methods from the interfaces are added to the project item. Learn about [Interfaces](#) in [Object-Oriented Programming](#).

- **Inspector Behavior**

Displays the Inspector Behavior window where you can customize the properties that display in the Inspector for classes and controls that you add to layouts. You can choose to display properties for your custom classes and subclasses and you can choose to hide properties that would normally be displayed. Add new properties using the “+” button on the left below the list of properties. Check or uncheck the check box next to the property name to determine if it is visible in the Inspector. You can also specify default values for any properties that are displayed. Drag properties in the list to reorder them or change their grouping in the Inspector. Use the Enumerations section to add a list of values that the user can select with a drop-down menu.

- **Edit Super Class**

Displays the parent (super) class for the currently selected class. This item only appears for subclasses. Learn about classes in [Object-Oriented Programming](#).

# Layout Editor

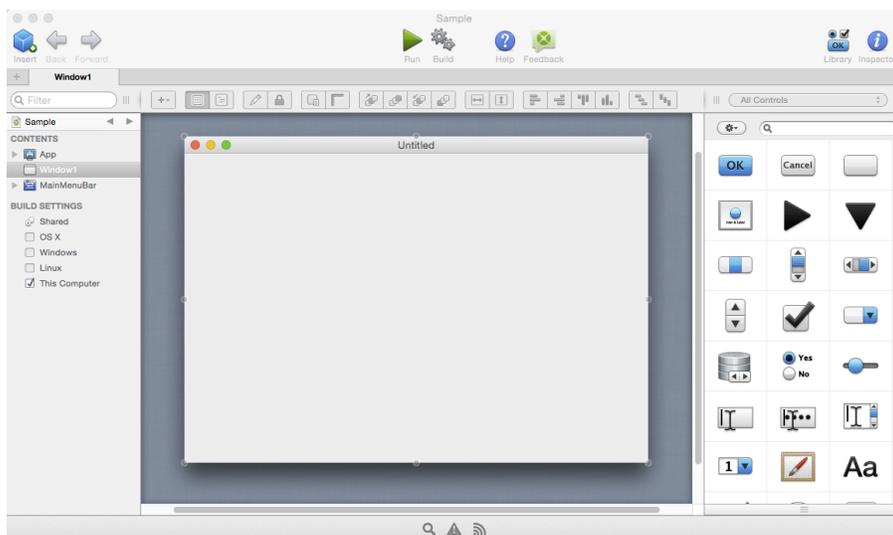
The Layout Editor is the primary editor you use to design the user interface for your application, whether it is a desktop, web or iOS app. In desktop apps, you use the Layout Editor to design your Windows and Container Controls. In web apps, you use the Layout Editor to design Web Pages, Container Controls and Web Dialogs. In iOS apps, you use the Layout Editor to design Views.

## Table of Contents

- [Layout Area](#)
- [Toolbar](#)
- [Shelf](#)
- [Default Values](#)
- [Event Handlers](#)
- [Alignment Guides](#)
- [User Controls \(Control Subclasses\)](#)
- [Working with Controls on the Layout](#)

## Layout Area

The Layout Area displays as either a window or a web page, depending on the type of project. In either case, you add controls to the area by dragging them from the [Library](#) or the [Navigator](#) onto the Layout Area.



## Toolbar

The Layout Editor has its own toolbar with the following features, in order from left to right:



### Add

The Add button is used to add code-related items to the window or web page. This includes:

- Event Handler (refer to the [Event Handlers](#) topic below)
- Menu Handler (desktop only)
- Method
- Note
- Property
- Computed Property
- Constant
- Delegate
- Enumeration
- Event Definition
- External Method
- Shared Computed Property
- Shared Method
- Shared Property
- Structure

### View Layout

This button is grouped with View Code and is a toggle. When viewing a Layout, this button is selected. When not viewing a layout, you can click this button to quickly switch back to the Layout Editor to see the last item you were working on.

### View Code

This button is grouped with View Layout and is a toggle. When viewing a Layout, this button can be used to switch back to the code editor for the last item being edited.

### Set Default Value

The Set Default Value button allows you to set the default value for various controls. Refer to the [Default Values](#) topic below.

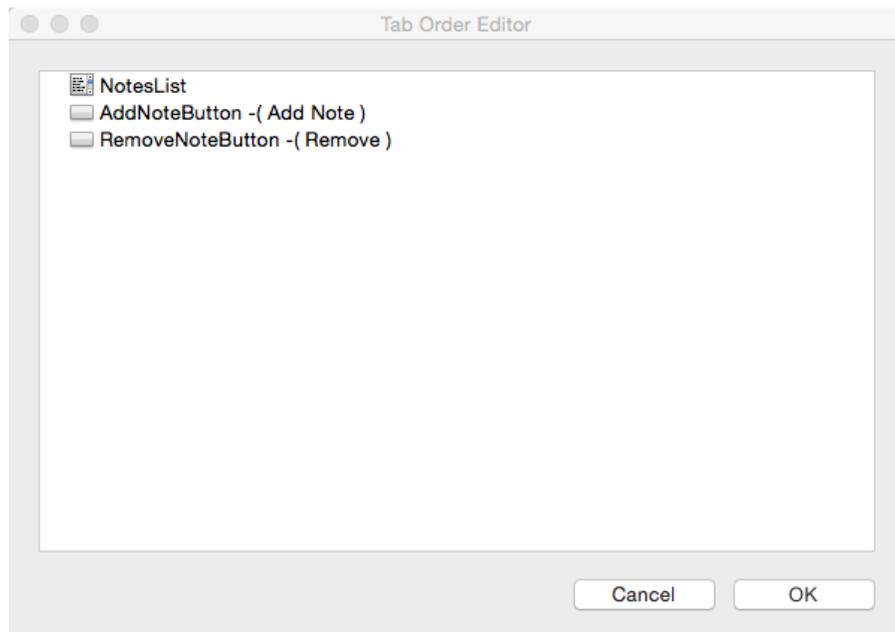
## Lock Position

The Lock Position button is used to lock controls so that they cannot be moved. You can use this feature to prevent your user interface from being accidentally changed while editing it.

## Show Tab Order

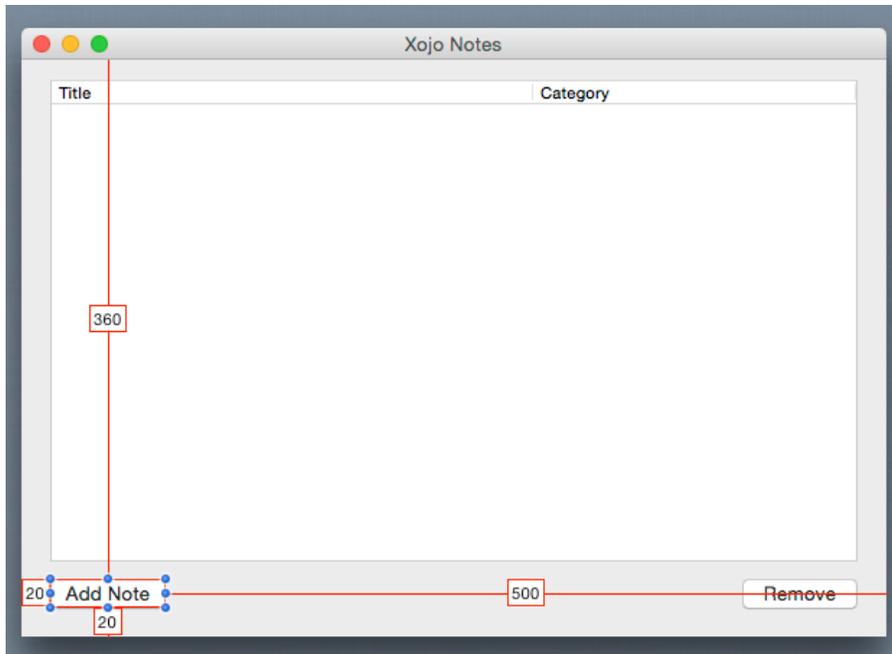
 Desktop and Web projects only.

Show Tab Order displays each control in the Tab Order Editor. The controls on the layout appear in the editor list in tab order. You can drag controls around to change their tab order on the layout.



## Show Measurements

The Show Measurements button allows you to better visualize your layout. When you click enable Show Measurements view, you can move the mouse around your layout to see various measurements, such as how far a control is from the top of the window or web page.



Select multiple controls to see various measurements, including distances between the controls.

## Ordering

The Ordering buttons (Order Forward, Order Front, Order Backward and Order Back) are used to change the ordering of the controls on the layout.

## Fill

The Fill Width and Fill Height buttons expand the selected control to fill the remaining space in its container.

## Alignment

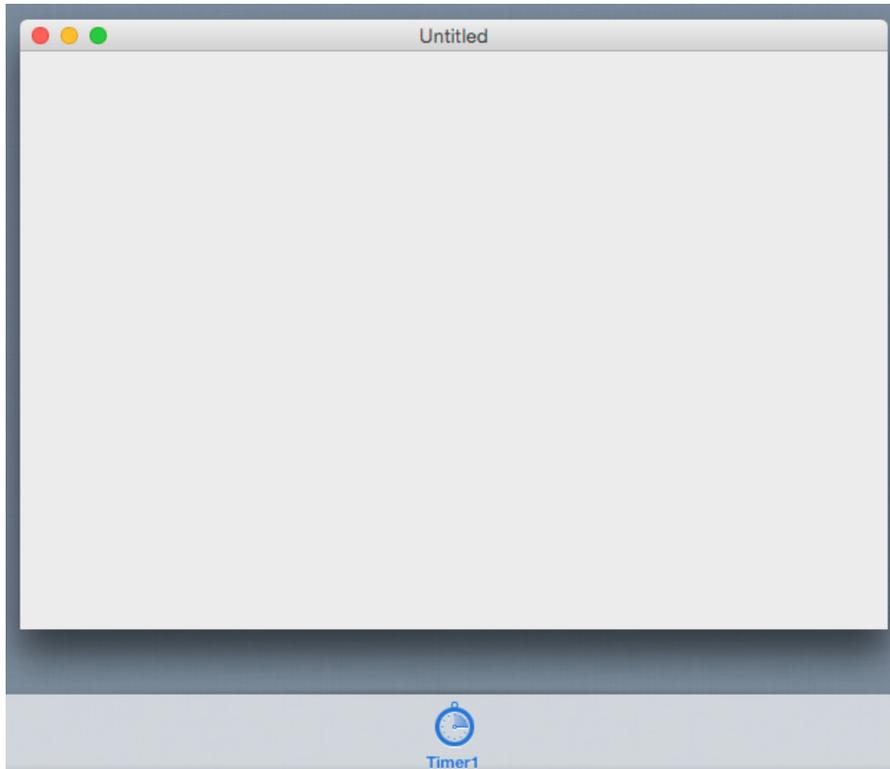
The Alignment buttons (Align Left, Align Right, Align Top and Align Bottom) are used to align controls on the layout with each other.

## Spacing

The Space Horizontally and Space Vertically buttons align the selected controls so that they are spaced equally apart.

## Shelf

Some controls that are added to the layout are not actually part of the window or web page, such as a Timer. When you add these types of controls to the layout, a Shelf is automatically displayed at the bottom of the Layout Editor and the control is added to it.

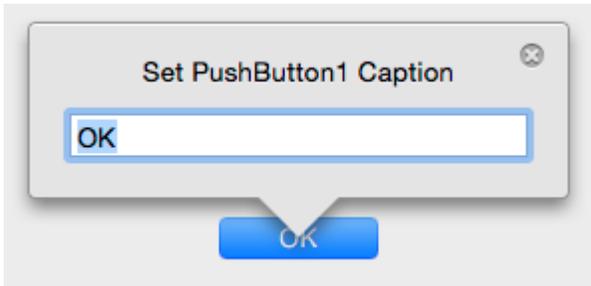


In web applications, Web Dialogs also appear on the shelf.

In desktop applications, a toolbar added to a Window appears on the shelf.

## Default Values

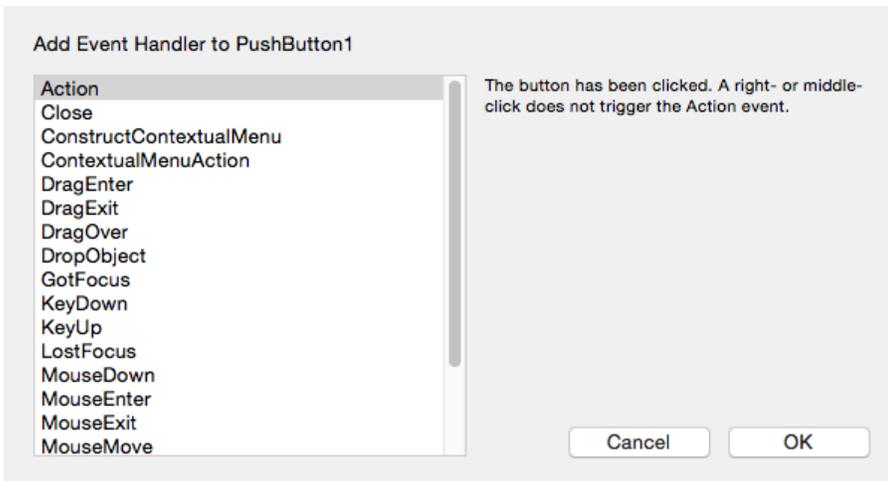
A control on the Layout Editor can have its default value specified by pressing Return while the control is selected, by clicking the Pencil rollover icon that appears when you move the mouse over a control, or by clicking the Set Default Value button on the Layout Editor Toolbar . This opens a popover window to enter the default value. For example, with a PushButton, you can specify the Caption. To close the pop-out window, click the small close icon, press Return, click outside the pop-out window or click the “Set Default Value” button on the Layout Editor toolbar.



## Event Handlers

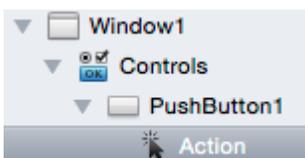
To add an event handler to your control, you click the Add button and select Event Handler.

This opens the Add Event Handler Dialog which displays the events that are available for the control (or the window or web page if that is what was selected). You can click on each event to view the description of the event. Select one or more events and click OK to create corresponding Event Handlers. You can also press ⌘-A (Ctrl+A on Windows and Linux) to select all the event handlers to add at once.



**i** You can add multiple events at once by selecting each of them and clicking OK.

Event handlers appear in the Navigator underneath the selected control. You can click on an event handler to see its code in the [Code Editor](#).



## Alignment Guides

As you move controls around on a layout you will see alignment guides appear to help with positioning. The guides appear as you drag the control near the edges of the layout or other controls. They are a great way to make sure that controls are aligned to the same positions as other controls on the layout.

To disable the display of alignment guides, hold down Command while dragging (Control on Windows and Linux).

## User Controls (Control Subclasses)

You can access user controls (subclasses of built-in controls) at the end of all the built-in controls in the [Library](#). If you have turned on "Show Group Banners" for the Library, these controls appear in the Project Controls group of the Library.

Or you can simply drag the subclass from the Navigator onto the Layout Area.

## Working with Controls on the Layout

You can right-click (Control-Click on OS X) on any control on the layout to display the contextual menu. From the contextual menu you have these options:

- **Add to**  
Use this command to select "Add Event Handler" from the submenu to add event handlers to the control.
- **Inspect**  
Displays the Inspector for the control.
- **Cut/Copy**  
Use to cut or copy the control.
- **Paste**  
Pastes a control in the clipboard onto the layout.
- **Delete**  
Deletes the control. To put it in the clipboard, use Cut.
- **Duplicate**  
Creates a copy of the control on the layout. You can also create a copy of a control by clicking on it with the mouse button and then holding Command (Shift+Control on Windows and Linux) while dragging the mouse.
- **Print**  
Prints all the source code in the event handlers of the control.
- **New Subclass**  
Creates a new class in the Navigator that uses the control as its superclass.

- **Select All**

Selects all the controls in the layout.

- **Select**

This submenu contains all the controls on the layout. Use it to find and select a control by name or to select a control that is not visible in the layout due to layering.

- **Lock/Unlock Position**

Locks the control at its position on the layout preventing it from being accidentally moved while editing the layout. To move the control, unlock it first.

- **Edit Superclass**

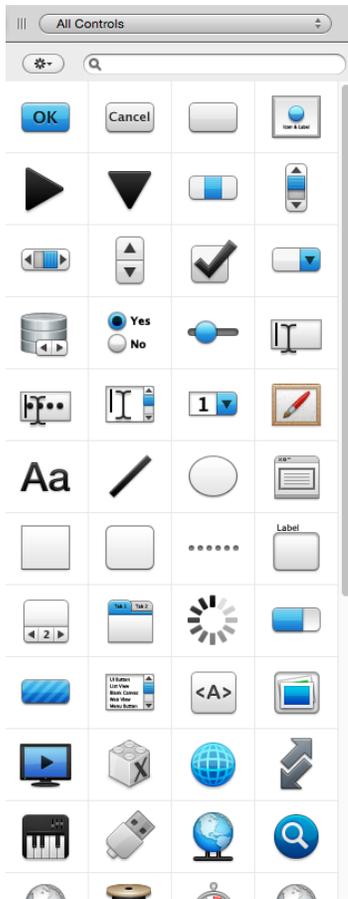
If the control has a superclass, then this command takes you directly to the superclass so you can edit it.

## Table of Contents

- [Library](#)
- [Inspector](#)
  - [Changing Properties](#)
    - [Numeric Properties](#)
    - [Boolean Properties](#)
    - [Text Properties](#)
    - [Constants](#)
    - [Choosers](#)
    - [Color Properties](#)
- [Palettes](#)

# Library

The Library is a list of all built-in controls that can be added to your layout for your project type. It also contains all the project items that you can add to your project. To show the Library, click the Library button on the toolbar, select **View** → **Library** from the menu or press the shortcut key (⌘-L on OS X, Ctrl-L on Windows and Linux). The Library displays on the right side of the workspace by default, but you can also set a preference to have it display as a floating palette.



The Library allows you to display controls in several ways. By default the controls are shown in Large size, but you can also change the size, add group headers and change the sorting. You can make these changes using the small “gear” icon at the top of the Library. Display settings include:

- Large Icons
- Large Icons and Labels
- Small Icons and Labels
- Large Icons and Descriptions
- Show Group Banners
- Sort Alphabetically

Hovering the mouse over any control displays its description in the Description Area at the bottom of the Library. If you do not need to see the description, you can resize the area to hide it.

You can also search the controls to quickly find the one you need. Using the drop-down at the top of the Library you can choose to show only the controls from a specific group. Or you can use the Search field (also at the top) to quickly search for and show controls by name or type. For example, type “button” to see all the button controls. Lastly, if a control is selected in the Library, you can just start typing to jump to specific controls.

You can drag controls from the Library onto the Layout Editor if it supports the control you have selected. You can

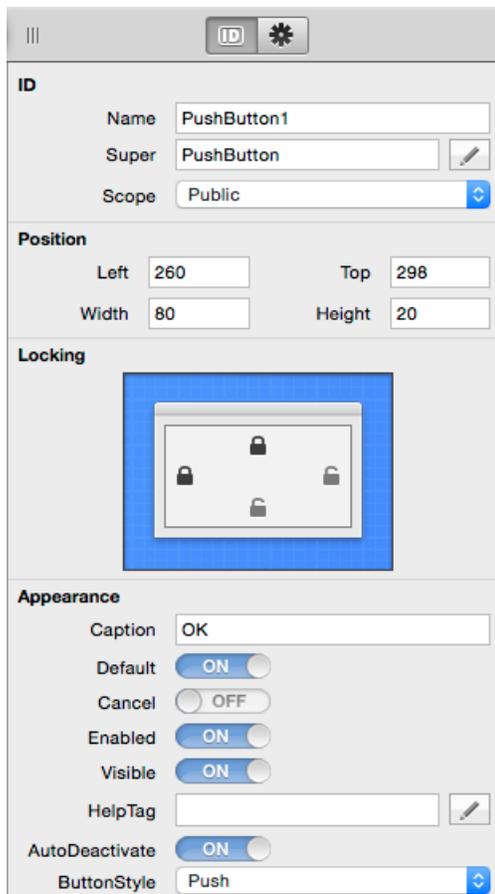
also drag controls directly to the Navigator to immediately create a subclass of the control. Lastly, you can also just double-click the control to add it to either the Navigator (as a subclass) or the Layout Editor, depending on which one has the focus.

At the very bottom of the Library (in a group called Project Controls, if you have group headings displayed) is a list of control subclasses from your project. This is an easy way to drag your own custom controls onto a layout.

## Inspector

The Inspector displays information (properties) for the currently selected item. This could be properties of controls on a Layout Editor, project items in the Navigator, Build Settings or methods or properties when using the Code Editor.

The Inspector shares its space with the Library on the right side of the workspace. To show the Inspector, click the Inspector button on the toolbar, select View → Inspector from the menu or press the shortcut key (⌘-I on OS X, Ctrl-I on Windows and Linux).



What you see in the Inspector depends on what you have selected. For controls in the layout editor, you will see all the properties available for selected control. These properties are grouped by topic to make it easy to find what you are looking for.

The Inspector may have more than one tab at the top depending on what is being viewed. More common items are contained in the "ID" tab and less commonly used (or advanced items) are in the "Gear" tab.

## Changing Properties

When more than one item is selected, the Inspector displays only those properties common to all of the selected items. When you change a common property, it is changed for all the selected items.

### Numeric Properties

If the property you want to set is numeric -- such as the Left, Top, Width or Height properties -- you can either enter a number or an expression that evaluates to a number.

To enter an expression, you can use the simple arithmetic operators: +, -, \*, /, \ (integer division), % (mod) and ^ (exponentiation). You can also use parentheses to control the order of evaluation. Additionally you can use references to property values by name, allowing you to write expressions such as "Top\*2" or "Width + Left".

If an expression is invalid on its own, the current value is prepended; this allows you to (for example) enter "\*2" as a handy shortcut for doubling the current value when multiple objects are selected. You add a value to the current value, use "+ + value". For example, to add 10 use "+ + 10", since "+10" will be treated as just 10.



These expressions are only evaluated once when the control first appears on the layout. They are not evaluated again (should you reference another property by name that changes, for example).

### Boolean Properties

The values of Boolean properties are displayed as ON/OFF switches in the Inspector. A value of False is indicated by OFF and a value of True is indicated by ON.

You change the value by clicking on the switch to toggle it.

## Text Properties

Text properties are entered simply by typing into the Text Field in the Inspector. If the text you want to enter is long, you can click the Text icon to open a larger window with a Text Area.

## Constants

You can use a constant as a value for a property. For example, you can create a constant that contains the caption for all the "Accept" buttons in your project. If you want to change the caption, you only need to change the value of the constant rather than edit the values for each button.

This technique is also useful for localization.

A good place to create constants for this purpose is in a [Module](#).

To use a constant for a property value, precede its name with "#" (pound sign). For example, if you want to use a global constant (in a module) named `AcceptButtonText`, you would refer to it in the Caption property of a button as `"#AcceptButtonText"`.

## Choosers

Some properties require you to choose a value from a fixed list, typically displayed as Pop-up Menus. Simply choose the value from the menu.

For controls that accept a picture as a property value, you can select the picture by choose it from the chooser for the property (all pictures in the project are added to the list automatically). The last menu item of the list of pictures is "Browse", which allows you to select a picture from your drive to add to the project and use.

## Color Properties

Color properties display as the selected color. Click on the color to display a Color Picker window for choosing another color.

# Palettes

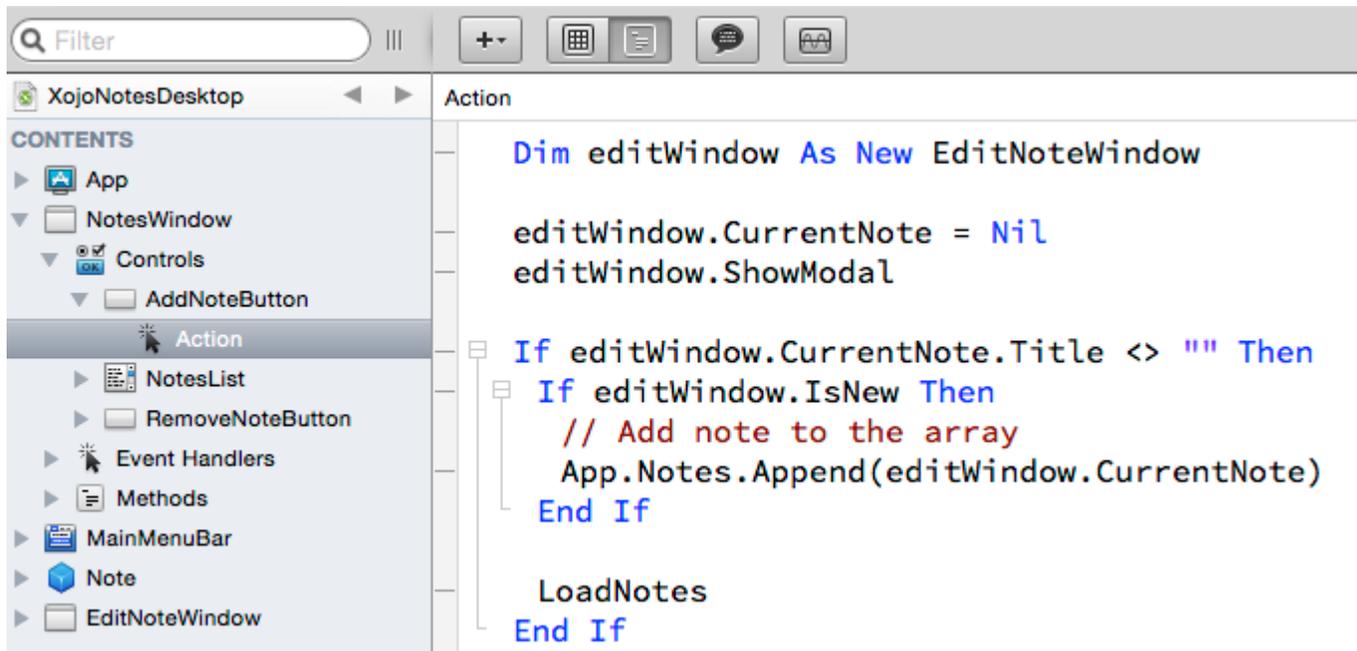
By default, the Library and Inspector display on the right side of the window and only one appears at a time. You can quickly toggle between the Library and Inspector using View → Toggle Palettes in the menu.

If you would prefer to position the Library and Inspector anywhere or have them both onscreen at the same time, you should use the preference to display them as separate Palettes, which you can then position however you want (even a second display).

In the General area of Preferences (Options on Windows and Linux) change "Show Library and Inspector" from "In the project window" to "As floating palettes".

# Code Editor

To create an application, you write code using the Xojo programming language in the built-in Code Editor. Code can be attached to nearly any item that is in your project, including windows, web pages, controls, classes, modules and more.



## Table of Contents

- [Editing Features](#)
  - [Syntax Highlighting](#)
  - [Auto Indentation](#)
  - [Auto-Complete](#)
  - [Syntax Help](#)
  - [Breakpoints and Bookmarks](#)
  - [Contextual Menu](#)
  - [Useful Keyboard Shortcuts](#)
- [Toolbar](#)
  - [Add](#)
  - [View Layout](#)
  - [View Code](#)
  - [Comment/Uncomment](#)
  - [Analyze Item](#)
- [Notes Editor](#)

## Editing Features

You can think of the Code Editor as a text editor that is specially designed for writing Xojo code. It has many features that will make it easy for you to program in Xojo.

### Syntax Highlighting

The Code Editor highlights your source code for you as you type. You can change the default colors for things such as keywords, string, integers, comments and more in Preferences.

### Auto Indentation

The Code Editor automatically indents your code as you type it. Code such as If...Then, While/Do loops, Select...Case and other commands that group code are indented for you automatically. Code that is indented is referred to as a **code block**. In the Code Editor, you can see that code blocks have a small "-" sign to their left.

You can click this to collapse the code block (changing it to a "+'), hiding all the code within it. The code will still run and is part of your project, it is just not visible in the Code Editor. Click the "+" to expand a code block.

```
[-] If editWindow.CurrentNote.Title <> "" Then
  [-] If editWindow.IsNew Then
    // Add note to the array
    App.Notes.Append(editWindow.CurrentNote)
  End If

  LoadNotes
End If
```

```
[-] If editWindow.CurrentNote.Title <> "" Then
  [-] If editWindow.IsNew Then ... End If

  LoadNotes
End If
```

Click on the vertical line connecting the code block to select all the code in the block.

```

If editWindow.CurrentNote.Title <> "" Then
  If editWindow.IsNew Then
    // Add note to the array
    App.Notes.Append(editWindow.CurrentNote)
  End If

  LoadNotes
End If

```

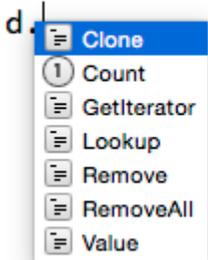
Expanding and collapsing code is a great way to hide code that you do not need to see when focusing on other code.

## Auto-Complete

As you are typing in the Code Editor you will often see three small ellipses that appear after the text you are typing. This indicates that the Code Editor has some auto-complete suggestions for you. When you see the ellipses, press the tab key to display a list of available commands that can be auto-completed. The auto-completion is smart and only tries to show you commands that are relevant based on the context of your code.

When the auto-complete list is displayed, you can choose the item you want using the mouse or you can continue typing to automatically move the selection in the list. When you've reached the selection you want, just press return. Here, "d." was typed and when the tab key was pressed the auto-complete menu appeared showing the methods and properties of a Dictionary that are relevant:

```
Dim d As New Xojo.Core.Dictionary
```



Auto-complete may also show you the remaining part of the text you are typing (in gray). Press tab to have the text fill in automatically. This can be much faster than typing out the name manually, especially if you have long variable, class or method names.

```
Dim d As New Dictionary...
```

## Syntax Help

There is a small area at the bottom of the code editor that is used to display syntax help. The Syntax Help Area displays the line number, column number, syntax information, method signatures, declarations and other information about the code that is under the mouse cursor.

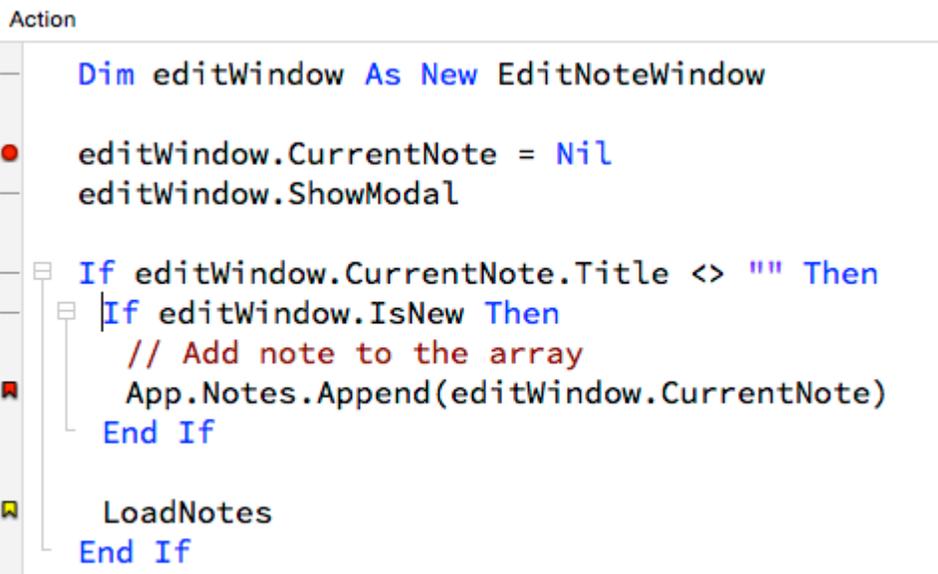


(18, 6) xojo.Core.Dictionary.Value(key As Auto) As Auto

## Breakpoints and Bookmarks

The gutter on the left side of the Code Editor is used to set **breakpoints** and **bookmarks**. A breakpoint is a line of code that will stop execution when your app is run in the debugger. A bookmark is a quick way to jump to specific code. Next to each line of code, you will see a dash ("-") in the gutter indicating that this is a line that can have a breakpoint or bookmark. Click on the dash to toggle a breakpoint on or off. A breakpoint is indicated by a red circle.

Option-click on OS X (or Control-click on Windows/Linux) to toggle a bookmark on or off. A bookmark is indicated by a bookmark graphic. If you have both a breakpoint and a bookmark set for the line, the indicator appears as a red bookmark graphic.



```

Action
Dim editWindow As New EditNoteWindow

editWindow.CurrentNote = Nil
editWindow.ShowModal

If editWindow.CurrentNote.Title <> "" Then
  If editWindow.IsNew Then
    // Add note to the array
    App.Notes.Append(editWindow.CurrentNote)
  End If

  LoadNotes
End If

```

The Project → Breakpoint menu has menu items to turn a breakpoint on or off, show all the breakpoints or remove all the breakpoints. Similarly, Project → Bookmarks has menu items to turn a bookmark on or off, show all the

bookmarks or remove all the bookmarks. When you Show All bookmarks or breakpoints, they appear in the [Find panel](#) at the bottom of the Workspace.

## Contextual Menu

You can right-click (or control-click) on a variable, method, property, class or any item in your project to display the contextual menu, which offers several useful features:

- **Cut/Copy/Paste/Delete**  
Provides the usual cut, copy, paste and delete functionality.
- **Select All**  
Selects all the text in the code editor.
- **Comment**  
Comments (or uncomments) the selected code using the comment character you have specified in [Coding Preferences](#).
- **Insert Color**  
Opens the Color Selector window and allows you to choose a color, which is added to the code editor as a color literal.
- **Go To**  
Jumps to the definition of the selected item in the code editor. This works for variables, methods, classes and any other item you have created in your project. You can also quickly jump by double-clicking on a variable, method, class while holding down the command key (OS X) or control key (Windows and Linux).
- **Switch To**  
Switches to another event, method or property.
- **Standardize Format**  
Applies standard formatting rules to the selected text.
- **Wrap in If/End If**  
Wraps the selected code in an If Then / End If code block.
- **Wrap in Do/Loop**  
Wraps the selected code in a Do / Loop code block.
- **Wrap in While/Wend**  
Wraps the selected code in a While / Wend code block.
- **Convert to Method**  
Moves the selected code to its own method and switches you to the method so you can specify a name for the method.
- **Convert to Constant**  
Moves the selected to its own constant definition and switches you to the constant.
- **Define Missing Method**  
Creates a new method using the selected method as the method name. If you have supplied parameters, their types are automatically added to the method definition.
- **Clean invisible ascii characters**  
When cutting and pasting code, sometimes invisible ASCII characters can confuse the code editor. Use this command to clean up the selected text.
- **Turn Break Point On/Off**  
Turns the Break Point for the current line on or off.
- **Help for / Open Language Reference**

Opens the Documentation window. If the command at the cursor is recognized as a Xojo command, then “Help for” appears, allowing you to jump directly to its entry in the documentation.

## Useful Keyboard Shortcuts

While working in the code editor, there are several keyboard shortcuts that will help make you more productive.

- Ctrl + Return (⌘-Return on OS X) completes the code block. For example, typing “If True” and pressing Ctrl + Return will automatically fill in the “Then” and the “End If” and place the cursor between them. This works for other code blocks such as Select, While, Do and For.
- Ctrl + Enter (Option-Enter on OS X) extends code to a new line by automatically adding the “\_” character.
- Ctrl + \ (⌘-\) toggles the breakpoint on the line on or off.
- Ctrl + ' (⌘-') comments or uncomments the selected code lines or the line the cursor is on if no code is selected.

Also refer to the [Full Keyboard Shortcuts](#).

## Toolbar

The Code Editor has its own toolbar with the following features, in order from left to right:

### Add

The Add button is used to add code-related items to the window or web page. This includes:

- Event Handler
- Menu Handler (desktop projects only)
- [Method](#)
- Note
- [Property](#)
- [Computed Property](#)
- Constant
- Delegate
- [Enumeration](#)
- Event Definition
- Shared Computed Property
- Shared Method
- Shared Property
- [Structure](#)
- Using Clause

## View Layout

This button is grouped with View Code and is a toggle. When viewing code, you can click this button to quickly switch back to the Layout Editor to see the last item you were working on.

## View Code

This button is grouped with View Layout and is a toggle. When viewing code, this button is selected. When not viewing code, you can click this button to quickly switch back to the Code Editor to see the last item you were editing.

## Comment/Uncomment

Use this button to comment out the selected code in the code editor. If no code is selected, then the current line is commented out. If the code is already commented, then this will uncomment the code.

This uses the comment character you have specified in [Coding Preferences](#).

## Analyze Item

Use Analyze Item to validate the source code for the current project item. Analyze checks the project item for compile errors and warnings (such as unused variables). Refer to [Analyzing the Project](#) for more information.

## Notes Editor

The Notes Editor is a simple text editor where you can enter notes related to the project item. Uses for Notes:

- technical documentation
- long comments
- commonly used source code
- design documentation

## Other Editors

Although you will spend most of your time in the Layout and Code Editors, there are several other editors used for features that are project-specific, such as for menus, toolbars, reports, file type sets and containers.

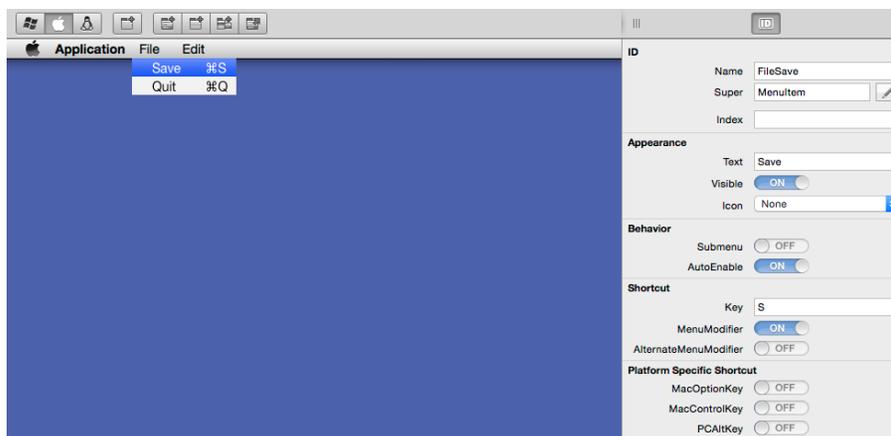
### [Table of Contents](#)

- [Menu Editor](#)
- [Toolbar Editor](#)
- [Report Layout Editor](#)
- [Container Control Editor](#)

## Menu Editor

 Desktop projects only

The Menu editor makes adding menu bars, menus, and menu items to your desktop projects easy. The standard Desktop Application template includes a menu bar that is used as the default menu bar for the entire application (MainMenuBar). You can use the default or create additional menu bars (using Insert on the toolbar or menu) for use with specific windows by assigning the menu bar to the Menu Bar property using the Inspector for the Window.



The default menubar for a Desktop Application, MainMenuBar, includes File and Edit menus and the standard File and Edit menu items. The File menu has one menu item, Exit (on Windows) or Quit (on OS X and Linux). The properties of the Quit/Exit menu item are supplied, so that the menu item works automatically. You don't need to modify or add to the menu item's properties in order to enable it.

Similarly, the Edit menu is populated with Undo, Cut, Copy, Paste, Delete, and Select All menu items.

You can drag menus around to rearrange them or move them to entirely new menus. The toolbar has functions for adding new menus and menu items.

You can also use Cut/Copy/Paste to move and copy menu items to other areas of the menu.

## Toolbar

The toolbar provides these commands:

- **Platform Display**  
The platform icons on the toolbar allow you to see how the menu looks on Windows, OS X and Linux.
- **Create New Top-Level Menu**  
Adds a top-level menu to the menu bar.
- **Create New Menu Item**  
Creates a new menu item under a selected top-level menu .
- **Create Separator**  
Creates a separator item below the selected menu item.
- **Create Submenu**  
Creates a submenu below the selected menu item.
- **Convert Select Menu to a Top-Level Menu**  
Use this command to convert the selected menu item to a top-level menu.

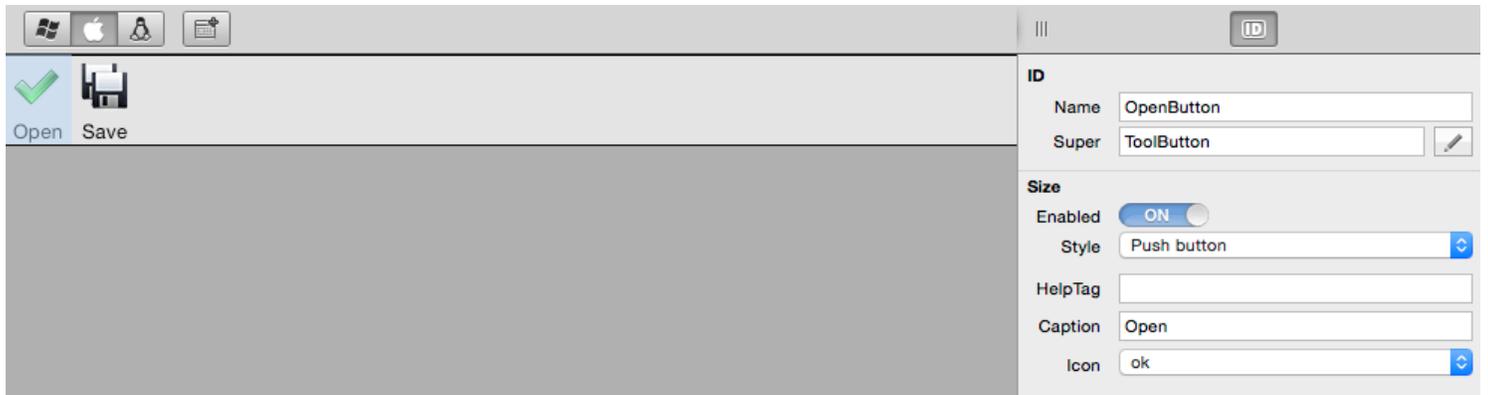
## Inspector

The Inspector displays the properties for the selected menu or [menu item](#).

## Toolbar Editor

 Desktop Projects only

The Toolbar Editor is used to design toolbars for your desktop apps. In this editor, you can drag the buttons around to reorder them.



Once you have created a toolbar, you can add it to a Window by dragging it from the Navigator onto the Window layout where it will appear in the [Shelf](#).

## Toolbar

The toolbar has these commands:

- **Platform Display**  
The platform icons on the toolbar allow you to see how the toolbar looks on Windows, OS X and Linux.
- **Add**  
Adds a new item to the toolbar. Once added, you can drag it to another position on the toolbar.

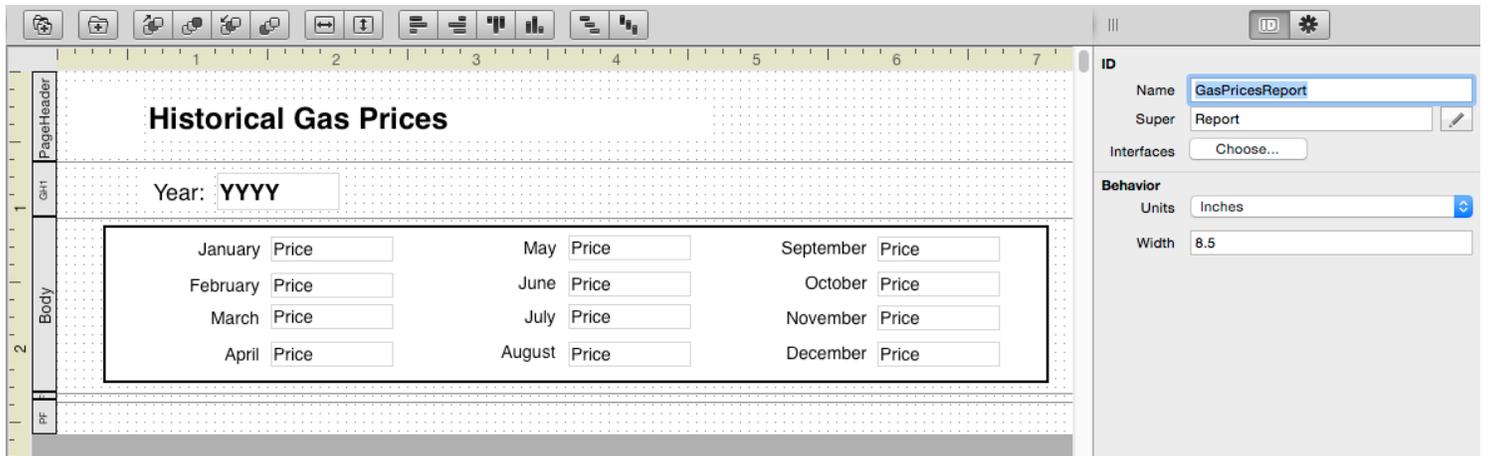
## Inspector

The Inspector displays the properties for the selected toolbar item.

## Report Layout Editor

**i** Desktop Projects only

The Report Layout Editor is used to design reports for desktop apps.



## Toolbar

The toolbar has these commands:

- **Add Group**  
Adds a new group section around the body.
- **Add Page Header/Footer**  
Adds a new page header and footer to the report.
- **Ordering**  
Specify the ordering of the controls on the report.
- **Fill**  
Fill the report control to available space.
- **Alignment**  
Adjust report control alignment.
- **Spacing**  
Adjust report control spacing.

## Library and Inspector

The Library displays the report controls that can be added to the report layout. The Inspector displays the properties for the selected report control.

## Container Control Editor

Container Controls are special project items that allow you to combine multiple controls into a single control to add to a layout. The Container Control Editor works the same as the [Layout Editor](#) for Desktop, Web and iOS projects.

# Find/Errors/Messages Panels

There are three buttons at the bottom of the workspace that open panels for Find, Errors and Messages. The panels may also be opened automatically as needed, such as when there are compiler errors or you want to show all bookmarks.

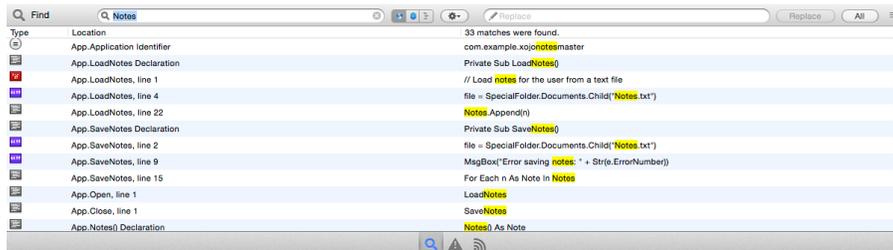
The Panel area is closed when the project is Run.

## Table of Contents

- [Find](#)
- [Errors](#)
- [Messages](#)

## Find

The Find Panel is used to search (and optionally replace) text in your project. It searches instantly as you type text in the Find field.



You can use the scope control to change the scope of the search and the “gear” button to change the matching criteria. The scope choices are:

- Global: Searches the entire project (this is the default).
- Current Project Item: Searches just the currently selected project item in the Navigator.
- Current Method or Event: If the Code Editor is visible, this searches just the text in the Code Editor.

By default your search text is matched as case-insensitive and will find partial matches. This means that searching for "List" will find things like "myList" or "NameListBox". You can change the matching criteria:

- Whole Word  
Only searches for your text as a whole word.
- Match Case  
Does a case-sensitive search.

- Use RegEx  
Use a Regular Expression to search for matching text.

Click once on a Find result to jump to where it is in your project.

 Find only searches the items displayed in the Navigator. If you have filtered the Navigator using either the [Filter](#) or [JumpBar](#), then Find only finds based on what is displayed.

## Errors

The Errors Panel displays compiler errors and warnings. This panel appears automatically when you Run or Build if there are compiler errors. Click on the issue to jump to where it is in your project.

Errors		
Type	Location	Issue
	NotesWindow.AddNoteButton.Action, line 3	Type "EditNoteWindow.EditNoteWindow" has no member named "CurrentNotes" editWindow.CurrentNotes = Nil

Warnings are only displayed when you use the “Analyze Project” or “Analyze Current Item” commands. You can choose to display errors or warnings by type or by location using the selector at the top of the Error Panel. You can choose which warnings appear by using the Analysis Warnings window (Project → Analysis Warnings).

## Messages

The Messages Panel is used when running your project using the debugger.

When you run your project, messages for Application Launched and Application Ended are automatically created. Additional system messages may also be displayed here.

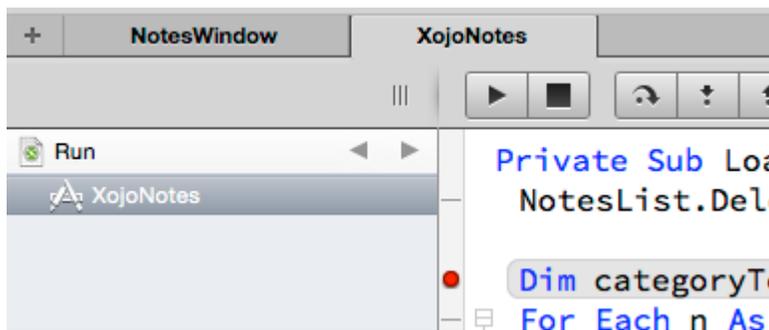
In addition, the output from the System.DebugLog method appears in the Messages Panel allowing your applications to generate their own logging messages to aid your debugging and testing.

Messages		Clear
4:19:29 PM	My Application Launched	
4:19:34 PM	Note added.	
4:19:40 PM	My Application Ended	

You can search for messages using the Search field. The Clear button clears all the messages in the list.

# Running Your Apps

While working on your project, you are going to want to test it to make sure it works like you expect. You can do this by clicking the Run button on the toolbar, selecting Project → Run from the menu or pressing ⌘-R (Ctrl-R on Windows and Linux). When you run your project like this, you are running it in debug mode. In this mode, Xojo is communicating in the background with your app. In the Workspace, a separate tab (containing the debugger) appears with the name of your app.



When this tab is selected, the Debugger displays. In this tab, the Navigator displays the Run section and the name of the app being debugged.

To learn more about Debugging, also read the [Debugging and Profiling](#) chapter.

## [Table of Contents](#)

- [Debugger](#)
- [Using the Debugger](#)
- [Setting Breakpoints](#)

# Debugger

The debugger can be used to watch how your app runs. It is divided into three main areas:

- **Stack**  
Displays the order that the code in your application was called.
- **Variables**  
Displays variables and their current values.
- **Source Code**  
Displays the source code that is currently running with the current line highlighted when the debugger is paused.

To see the Stack, Variables and currently running source code, you can set a breakpoint in your source code or you can click the pause button on the toolbar.

## Toolbar

The debugger toolbar has these commands:

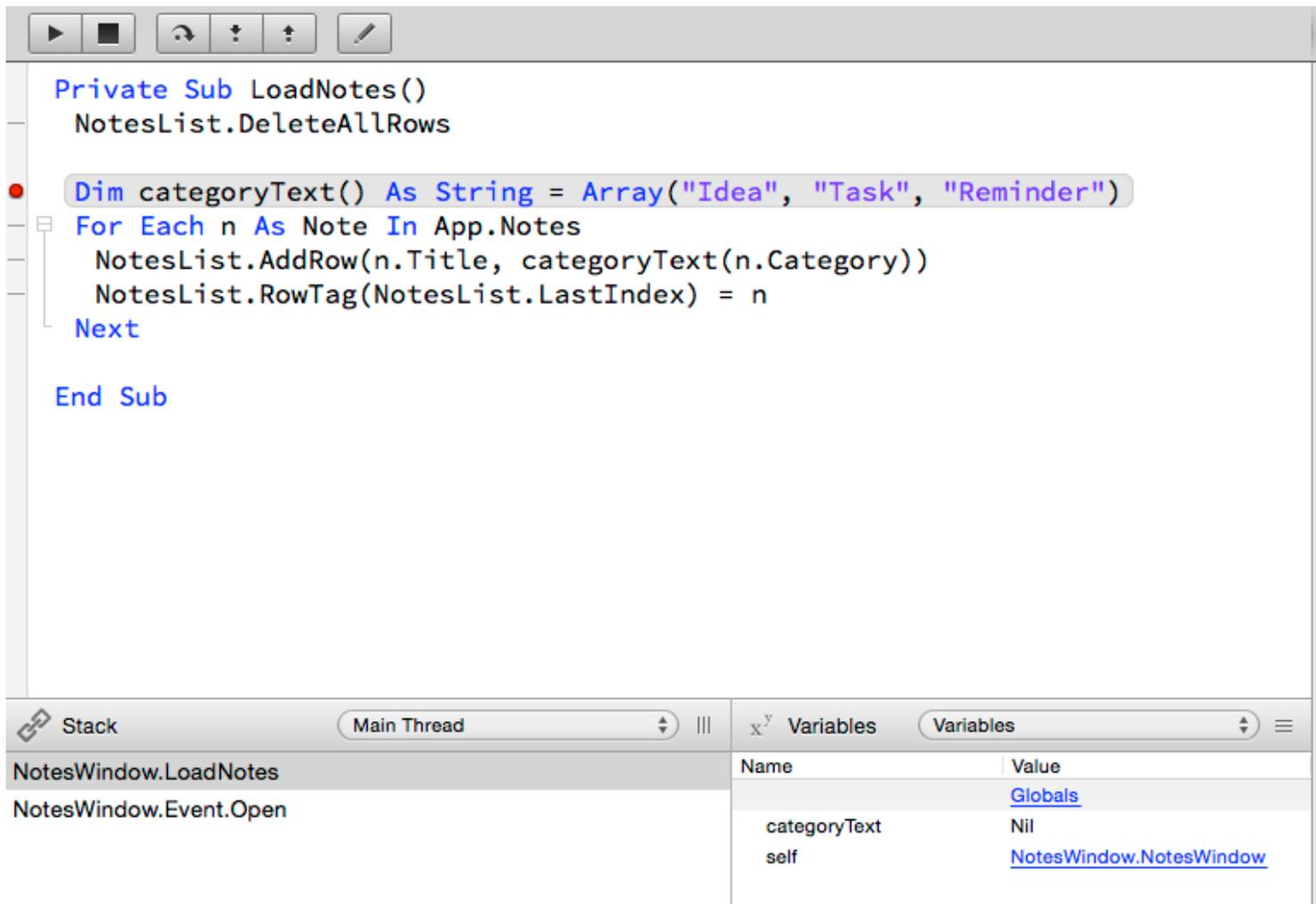
- **Pause**  
Pauses the app at the currently running line of code. The debugger will display the line of code highlighted in gray.
- **Stop**  
Stops the running app, closing the debugger.
- **Step**  
When Paused, steps through the code one line at a time.
- **Step In**  
When Paused, if the current line of code is a method then this action steps you into the first line of code in the method.
- **Step Out**  
When Paused, if you are within a method, this action runs the rest of the code in the method and pauses debugging at the line of code after the method call.
- **View Source**  
Takes you to the project item and displays the source code so that you can edit it. Changes you make to the code do not take affect until you run the project again.

## Using the Debugger

You have three ways to Pause the debugger so you can review running code:

- Use the Pause button, which does not allow you to control where the code will pause.
- Set a breakpoint in your code.
- Use the Break command in your code.

Once you are in the debugger, you can view the source code but not make any changes to it. To edit code, use the View Source button to find the code in the project to edit. Any changes you make do not take affect until you run the project again.



```

Private Sub LoadNotes()
    NotesList.DeleteAllRows
    Dim categoryText() As String = Array("Idea", "Task", "Reminder")
    For Each n As Note In App.Notes
        NotesList.AddRow(n.Title, categoryText(n.Category))
        NotesList.RowTag(NotesList.LastIndex) = n
    Next
End Sub

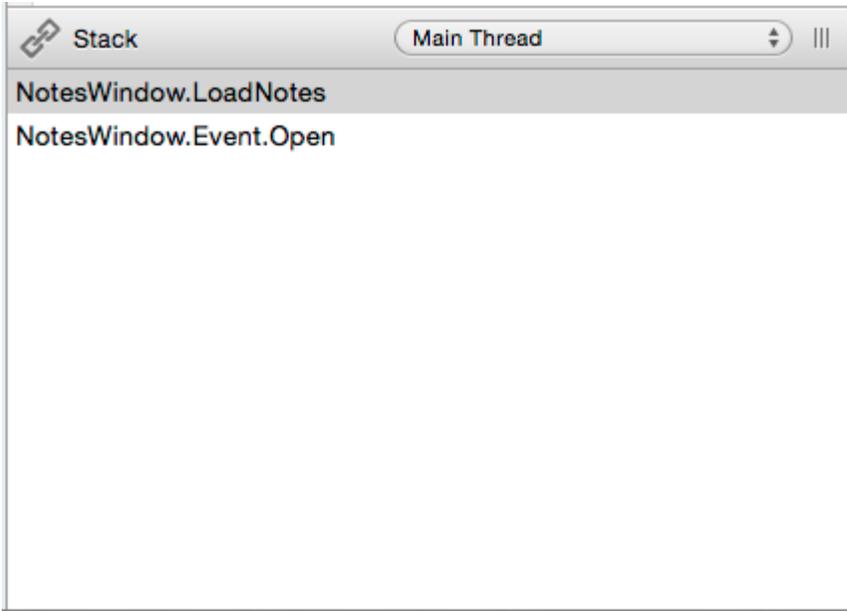
```

Stack		Variables	
Main Thread		Variables	
NotesWindow.LoadNotes		Name	Value
NotesWindow.Event.Open			<a href="#">Globals</a>
		categoryText	Nil
		self	<a href="#">NotesWindow.NotesWindow</a>

To resume running your application after it has paused, click the Resume button on the toolbar, which appears in place of the Run button. You can also completely stop the running application by clicking the Stop button on the toolbar. This does not take you to the debugger since the app has been terminated, but it can be useful if you need to quickly stop the app because of an infinite loop or some other programming error.

## Stack

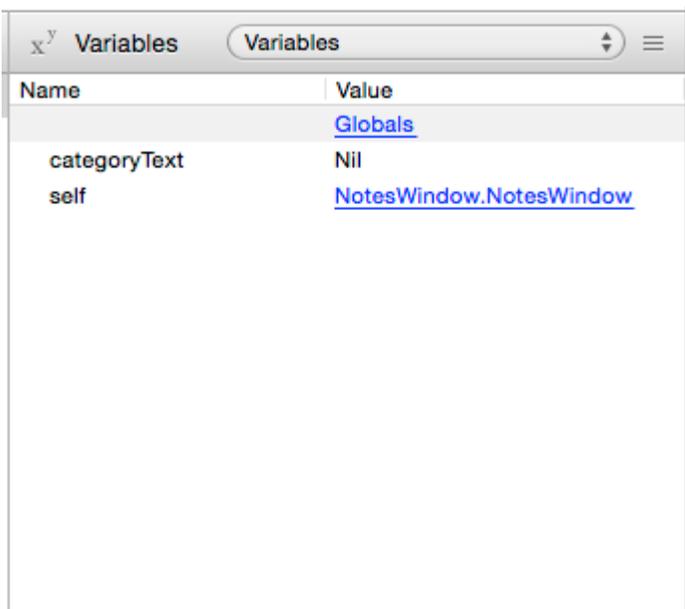
The Stack area (lower left corner) displays the name of the objects in the order they were called with the current object at the top of the list. You can click on other items in the list to view the code that previously ran.



The PopupMenu in the Stack area usually just displays “Main Thread”. If you are specifically using Threading, you can use this popup to look at code that is running in other threads.

## Variables

The Variables area (lower right corner) displays the variables that are local to the currently running method. The left column displays variable names and the right displays their values. You can click on values on the right side to display them in more detail. Some values (such as numbers and strings) can have their values edited so that you can affect the outcome of subsequent code.



As you click to view details of variables, the PopupMenu in the Variables section changes to display what is being viewed. You can use this popup to go back to the prior variables.

In addition, the “Globals” item in the Variables section will let you review global values in the app, such as modules and the contents of the App object.

## Setting Breakpoints

In the Code Editor, the gutter in the left margin shows horizontal dashes for lines of code that can have their breakpoint set. To set the breakpoint, just click on the dash. This causes a red dot to appear indicating that a breakpoint is set. To turn off the breakpoint, click on the red circle. You can also set breakpoints using Project → Breakpoint → Turn on (⌘+\ on OS X, Ctrl+\ on Windows and Linux).

```
LoadNotes
NotesList.DeleteAllRows
• Dim categoryText() As String = Array("Idea", "Task", "Reminder")
  For Each n As Note In App.Notes
    NotesList.AddRow(n.Title, categoryText(n.Category))
    NotesList.RowTag(NotesList.LastIndex) = n
  Next
```

To disable all the breakpoints in your app, use Project → Breakpoint → Clear All. To view all the breakpoints in your project, choose Project → Breakpoint → Show All, which displays the breakpoints in the [Find panel](#).

You can also use the [Break](#) command to cause your application to pause. When your code reaches a Break command, the debugger pauses at that line. In a standalone app, the Break command is ignored. You can use the Break command to conditionally pause your app and activate the debugger based on specific situations.

For example, if your app is in a loop, you may only want to break into the debugger when a certain value is reached:

```
For i As Integer = 1 To 100
  If i = 50 Then Break
Next
```

# Project Types

## Table of Contents

- [Project Formats](#)
  - [Binary](#)
  - [XML](#)
  - [Text](#)
- [External Items](#)
- [Encrypting Project Items](#)
- [Importing Project Items](#)
- [Exporting Project Items](#)
- [Collect Project Items](#)
- [Templates](#)
  - [Changing Default Projects](#)

A Xojo project is a document that contains all the items that make up your application. You can have several projects open at once and you can have several windows open per project. When you choose to create a new project, you have these types from which to choose: Desktop, Web, Console or iOS.

Desktop	Desktop projects allow you to create applications with a graphical user interface that run on Windows, OS X and Linux desktop operating systems.
Web	Web projects allow you to create applications that run on the web. Users interact with these applications using a web browser.
Console	Console projects are used to create text-based applications that run from the command line, terminal or as a background application (service or daemon). Windows, OS X and Linux are supported.
iOS	iOS projects are used to create apps that run on iOS devices such as iPhones and iPads

## Project Formats

You can save your projects in several formats, including the binary single-file format (`xojo_binary_project`), an XML format (`xojo_xml_project`) and a text format (`xojo_project`) which uses separate files for project items.

 Xojo can open and save existing projects in Real Studio project file formats.

In [Preferences](#) you can change the format that is used by default.

### Binary (`xojo_binary_project`)

This is a binary file format. Your project is stored in a single file (except for external items such as pictures) that is easy to distribute. This is the only file format that can be used without a license.

## XML (xojo\_xml\_project)

The XML file format is simply an XML representation of the binary file format. It is also a single file, but it is entirely XML. Being XML, this file format can be larger than the same file saved as Binary, but it is text so you can view it in an external text editor.

## Text (xojo\_project)

The Text file format saves your project as separate text files, one for each project item. Because of this, make sure that you save each Text project in its own folder and do not try to save multiple text projects into the same folder.

The Text file format is ideal for use with version control systems such as Subversion or Git because you can see exactly what files have changed and track the history of changes to files.

This format uses the following extensions for your project items:

xojo_code	Contains source code project items such as modules, classes, web pages, iOS views and anything else not listed below.
xojo_window	Contains windows and container controls from desktop projects.
xojo_menu	Contains menus from desktop projects.
xojo_toolbar	Contains toolbars from desktop projects.
xojo_report	Contains toolbars from desktop projects.
xojo_resources	Contains binary information such as icons, encrypted items and other binary data.
xojo_uistate	Contains the UI layout settings such as window positions and sizes.

When used with a version control system, you should add all the files to your repository except for the xojo\_uistate file (which you'll typically want to mark as ignored).

When you delete (or rename) files or folders from a text project format, they are left on disk so that you can manually handle the change using your version control software.

## External Items

Certain project items are always stored as external items and are not included in the project file. These are: pictures, movies, sounds, text files and other files added to the project.

It is also possible to include windows, classes, or modules in a project that are actually stored as external files. This feature allows more than one project to share the project item. When you modify an external project item and then save your project, your changes are written out to the external file on disk. When you open any other project that refers to the same shared file, that project will reflect your changes.

To convert a project item to an external item, select it in the Navigator and right-click (Control-click on OS X) to

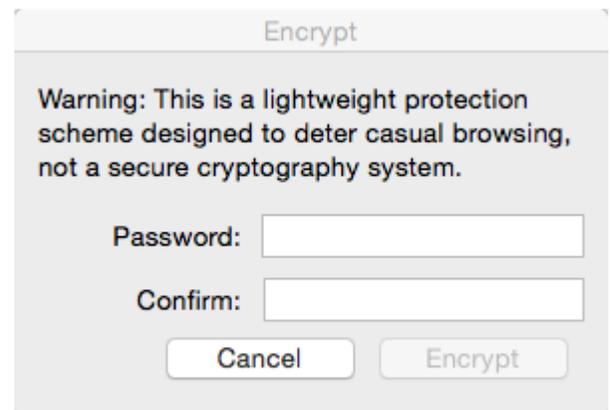
display the contextual menu for the item and choose Make External. This displays a Save File dialog box. Navigate to the directory in which you want to save the item, name it, and click Save. When you have successfully saved the item, it is shown in the Navigator with a shortcut badge and its name is in italics.

To add an item to a project as an external item, hold down the Alt key (Option+Command key on OS X). The File → Import menu command changes to Import as External. Choose that command and select the item to be imported. You'll know you've done it correctly because the icon in the Navigator appears in italics with a shortcut badge.

If an external project item is set to read only (Windows or Linux) or locked (OS X), you can't modify that item within the project, though you can still view it. This provides a convenient way to protect external items (which may be shared by many projects) from accidental modification. Note that if an external file is changed on disk by something other than Xojo — such as a version control system — you'll need to close and re-open the project to reload that item.

## Encrypting Project Items

If you want to distribute a copy of project items for others to use but you do not want them to be able to view or edit your code, use the Encrypt command to protect the object prior to exporting it. An encrypted item displays in the Navigator with an icon containing a small key in its lower-right corner.



You can encrypt (protect) or decrypt (unprotect) a project item while it is in your project using the contextual menu "Encrypt ProjectItemName...". An encrypted item cannot be opened and no one can access any code (or layout) associated with it unless they know the password to decrypt it.

When encrypting an item, you supply a password that can be used to decrypt it later. Do not forget the password as there is no other way to decrypt the item.

## Importing Project Items

To import a file you wish to use in your project, simply drag it from the desktop and drop it in the Navigator. Or, if the file is not conveniently located on the desktop, choose File → Import. An Open File dialog box appears, allowing you to navigate to and import the file.

For code, open the method that you wish to import code into and drag the text clipping into the body of the method. You cannot use the File → Import command to import text files into the body of a method.

To delete a file that has been imported to the project, highlight it in the Navigator and press the Delete key on the keyboard or choose Edit → Delete. You can also delete a file in the Navigator by Control-clicking on the item and choosing Delete from the contextual menu.

## Exporting Project Items

The code for methods, events, constants, properties, and so forth can be dragged to the desktop as text clippings or into a text or word processor (OS X only). You can also of course copy code to the Clipboard and paste it into a text editor or word processor.

You can export your source code to an XML file or a binary file format using the Export... menu command, which can be useful for sharing code.

## Collect Project Items

Some of your project items may be external to the actual project files. These can be things such as graphics or even project items that are shared across multiple projects. For distribution, it can be helpful to collect all project items from their various locations next to a saved project.

External items remain as external items, but they are now grouped (pictures, data, scripts, etc.).

This process does not rename items on disk, so duplicate filenames will be reported as errors.

Collecting a project does not save the project. Typically you will want to do a “Save As” after collecting the project.

Choose File → Collect Project Items to collect the current project.

## Templates

If you have several items you commonly use in every project, you can save them in a project file and make the project file a template for new projects. The template can include custom windows, classes, modules and other project items.

When you create a new project based on the template, Xojo creates a new untitled project that is an exact copy of the template. The template project itself remains unchanged. This lets you create a new project using existing project items without worrying about modifying the original items.

The most efficient way to use a template is to place it in a special directory in the same directory as Xojo called “Project Templates”. If you do so, your list of templates will be listed in the Project Chooser whenever you choose File → New Project.

Here are descriptions of the project templates that are included by default. Although these templates are built-in, but they can be overridden by providing templates that use these names and placing them in the Project Templates folder.

- **CGIApplication:** This is a specialized version of the Console Application template that is intended as a web application that interfaces with the Apache server. Its App class is derived from the CGIApplication class (in the project) that in turn is based on the ConsoleApplication class. CGI (Common Gateway Interface) is how a web application works with Apache. More detailed notes on the CGIApplication are found in the “Notes” section of the App class in the Template project. The HTTP module contains a custom class that manages HTTP requests.

It contains properties and methods that you will use to build the interface to Apache.

- **EmptyService:** This is a service application template. Its App class is based on the ServiceApplication class. It also runs in the background with no user interface. Please see the Notes section for the ServiceApplication class in the Language Reference for more information.
- **Event Driven Console:** This is also a template for a Console Application. Its App class is also based on the ConsoleApplication class and it includes a custom class, MyApplication, that contains shell methods and properties for a Console Application. See the Notes section for the Console Application class in the Language Reference for information on how a console application works.

## Changing Default Projects

When you create a new desktop, web or console project, a simple project is opened with just the basics to get you started.

You can use your own project files as a substitute for the simple projects used for desktop, web and console projects.

To do so, create projects (in Binary format) and add the project items you want to include by default. Save them with the following names in the Project Templates folder:

- Default Desktop Project
- Default Web Project
- Default Console Project
- Default iOS Project

For example, if you create Default Desktop Project that has two windows and some standard modules and classes that you typically use in all your projects, they will be immediately available when you select Desktop from the Project Chooser.

# Preferences

There are several preferences that you can use to alter default behavior. The preferences are grouped into these areas: General, Printing, Coding, Layout, Building and Debugging.

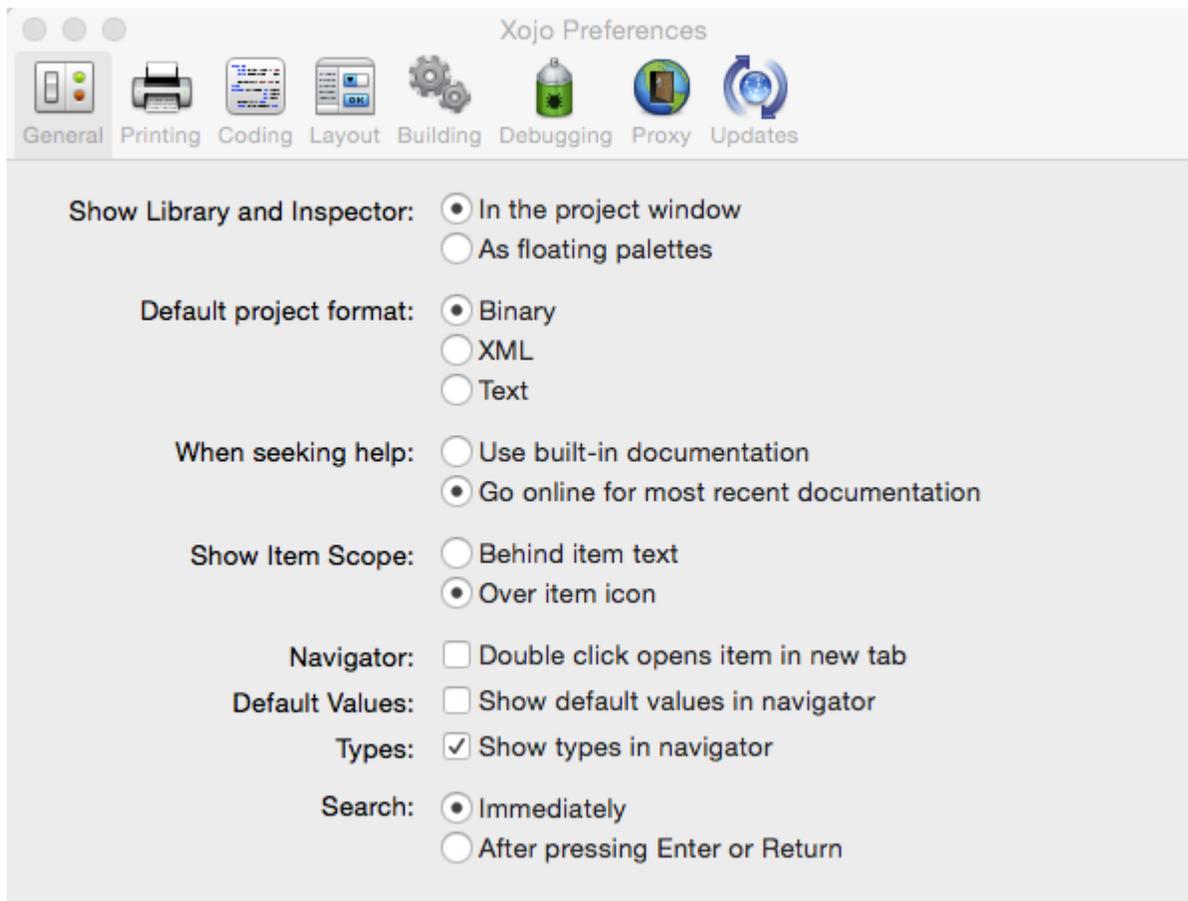
## Table of Contents

- [General](#)
- [Printing](#)
- [Coding](#)
- [Layout](#)
- [Building](#)
- [Debugging](#)
- [Proxy](#)
- [Updates](#)

 On Windows and Linux, the Preferences window is called Options.

## General

The General area contains the most common preferences.



## Show Library and Inspector

Specifies how you want to see the Library and Inspector. By default they appear on the right side of the Workspace, both sharing the same area. You can instead choose to display them as floating palettes so that they are both visible on the screen at the same time and do not take up space in the Workspace.

## Default project format

When you save your projects, Xojo uses the standard single-file binary format by default (`xojo_binary_project`).

Use this setting to change the default project format. Refer to the [Project Types](#) section to learn more about the project formats.

## When seeking help

Xojo looks up commands using the Language Reference. By default, it uses the online Language Reference so that you get the most up-to-date information, but this does require an Internet connection.

If you would rather use the local copy of the Language Reference, change this setting.

## Show Item Scope

This setting determines how the scope for properties and methods is displayed in the Navigator. Choosing “Behind item text” has the scope color displayed as the background color behind the item in the Navigator. Choosing “Over item icon” displays an overlay on the item in the Navigator indicating its scope.

## When closing last window

 This setting is only available on Windows.

When “Keep Xojo running” is selected, a new icon appears in the Taskbar Notifications area (also known as the System Tray) and Xojo remains running when you close the last Workspace window (instead of quitting). You can right-click on the Notification icon to display a menu with “New Project”, “About Xojo” and “Exit”. “New Project” opens the Project Chooser window, “About Xojo” displays the About window and “Exit” quits Xojo completely.

## Navigator

The “Double-click opens item in a new tab” setting controls the behavior of double-clicking project items in the Navigator. The default behavior is to “drill into” the project item using the Jump Bar.

- Check the box to instead open the project item in its own tab when it is double-clicked.
- Check “Show default values in navigator” to display default values of properties in the Navigator.
- When “Show types in navigator” is checked, the types of properties are displayed in the Navigator.

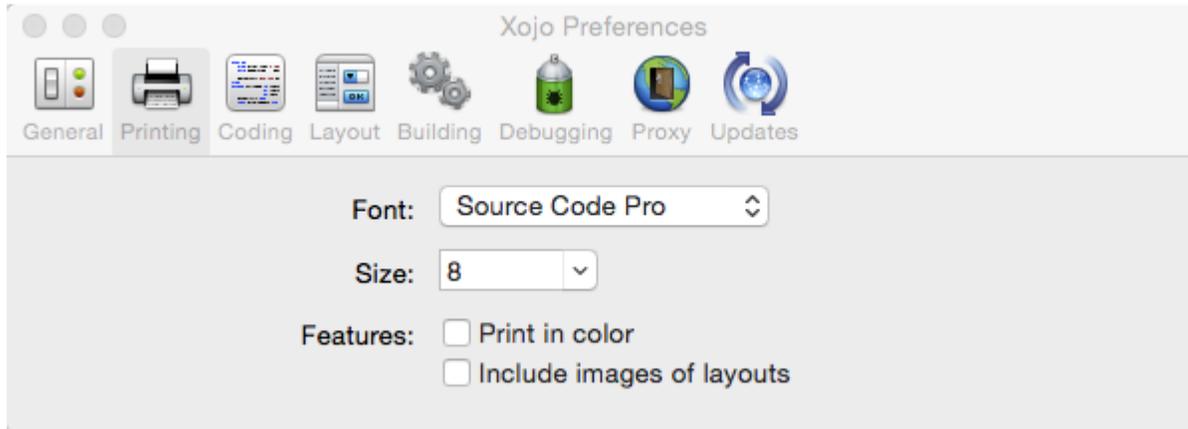
## Search

Here you can choose to have search results appear immediately when you pause typing or only when you press

return in the search field.

## Printing

The Printing preferences control how your code looks when it is printed.



### Font

Choose the font to use for the printed text.

### Size

Choose the font size to use for the printed text.

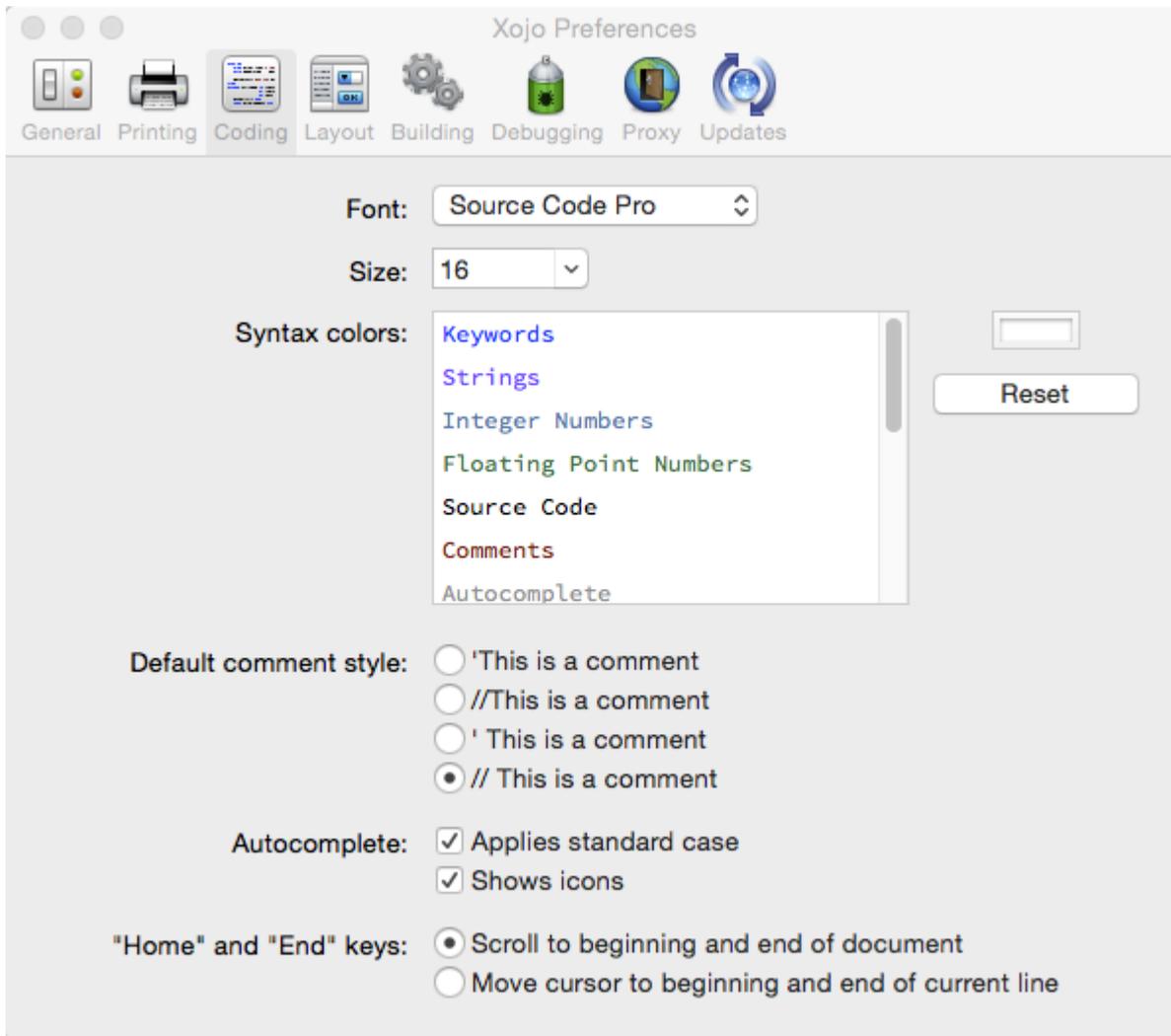
### Features

Check "Print in color" if you want the source code printed using the syntax colors specified in the Coding preferences.

Check "Include images of layouts" if you want printouts to include images of your user interface layouts (windows and web pages).

## Coding

The Coding preferences control how the code is displayed in the Code Editor.



## Font

Select the font to use in the Code Editor from the list of available fonts installed on your system. For best results, use a monospaced font.

## Size

Specifies the font size to use in the Code Editor.

## Syntax colors

You can customize the colors of various parts of your source code. Select a particular syntax and then click the color box to choose a color for it.

## Default comment style

This setting determines the comment style that is used by the Comment button and function on the contextual menu.

## Autocomplete

These two options allow you to control how Autocomplete works. Apply standard case will use the correct case when auto-completing (MsgBox instead of msgbox, for example).

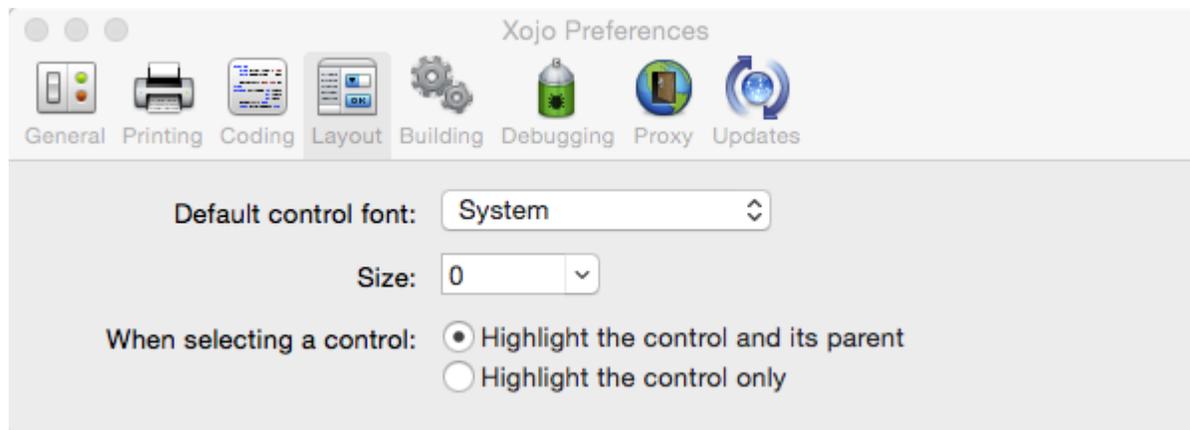
Show icons will show icons next to the values in the autocomplete list.

### “Home” and “End” keys

Determine how you want the Home and End keys to work in the code editor.

## Layout

The Layout preferences adjust how the Layout Editors work.



### Default control font

By default, Xojo uses **System** as the Font for all controls. This allows the platform to substitute its own default font at run-time. If you would rather use a specific font, you can choose it here.

### Size

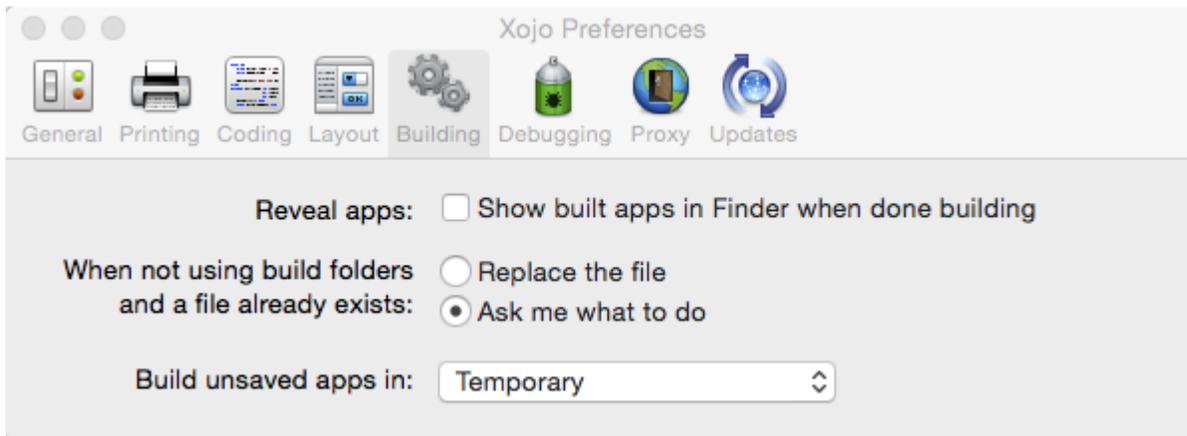
By default, Xojo uses 0 as the Font Size for all controls. This allows the platform to substitute its own default font size at run-time. If you would rather use a specific font size, you can set it here.

### When selecting a control

When you have nested controls (one control on top of another control), the Layout Editor highlights the parent control with a red outline. If you would rather not see this, you can turn it off here.

## Building

The Building preferences control what Xojo does when building or running your applications.



## Reveal apps

After Building your application, Xojo can display the location of the built app using the Finder or Explorer.

## When not using build folders and a file already exists

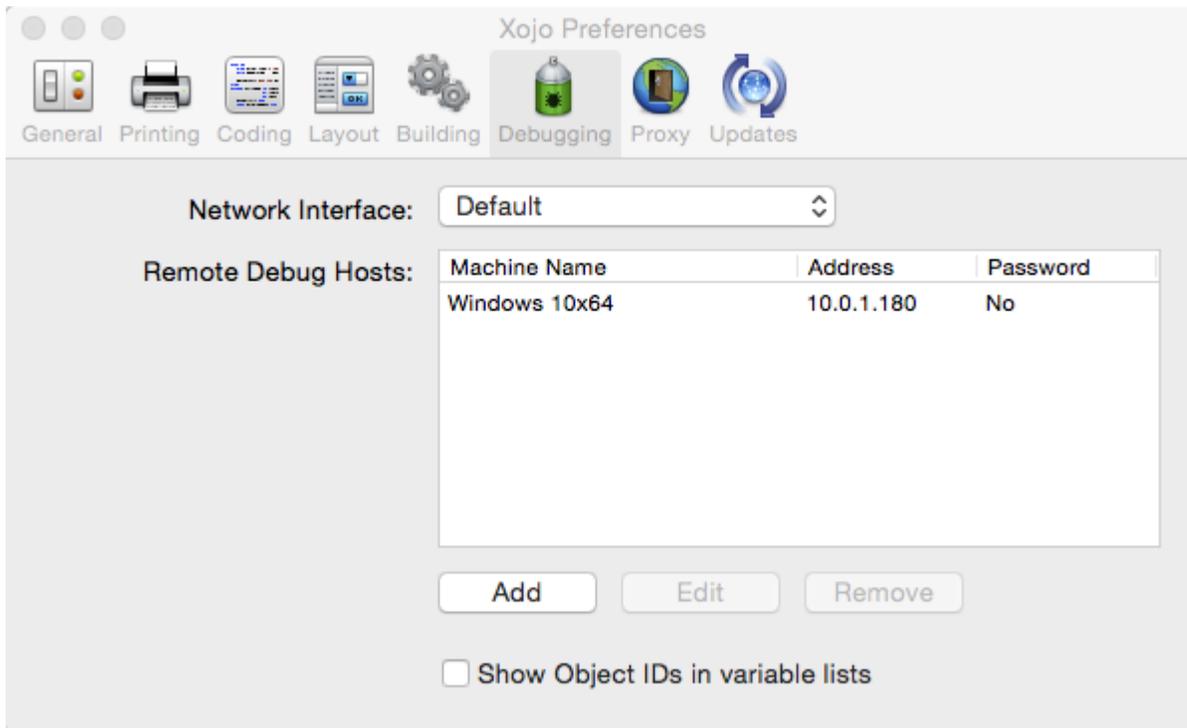
When Building your application (with Use Build Folders set to OFF in Shared Build Settings), Xojo normally replaces any existing built applications (and related files) automatically. If you would rather be prompted when a file will be overwritten, you can change this here.

## Build unsaved apps in

If you try to build an unsaved project, this location is used to store the built application.

## Debugging

The debugging preferences are used to control remote debugging.



## Network Interface

Specify the networking interface to use to connect to the remote debugger.

## Remote Debug Hosts

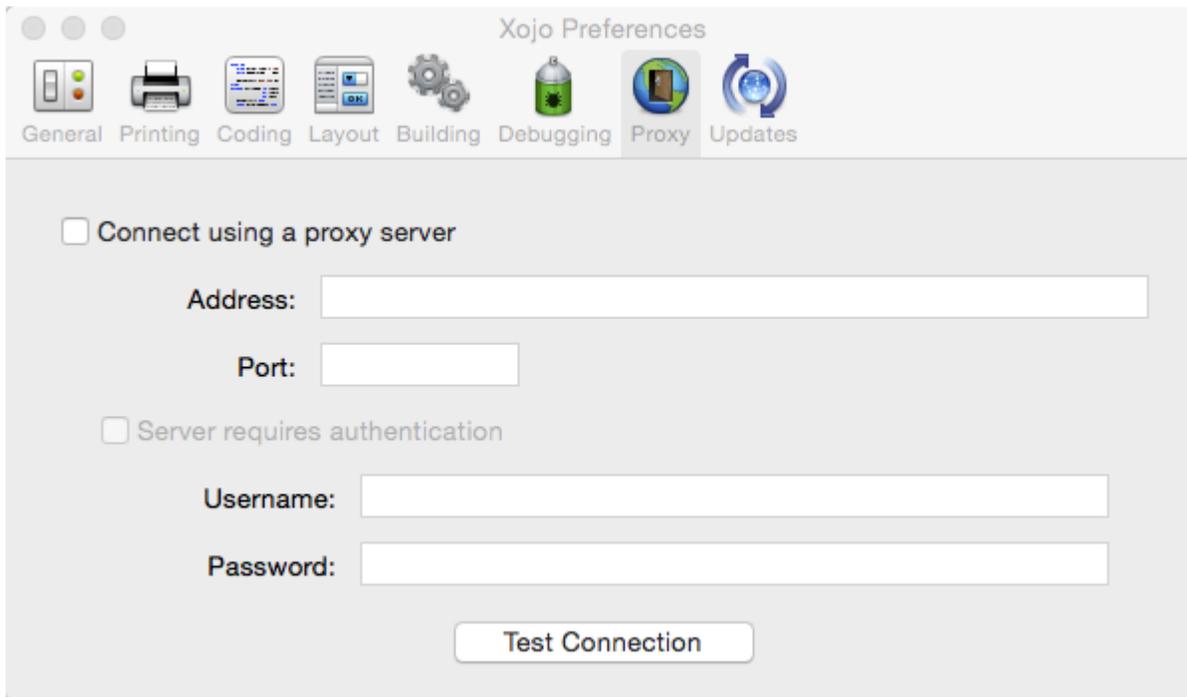
Here you can add, edit or remove remote debugging locations.

## Show Object IDs in variable lists

Displays Object IDs in the debugger Variable area.

## Proxy

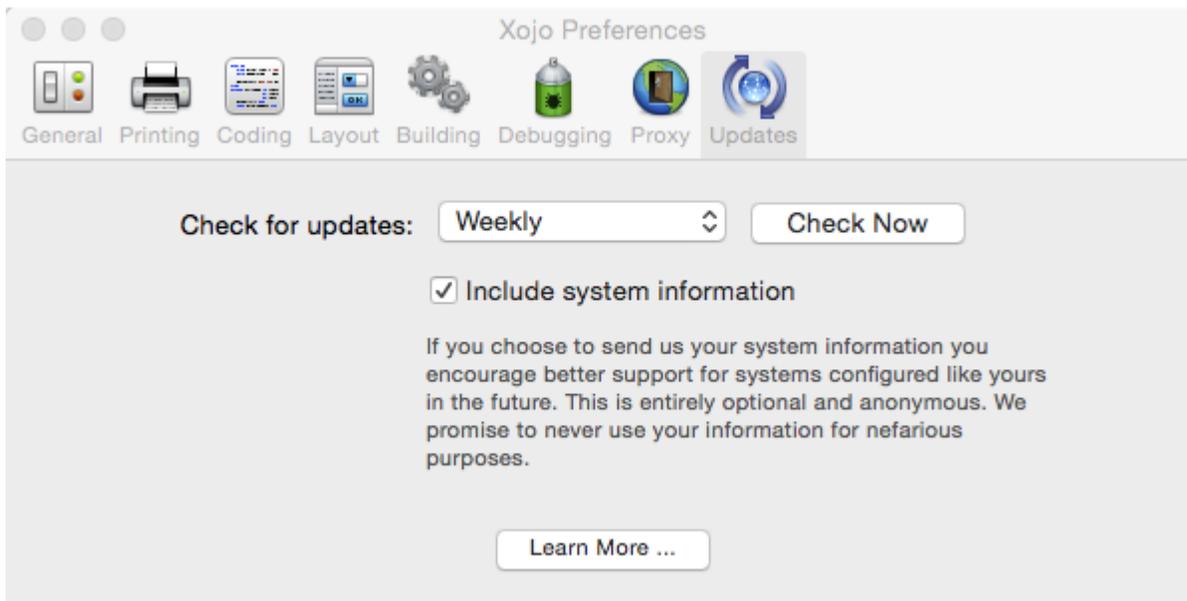
The Proxy preferences are used to specify proxy information so that Xojo can connect to the server to validate your licenses. These settings are used when you “Sign In” to Xojo.



Select the “Connect using a proxy server” checkbox to enable use of the specified proxy server. Fill in the appropriate fields to specify the location of the proxy server and any information needed for authentication.

## Updates

The Updates preferences allow you to choose how often Xojo checks for new versions. You can also specify whether you will allow your system information to be sent to Xojo servers.



# Writing Code

In the [Welcome](#) to this User Guide, you created a simple app with a button that displayed a message. Odds are you want your apps to do a bit more than that, so you're going to have to write some **code**. You do this using the Xojo Programming Language. This language is an easy-to-read, object-oriented programming language that makes it simple to create apps. The Xojo programming programming commands that you write are referred to as code.

To write code you use the [Code Editor](#), which is essentially a text editor specifically designed for writing and editing Xojo code. Your code consists of a series of lines, with each line containing Xojo language commands. Like with any text editor, the text you write in the Code Editor does not "wrap" when you reach the end of the line. Instead the Code Editor scrolls. You create new lines by pressing Return.

The Xojo language consists of the series of commands that you write. With Xojo you generally have one command per line of code.

## Where Does the Code Go?

Xojo does not provide a big blank text editor for you to write your code. Code belongs to a project item or a control on a layout. If you recall from the simple example at the beginning, the code was placed in the Action event handler for a button. Even Handlers are a common place to put your code, but you can also create your own **methods** to contain code.

You can add Event Handlers to any control in your layout by selecting the control and choosing "Event Handler" from the Add button on the Layout Editor toolbar, the Insert button on the main toolbar or the Insert menu.

You can add your own methods (which can be called by other code in your app) by choose "Method" from the Add or Insert options.

 A method is a container for your code. Event Handlers are a special type of method.

## I Already Know How to Program

If you already know how to program, you will likely only need to skim the materials in this section to get an understanding of the Xojo commands that are available. You can also refer to the [Language](#) section of the Reference Guide.

## Naming Rules

In the following sections, you will learn how to create variables, constants, methods, modules, classes and more. All of these items follow the same naming standards, which are as follows:

- Names must start with an ASCII letter (A-Z) or (a-z)
- The remainder of the name can contain any of the following:
  - alphanumeric ASCII characters (A-Z, a-z, 0-9)
  - underscore (\_)
  - Any Unicode character with code point greater than 127

- Names can be any length
- Names are case-insensitive (**age** and **Age** are the same)

If you use an invalid character, you will get a compile error (usually reported as a Syntax Error) when you try to run the project. For names that are specified using the Inspector, you will get a prompt indicating that the name is invalid and the name will revert to its previous name.

# Variables and Constants

## Table of Contents

- [Variables](#)
- [Constants](#)

A computer has two types of memory: temporary and permanent. Temporary memory is used to remember things for as long as the computer is on. As soon as the computer is turned off (or restarted), anything in temporary memory is lost. This is often referred to as RAM (Random Access Memory). Permanent memory is remembered even after a computer is turned off. This type of memory is usually a hard disk, solid-state disk or flash memory. You save things to permanent memory using files, which are discussed in the Framework book.

When you are writing code, you often need to remember things. You use a concept called **Variables** to store information in the temporary memory. Variables are great for holding information such as counter values, field values and anything else you need while your program is running. The term “variable” is used because the value that it contains can be changed by your program, so its value can vary.

## Variables

A variable is the simplest way to store values. All variables are declared in your code using the Dim statement and each variable must be declared before it is used. When you declare a variable you tell it the type of data that it can contain.

A variable declaration consists of three parts: the Dim keyword, the variable name, and the data type. Here is an example variable declaration:

```
Dim age As Integer
```

There are several simple data types: Text (String is also available for some project types), Integer, Double, Currency, Boolean, and Color. Data Types are covered in the next section, Data Types and Storage.

Once you have declared a variable, you can assign it a value:

```
Dim age As Integer  
age = 42
```

Variables can be declared anywhere within a method, as long as the declaration precedes its first usage. If you have several variables of the same type, you can declare them all with one Dim statement:

```
Dim i, j, k As Integer
```

If you want to declare variables of different types, you can also declare them in one Dim statement. For example, the following Dim statement is valid:

```
Dim name, address As Text, shoeSize As Integer
```

But in general it is considered better form to declare variables of different types on separate lines.

When you create a variable with the Dim statement you can also assign it a value. You do so by following its data type with an equals sign and the value. For example:

```
Dim i As Integer = 1
Dim Name As String = "Igor", Address As String = "15 Rue de Vallee"
```

You can also mix variables with and without initial values, as in:

```
Dim a As Integer, b As Integer = 15
```

In this case, the variable *a* is declared as an integer but is not assigned a value. If you examine the value of *a*, it will be zero. The variable *b*, on the other hand, gets the value of 15.

Notice also that the following is valid:

```
Dim a, b, c As Integer = 15
```

This statement declares three integer variables and assigns all of them the value of 15. Although this is valid, you should be careful about assigning values to more than one variable in a single Dim statement because you could easily lose track of the assignments.

## Accessing a Variable

A variable can be accessed only within the method in which it was declared. When the method is finished, the memory that was used to store the variable's value becomes available for other uses. This means that another method in the application cannot access the variable value.

The term **scope** is used to describe where something, such as a variable, can be accessed. Variables and constants declared within methods have what is called a local scope because they are locally available only to the method.

If you declare a variable or constant inside of a code block (such as an If statement, Select statement or any looping statement), its scope is local to the code block and not the entire method.

For example, you can write:

```
If x > 5 Then
  Dim y As Integer
  y = 10
End If
```

' y goes out of scope here; the variable name y can now be reused.

' It is redeclared as a string in the following If statement

```
If x < 5 Then
  Dim y As String
  y = "hello"
End If
```

You can also assign variables a value by using another variable. Doing this does not affect the original variable when you are using simple data types. When using the simple data types, the new variable gets a copy of the data in the original variable. Changing the new variable does not change the value in the original variable.



This behavior is different when working with objects. Learn more about object assignment in the Object-oriented Programming chapter.

This example creates a text variable, assigns it a value and then assigns its value to another text variable:

```
Dim s1 As Text = "hello"  
Dim s2 As Text  
s2 = s1 // s2 now contains "hello"  
MsgBox(s2)
```

```
' Changing s2 does not affect s1  
s2 = "world"  
MsgBox(s2)
```

```
' s1 still contains "hello"  
MsgBox(s1)
```

## Constants

A constant behaves similarly to a variable with these significant differences:

- You use the [Const](#) keyword to declare the constant
- You do not have to specify the type when declaring it
- Its value cannot be changed once it has been set

Here are some example constant declarations:

```
Const pi As Double = 3.14159  
Const value = 256 * 10
```

```
Const kDefault As Text = "DefaultName"
```

Constants can be assigned only a literal value, another constant value or the result of a constant expression.

# Using Data Types

Xojo is a strongly-typed programming language. This means that you must specify the type of every variable you create. Strongly-typed programming languages are safer to use because the compiler can inform you of programming mistakes in variable usage before they make it into your app.

Xojo has several built-in data types (called intrinsic types) which fall into these categories: [Boolean](#), [Text](#), [Numbers](#) and [Color](#). In addition, [Date](#) is a commonly used class that is worth covering here.

## Table of Contents

- [Boolean](#)
  - [Boolean Expressions](#)
- [Text](#)
  - [Combining Text](#)
  - [Text Management](#)
  - [Encodings](#)
  - [String](#)
- [Numbers](#)
  - [Integer](#)
  - [Double](#)
  - [Currency](#)
  - [Mathematical Computations](#)
  - [Converting Between Numbers](#)
- [Color](#)
- [Date](#)
- [Typeless Variables](#)
  - [Auto](#)
  - [Variant](#)
- [Converting Between Data Types](#)

## Boolean

[Boolean](#) is one of the simplest data types. It can only contain one of two values: True or False (the default). This is how you declare a Boolean variable:

```
Dim b As Boolean
```

The special keywords True and False used to change the value of a Boolean variable. If you want to declare a Boolean variable and assign it to True, you can do so in one step:

```
Dim b As Boolean = True
```

Boolean variables can also be assigned the result of a Boolean expression. A Boolean expression is an expression that evaluates to True or False. For example, this code sets a variable to True or False depending on whether an Integer value is greater than 5:

```
Dim i As Integer = 10
Dim b As Boolean
b = (i > 5) ' b gets set to True
```

## Boolean Expressions

Boolean expressions deal only with the values True and False. True and False are values that Xojo can manipulate, just as it does numbers and strings.

There are two types of Boolean expressions:

- **Simple:** Boolean expressions state something about a value in the program, generally using operators like = (equals), <> (not equals), <= (less than or equal), or >= (greater than or equal). Sometimes a property or something else in the program will actually just be a Boolean value. The toggle switches you see in the Inspector are good examples, such as the Visible property of a control. For example, Label1.Visible evaluate to True when the label's Visible switch is set to ON; and it evaluates to False when the switch is set to OFF.
- **Compound:** Boolean expressions that are made out of other Boolean expressions (which can themselves be either simple or compound), using Boolean operators Not, And or Or.

As with other types of expressions, you will often need to use parentheses in Boolean expressions to make it clear in what order operations should be carried out. Also, as with other types of expressions, Xojo works out the final value from the inside out.

### Example Expressions

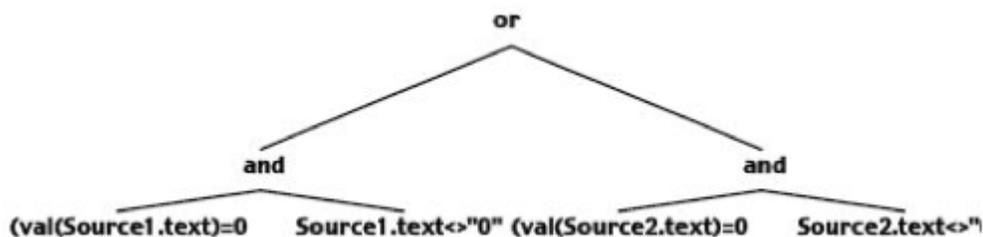
Consider this expression:

```
(Val(Source1.Text) = 0 And Source1.Text <> "0") Or (Val(Source2.Text) = 0 And Source2.Text <> "0")
```

A compound expression such as this is evaluated according to the left-to-right rule. The two branches of the OR expression are evaluated separately prior to ORing them.

If a compound expression is joined by AND, then any sub-expression that evaluates to False ends the process of evaluating any subsequent sub-expressions within that compound expression. This is because one False sub-expression means that the entire compound AND expression must also be False.

An expression like this can be represented as a syntax tree:



The expression is evaluated from the bottom of the tree up. Each level of the tree is evaluated left to right.

In this example, the first step is to evaluate the left branch of the OR expression. It is a compound expression of its own, as its branches use AND.

It starts with

```
Val(Source1.Text) = 0
```

If this expression returns True then it evaluates:

```
Source1.Text <> "0"
```

Lastly, if

```
Val(Source1.Text) = 0
```

is False, there is enough information to evaluate the “AND” so False is returned to the next higher level without evaluating the second expression.

Since the two compound expressions are joined by OR, it needs to evaluate the right compound expression at the base of the tree even if the left compound expression evaluates to False. If the left branch is False, but the right branch is True, then the complete compound expression is True.

The same procedure is followed in evaluating the right branch of the compound OR statement.

## Text

Text is any series of characters. Xojo has two data types that can be used to store a series of characters: Text and String. In most situations you will want to use Text as it is far more flexible, but String is sometimes needed when working with code that is written using the Classic Framework.

### Text

 Text stores its text as Unicode.

The maximum size of Text is limited only by available memory. To declare a variable containing Text:

```
Dim t As Text
```

The default value of text is simply an empty text, often written as "". You can also check if text is empty using the Empty method:

```
Dim t As Text
If t.Empty Then
  ' Do something if text is empty
End If
```

You can also directly assign text to the variable as part of its declaration:

```
Dim t As Text = "Hello, World!"
```

Basically, any type of characters can be stored as Text, such as: "Lucas", "12/30/2015", "42.15".

Yes, even though those last two values look like a date and a number, they are actually Text because they are a

series of characters. If you want to treat them as if they are an actual date or a number then you need to use a specific data type for that.

## Combining Text

You can combine multiple text values together by using the addition (+) operator. This concatenates the two values together. For example:

```
Dim name As Text = "Lucas"  
Dim age As Text = "11"
```

```
Dim combined As Text = name + age ' result is "Lucas11"
```

Remember, that text values combined in this manner are only ever concatenated, regardless of what value they contain. For example, even if it looks like the text contains numbers, adding them will not calculate their sum:

```
Dim value1 As Text = "42"  
Dim value2 As Text = "10"  
Dim value3 As Text = value1 + value2 ' result is "4210", not "52"
```

## Text Management

Text has many methods that can be used to manage its contents. For example, you may want to check if a Text variable contains another text value or you may want to remove whitespace from a text value.

To learn more about the methods that are available with Text variables, refer to the [Text](#) data type in the Reference Guide.

## Encodings

In order to store text outside of your apps (in files or databases, for example), the text needs to be encoded. When text is encoded, it is converted to a series of bytes that can be processed by other applications and systems. If you are loading text from these outside sources into your app, then you'll also need to know the encoding.

The most common encoding for Unicode text is UTF-8, but if you may run across other encodings (such as ASCII, Win-Latin, MacRoman, etc.).

Encodings are covered in detail in the Framework section of the User Guide.

## String

 Only available with Desktop, Web and Console projects.

String stores the text in memory as bytes with an optional encoding (usually UTF-8). A String without an encoding may or may not contain text (it could actually be binary data), which can lead to all kinds of hard-to-find bugs when trying to display the String as text.

String remains available for compatibility with projects that use the Classic Framework (Desktop, Web and Console). If you have a String variable, you can convert it to Text by calling the ToText method:

```
Dim s As String = "Hello"  
Dim t As Text = s.ToText
```

Converting a String to a Text works only when the String has a known encoding. If the encoding of the String is not known (because it was loaded from an outside source such as a file, a database or memory), then you will have to first set the encoding using the [DefineEncoding](#) method.

Conversely, you can always convert a Text variable to a String because the text is Unicode and can always be correctly converted.

## Numbers

Xojo has separate data types depending on the type of number you need. The Integer type is used for whole numbers (positive or negative and 0). If you need a number that contains a decimal, you can use either Double or Currency.

Because these data types contain actual numbers, you can perform mathematical computations on them.

### Integer

An [Integer](#) contains whole numbers and is declared like so (default value is 0):

```
Dim value As Integer
```

You can assign it a value when you declare it:

```
Dim value As Integer = 42
```

If you try to assign a value with a decimal to an Integer, the decimal value is truncated (not rounded) before it is assigned. For example:

```
Dim value As Integer  
value = 42.65 ' value will contain 42
```

### Double

A [Double](#) is a number that can contain a floating-point decimal value. In other languages, Double may be referred to as a double precision real number. Because Doubles are numbers, you can perform mathematical calculations on them.

You should avoid using Double to store monetary values. As an IEEE floating-point number, there  are some decimal values that cannot be represented as a Double and this [could cause rounding](#)

 issues. In these situations, use Currency instead.

The default value of a Double is 0.0:

```
Dim value As Double
```

You can also assign a value when you declare it:

```
Dim value As Double = 3.14</code>
```

## Currency

A Currency is a fixed-point number format that holds approximately 15 digits to the left of the decimal point and 4 digits to the right. It is always accurate to four decimal places. Currency is useful for calculations involving money and for calculations where accuracy is very important. It is compatible with the Currency data type offered in some versions of Visual Basic.

The default value of a Currency variable is 0.0.

Here is are some declarations:

```
Dim value As Currency
Dim amount As Currency = 42.56</code>
```

## Mathematical Computations

Performing mathematical calculations is a very common task in programming, and all of the common mathematical operations are supported for the number data types.

This is a list of mathematical operators:

Operation	Operator	Example
Addition	+	$2 + 3 = 5$
Subtraction	-	$3 - 2 = 1$
Multiplication	*	$3 * 2 = 6$
Floating Point Division	/	$6 / 4 = 1.5$
Integer Division	\	$6 \setminus 4 = 1$
Modulo	Mod	$6 \text{ Mod } 3 = 0$ $6 \text{ Mod } 4 = 2$
Exponentiation	^	$2 ^ 3 = 8$

There are also many built-in mathematical functions that are covered in detail in the [Reference Guide](#).

Expressions support standard mathematical precedence. This means that equations surrounded by parentheses are handled first. The expression is evaluated beginning with the set of parentheses that is embedded inside the most

outer sets of parentheses. Next, exponentiation is performed, followed by any multiplication or division from left to right. Finally any addition or subtraction is performed. As an example, these three expressions return different results because of the placement of parentheses.

Expression	Result
<code>2 + 3 * (5 + 3)</code>	26
<code>(2 + 3) * (5 + 3)</code>	40
<code>2 + (3 * 5) + 3</code>	20

For more information, view the [Operator Precedence](#) table.

## Converting Between Numbers

You can convert between numbers without any special commands, but you have to keep in mind that the numbers may be rounded or truncated. For example, converting a Double to an Integer does not round the value:

```
Dim d As Double = 4.6
Dim i As Integer = d // i = 4
```

Use the [Round](#) method to round a value.

You can also directly assign an Integer to a Double:

```
Dim i As Integer = 42
Dim d As Double = i
```

## Color

A Color is a data type that stores the value of a color. A Color “value” actually consists of three numeric values that can be set using any of the three popular color models, Red-Green-Blue, Hue-Saturation-Value, or Cyan-Magenta-Yellow. Each RGB value is stored as a byte. That means that each number can take on 256 possible values.

A common way to create a specific color is to use the RGB method. This creates a color variable containing a blue-gray color:

```
Dim c As Color = Color.RGB(124, 124, 163)
```

## Date

Date is a class and not an intrinsic data type. But it is commonly used and worth mentioning here. Because it is a class, you have to create an instance of it before you can use it. The easiest way to get a [Date](#) instance is to use the Now method which gives you the current date and time:

```
Dim d As Xojo.Core.Date = Xojo.Core.Date.Now
```

You can also create a specific date by providing the values like this:

```
Dim d As New Xojo.Core.Date(2015, 11, 20, Xojo.Core.TimeZone.Current) ' Nov 20, 2015
```

Once you have a Date, you cannot modify it. You can however subtract dates to get a DateInterval of the time between the two dates or you can add a DateInterval to a date to get a new Date.

This code gets the date two months before today:

```
Using Xojo.Core
Dim twoMonths As New DateInterval
twoMonths.Months = 2 ' 2 month interval

' Get date two months before today
Dim past As Date = Date.Now - twoMonths
```

This calculate the interval until January 1, 2030:

```
Using Xojo.Core
Dim d2 As New Date(2030, 1, 1, TimeZone.Current)
Dim interval As DateInterval
interval = d2 - Date.Now
```

## Typeless Variables

There may be times where you need a variable that can contain the value of any other variable, regardless of its type. The Auto data type is used for this purpose.

### Auto

The `Auto` type is a replacement for Variant. It can contain any type of data and essentially automatically takes on the type of the data assigned to it.

For example, if you assign an Integer to an Auto variable, you can not later use it as a Text. You'll have to first assign it to an Integer and then convert it to a Text using `intValue.ToText`.

Here are some examples:

```
' Add an Auto containing an integer
Dim num As Auto
num = 42

Dim sum As Integer
sum = num + 10

' Display an Auto containing an integer
Dim numInt As Integer = num
Dim output As Text
```

```
output = numInt.ToText ' output contains "42"
```

```
' Convert an Auto containing a Double to Text
```

```
Dim value As Auto = 5.5
```

```
Dim t As Text = CType(value, Double).ToText
```

You can use Introspection to check the type of the value contained in an Auto variable. Introspection is covered in more detail in later chapters, but here is a quick example that displays the type of a value in an Auto variable:

```
Dim autoVar As Auto = 42
```

```
Dim info As Xojo.Introspection.TypeInfo
```

```
info = Xojo.Introspection.GetType(autoVar)
```

```
Label1.Text = "Type: " + info.Name ' Displays Int32
```

You can assign a value of a different type to an Auto that already had a value. So if an Auto variable originally contained an Integer, you can reassign it with a Text variable without an error:

```
Dim value As Auto
```

```
value = 42
```

```
value = "Hello"
```

## Variant

 Only available in Desktop, Web and Console projects.

The Variant type is similar to Auto, but it has the ability to implicitly convert its values to other data types. This can result in hard-to-find bugs in your code when you run into unintended behavior from type conversions. Variant remains available for compatibility purposes, but you should use Auto in most situations.

## Converting Between Data Types

There may be times when you need to change a value from one data type to another. This is usually because you want to use the value with something that is designed to work with a different data type. For example, you might want to include a number in a Label's Caption, but the caption is text, not a number. Consequently, if you try to assign a number to the caption, you will get an incompatible type error when you try to run your project.

Instead, you need to convert the number to the appropriate type. Most commonly, you will need to convert from numbers to text and from text to numbers. You can do this using the ToText and FromText methods on the data types.

For example, to convert a number to text:

```
Dim i As Integer = 42
```

```
Dim t As Text = "The number is " + i.ToText
```

To convert text to a number:

```
Dim value As Integer
value = Integer.FromText("42") ' value = 42
```

If your source text may not contain a number, you may want to use the Parse method instead as it is less restrictive about the incoming text:

```
Dim i As Integer
i = Integer.Parse("123ABC")
' i = 123
```

You may also want to convert a Date to text for display purposes. It also has a ToText method. This code display the current date and time:

```
Dim d As Xojo.Core.Date = Xojo.Core.Date.Now
Label1.Text = d.ToText
```

To get just the date portion, you provide optional format styles for the date and time parts of the date:

```
Using Xojo.Core
Dim d As Date = Date.Now
Dim t As Text = d.ToText(Locale.Current, Date.FormatStyles.Short,
Date.FormatStyles.None)
```

# Collections of Data

Programs often have a lot of data to manage. Two common techniques for managing data include arrays and dictionaries. Although they both store large collections of information, they do so in different ways.

## Array

An array is a special type of variable that contains several values of the same data type. Each variable in the array is called an element of the array. To refer to a particular element, you use an index number.

The Dim statement also lets you create and type arrays. When you declare an array, you specify the number of elements it has. Later on, you can change the number of elements if necessary.

### Creating Arrays

You declare an array by specifying the index of the last element of the array. The index that you specify in the Dim statement is actually one less than the number of elements in the array because arrays always have an element at position zero (0). Such an array is sometimes referred to as a zero-based array. Since arrays are zero-based, this statement creates a Text array with eleven (11) elements:

```
Dim names(10) As Text
```

This statement creates an array with one element, element zero:

```
Dim names(0) As Text
```

In other words, you declare a variable as an array simply by adding the index of the last element to the Dim statement. The index of the last element must be either a number or a constant. You cannot use variables with this syntax.

```
Const kSize As Integer = 10  
Dim names(kSize) As Text
```

If you don't know the size of the array you need at the time you declare it, you can declare it as a null array, i.e., an array with no elements. You do this by using an index of -1 in the Dim statement or leave empty parentheses. This means "an array of no elements." These two examples create an array with no initial elements:

```
Dim firstName(-1) As Text  
Dim lastName() As Text
```

Since the array does not have any elements, you'll have to actually add elements to it. You can do this by resizing it using the Redim statement or by adding elements to it using the Array method (at assignment) or by using the Insert or Append methods.

This uses the Array method to populate an array, setting its size to the number of elements specified:

```
Dim values() As Integer = Array(5, 8, 42, 56, 32)
```

This example uses the Append method to add rows to an empty array. This technique is useful when the data to go in the array comes from another source that may vary in size, such as a file or a database:

```
Dim names() As Text
names.Append("Bob")
names.Append("Phil")
names.Append("Larry")
' The array now has a size of 2, with indexes 0, 1 and 2.
```

## Multidimensional Arrays

You can create multi-dimensional arrays. For example, a spreadsheet layout can be thought of as a two-dimensional array, rows by columns.

Each dimension is referred to by its own index. For example, the elements of an array with two dimensions are referred to by one index for the rows and the other for the columns. The first element in the upper-left corner is element 0,0.

You create a multi-dimensional array by specifying an index for each dimension. For example, the statement creates a two-dimensional array with 3 rows and 11 columns:

```
Dim names(2, 10) As Text
```

You can use constants in Dim statements to set the size of an array:

```
Const kLanguages As Integer = 100
Dim names(10, kLanguages) As Text
```

## Referring to Array Elements

You refer to an element of an array by placing the desired element in parentheses. This example places the text "Frank" in array element (1,1):

```
names(1, 1) = "Frank"
```

## Getting the Index of the Last Element

The Ubound method returns the index of the last element of a one-dimensional array. For example:

```
names.Ubound
```

The number of elements is one greater than this number, since the array has an element zero.

For example, the following example returns 5 in the variable i:

```
Dim i As Integer
Dim names(5) As Text
i = names.Ubound
```

## Initializing Arrays

After you have declared an array, you can assign initial values to the elements with the Array function as well as individual assignment statements, such as shown above. The Array function takes a list of values and assigns the values to the elements of the array, beginning with element zero. In other words, it provides the same functionality as separate assignment statements for each element of the array.

For example, the following statements initialize the array using separate assignment statements for each array element:

```
Dim names(2) As Text
' Separate assignment statements
names(0) = "Fred"
names(1) = "Ginger"
names(2) = "Stanley"
```

The following statements use the Array function to accomplish the same thing. Note that you don't have to declare the exact size of the array in the Dim statement.

```
Dim names() As String
names = Array("Fred", "Ginger", "Stanley")
```

The Array function will add elements to the array as needed.

If you declare the array as a fixed size but don't specify as many values as elements, the Array function will start with element zero and use as many elements as are specified.

## Array Assignment

If you have two arrays of compatible data types, you can assign one array to the other array. Simply use the assignment statement without the parentheses. Here is a simple example:

```
Dim names(2), copyNames(2) As Text
names = Array("Fred", "Ginger", "Stanley")
copyNames = names
```

The last statement assigns the values of all three elements of names to the first three elements of copyNames. If copyNames had fewer elements than names, then additional elements would first be added to copyNames and the assignment of all the elements of names to copyNames would be completed. For example, the following is valid:

```
Dim names(3), copyNames(2) As String
names(0) = "Fred"
names(1) = "Ginger"
```

```
names(2) = "Tommy"
names(3) = "Woody"
copyNames = names
```

After the code runs, the copyNames array has a fourth element for storing the value “Woody”.

It is important to understand that when you copy an array like this, both variables are essentially pointing to the same array contents. Changing an element in copyNames would also change the same element in the original names array.

If you need to actually make a copy of the individual elements to a new array, then you will have to loop through and copy each element one by one:

```
Dim names() As Text = Array("Fred", "Ginger", "Tommy", "Woody")
Dim copyNames() As Text
```

```
For i As Integer = 0 To names.Ubound
  copyNames.Append(names(i))
Next
```

## Resizing Arrays

There are several ways to resize arrays after they have been created.

- **Append:** The Append method adds an element to a one-dimensional array, increasing its size by one. You pass the value you want to add to the array when you call Append. For example, the following statement adds an element to the array names and sets the value of this element to the string, “Dave”:  
`names.Append("Dave")`
- **Insert:** The Insert method creates an additional element in a one-dimensional array and inserts it in the place you specify. It takes two parameters, the index of the element to be inserted and the value of the new element. For example:  
`names.Insert(9, "Hal")` After this statement runs, the value of names(9) would be “Hal”. The old element(9) would be shifted up to element(10) and so forth. The size of the array would be increased by one, as with Append.
- **Remove:** The Remove method deletes the element whose index you specify. This example removes the element with index 9, decreasing the size of the array by one and shifting the array elements after the removed element down by one:  
`names.Remove(9)`
- **Redim:** The Redim statement resizes an existing array. You pass the new values of the array’s indexes but you don’t specify the data type, which is set by the initial Dim statement. This example resizes the array names to 101 elements:  
`Redim names(100)` Redim works on both one- and multi-dimensional arrays. For multi-dimensional arrays, you can only resize the existing dimensions; you cannot add or reduce the number of dimensions themselves.  
 A difference between Dim and Redim is that Dim accepts only integers or constants, while Redim accepts any expression that returns an Integer. This includes, for example, a user-written function that returns an Integer value or a simple variable.

Using this feature, you can dimension your arrays on-the-fly.

If you don't know the size of the array you need at the time you declare it, you can declare it as a null array, i.e., an array with no elements, and use the `Redim` command to resize it later. If your program needs to load a list of names that the user enters, you can wait to size the array until you know how many names the user has entered. You can write a function to figure out what that number is and use it with `Redim` or use the `Append` method to add the required elements to the array one-by-one.

## Converting to and from an Array to Variables

Two functions enable you to take a take an array and break it up into separate variables and take a single string variable and convert it into an array.

- [Split](#): The `Split` function takes a `Text` variable and creates a one-dimensional array by dividing the text up into elements separated by a delimiter. The delimiter is a character or series of character that signals the end of one element and the start of the next element. By default, the delimiter is a space, but you can specify another delimiter.

Here is an example that divides up the contents of a string into array elements. It specifies the comma as the delimiter. After this call, the resulting array, `names`, has three elements:

```
Dim s As Text
// resize using the Split method
Dim names() As Text
s = "Juliet,Taylor,Casting"
names = s.Split(",")
' names(0) = "Juliet"
' names(1) = "Taylor"
' names(2) = "Casting"
```

- [Join](#): `Join` does the opposite of `Split`. It takes a one-dimensional array and returns text containing all the elements separated by the specified delimiter. This example takes an array and returns its contents as a single text variable:

```
Dim names(2) As Text = Array("Bob", "Phil", "Larry")
Dim t As Text
t = Text.Join(names, ",") ' t = "Bob,Phil,Larry"
```

## Dictionary

A [Dictionary](#) is an object that is made up of a list of key-value pairs. That is, each value is paired with an identifying key. The interesting feature of Dictionaries is that both the key and the value are [Auto](#) so they can contain any data type. This means that a dictionary can store a mixture of data types — and that the key doesn't have to simply be an integer. You can look up a value in a Dictionary by specifying either its key.

A Dictionary is a great way to have fast, random lookups of information and can be considerable faster than a large array.

Although technically a class (and classes are not described until later in the User Guide), Dictionary is so commonly used that it is worth covering here.

As a class, you have to create an instance before you can use a Dictionary:

```
Dim d As New Xojo.Core.Dictionary
```

You add values to the Dictionary using the Value method to specify a key name and assign it the value:

```
d.Value("FirstName") = "Bob"
```

You can look up a value using the key:

```
Dim name As Text  
name = d.Value("FirstName")
```

Once you have a Dictionary populated with values, you can iterate through them using a For..Each loop:

```
Dim pets As New Xojo.Core.Dictionary  
pets.Value("Cat") = "Shawmut"  
pets.Value("Dog") = "Seltzer"  
pets.Value("Gerbil") = "Squeakers"
```

```
For Each entry As DictionaryEntry In pets  
  TextArea1.Text = TextArea1.Text + entry.Key + " is " + entry.Value + ", "  
Next
```

## Pair

The Pair class is similar to the Dictionary. It has two properties: Left and Right. Thus, each Pair instance consists of a key-value pair. As it the case with the Dictionary, the values in the pair are variants. You use the ":" operator to assign the Left and Right values when the Pair is declared.

For example:

```
Dim p As Pair = "Telephone Number" : "(406) 737-8946"
```

This assigns "Telephone Number" to the Left property of the pair and the value of the number to the Right property. It also can store a linked list of pairs when it is passed a list of items, such as:

```
Dim p as Pair = 1 : 2 : 3 : 4 : 5
```

The first pair consists of the pair “1” (Left property) and the second pair object (Right property); the next pair consists of the “2” and the third pair, and so forth.

## Comparing Values

There are many times when you need to compare two values to determine whether or not a particular condition exists. When making a comparison, what you are really doing is making a statement that will either be True or False. For example, the statement "My dog is a cat" evaluates to False. However, the statement "My dog weighs more than my cat" may evaluate to True.

Description	Symbol	Example	Result
Equality	=	5 = 5	True
Inequality	<>	5 <> 5	False
Greater Than	>	6 > 5	True
Less Than	<	6 < 5	False
Greater Than or Equal To	>=	6 >= 5	False
Less Than or Equal To	<=	6 <= 5	True

Text and Boolean values can also be used for comparisons. Text comparisons are case-insensitive and alphabetical. This means that "Steve" and "steve" are equal. But "dave" is less than "steve" because "dave" falls alphabetically before "steve". If you need to make case-sensitive or lexicographic comparisons, you use the [Compare](#) method of the Text data type.

```
Dim dog As Text = "Dog"
Dim cat As Text = "Cat"

Dim result As Integer
result = dog.Compare(cat)
' result = 1
```

In addition, there is a floating point equals operator that allows you to determine whether two floating point numbers are close enough to be considered equal. Use it to account for the imprecision of operations such as floating point division. The floating point equals operator is the Equals method of the Double data type.

For example, if you are comparing 10000 to a value and specify x=1, then the acceptable values are 10000.0000000000002, 10000.0 and 9999.999999999998.

```
Dim d As Double = 10000
If d.Equals(compareValue, 1) Then
  // take action here..
End If
```

## Logical Comparisons

You can test more than one comparison at a time using the And, Or, and Not operators. When passed Boolean

values, these operators determine whether the expression is True or False.

## And Operator

Use this operator when you need to know if all comparisons evaluate to True. In the example below, if the variable x contains 10 then the expression evaluates to False because 10 is not both greater than 1 and less than 5:

```
x > 1 And x < 5
```

## Or Operator

Use this operator when you need to know if any of the comparisons evaluate to True. In the example below, if the variable x contains 10 then the expression evaluates to True:

```
x > 1 Or x < 5
```

This is because 10 is greater than 1. The fact that it is not less than 5 does not matter because you only care if at least one of the comparisons is True.

## Xor Operator

Use this operator when you need to know whether any of the comparisons evaluate to True, but not all of them. In the example below, if the variable x contains 10 then the expression evaluates to True.

```
x > 1 Or x < 5
```

This is because only one of the comparisons is True. However, if x contains 3 then the above expression returns False because both sub expressions are True.

## Not Operator

Use the Not operator to reverse the value of a boolean variable. For example, this return True when x is equal to or greater than 0:

```
Not x < 0
```

## Truth Table

A truth table can help you determine what the results of these various comparisons are.

Expression 1	Expression 2	And	Or	Xor
True	True	True	True	False
True	False	False	True	True
False	True	False	True	True
False	False	False	False	False

## Bitwise Comparisons

Sometimes it is helpful, usually in more advanced situations, to be able to compare the actual bits that make up Integers. You can do this using Bitwise comparisons, which work by treating the Integer as a binary number. With a binary number, you can then compare each individual bit to get a new set of bits, which results in a new number.

Bitwise comparisons work with the And, Or and Xor operators when they are used with Integers instead of Boolean expressions.

For example, consider the decimal numbers 5 and 3. Written in binary, they are:

101 (5)

011 (3)

To evaluate 5 Xor 3, you do the comparison on each bit. So for this example:

1 Xor 0 = 1

0 Xor 1 = 1

1 Xor 1 = 0

The new binary value is 110, which is decimal 6.

So this is the final result of the equation: 5 Xor 3 = 6.

The Not operator can also be used to compare Integers. If you pass an integer to Not, it simply reverses each bit value.

To evaluate Not 5:

Not 1 = 0

Not 0 = 1

Not 1 = 0

The new binary value is 010, which is decimal 2.

### Bitwise Tables

Results for And, Or and Xor:

Bit 1	Bit 2	And	Or	Xor
1	1	1	1	0
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0

Result for Not:

Bit 1	Not Bit 1
0	1

<b>Bit 1</b>	<b>Not Bit 1</b>
1	0

# Controlling Code Flow

## Table of Contents

- [If...Then...End If](#)
- [If...Then...Else...End If](#)
- [If...Then...Elseif...End If](#)
- [If...Then...Else](#)
- [If Operator](#)
- [Select...Case](#)

The methods you write execute one line at a time from top to bottom, left to right. There will be times when you want your application to execute some of its code based on certain conditions (using comparisons). When your application's logic needs to make decisions it's called branching. This allows you to control what code gets executed and when. There are two branching statements: `If...Then...End If` and `Select...Case`.

## If...Then...End If

The `If...Then...End If` statement is used when your code needs to test a boolean (True or False) expression and then execute code based on its result. If the expression you are testing is True, then the lines of code you place between the `If...Then` line and the `End If` line are executed, otherwise they are skipped.

```
If condition Then
  // [Your code goes here]
End If
```

Say you want to test the Integer variable `month` and if its value is 1, run your code:

```
If month = 1 Then
  // [Your code goes here]
End If
```

The part "`month = 1`" is a boolean expression; it's either True or False. The variable `month` is either 1 or it is not 1.

Suppose you have a Button that performs an additional task if a particular CheckBox is checked. The `Value` property of a CheckBox is Boolean so you can test it in an If statement easily:

```
If CheckBox1.Value Then
  // [Your code goes here]
End If
```

Remember that you can declare local variables using the `Dim` statement inside an If statement. However, such variables go out of scope after the `End If` statement. For example:

```
If error = -123 Then
  Dim a As Text
  a = "Oops! An error occurred."
```

```
End If
Label1.Text = a // out of scope!
```

If you need the variable after the End If statement, you should declare it local to the entire method, not within the If...End If statement.

## If...Then...Else...End If

An If...Then...End If statement can be expanded by including an additional Else clause. In some cases, you need to run some code if a boolean expression is True, but a different set of code if the boolean expression is False. In these situations, you can add the Else clause for the code that will run when the boolean expression evaluates to False.

This code sets the text based on a month variable:

```
Dim t As Text
If month = 1 Then
  t = "It is January"
Else
  t = "Not January"
End If
```

## If...Then...Elseif...End If

The next step in enhancing your If...Then...End If statement is to have multiple tests when the initial boolean expression is False. For each additional test, you use the Elseif statement.

```
Dim t As Text
If month = 1 Then
  t = "It is January."
ElseIf month < 4 Then
  t = "It is still Winter."
Else
  t = "It is not Winter."
End If
```

You could, of course, use an additional If...Then...End If statement inside the Else portion of the first If statement to perform another test. However, this adds another End If and needlessly complicates your code. Instead, you can use as many Elseif statements as you need, followed by an optional final Else statement:

```
Dim t As Text
If month = 1 Then
  t = "It's January."
ElseIf month < 4 Then
  t = "It's still Winter."
ElseIf month < 6 Then
  t = "It must be Spring."
Else
```

```
t = "Summer or Fall".  
End If
```

If the initial condition is False, your code continues to test the ElseIf conditions until it finds one that is True. It then executes the code associated with that ElseIf statement and continues executing the lines of code that follow the End If statement.

## If...Then...Else

Dialing it down a bit, a simple If statement can be written on one line, provided the code that follows the Then and the (optional) Else statements can all be written on one line. When you use this syntax, you omit the End If statement. Some examples:

```
Dim t As Text  
If error = 123 Then t = "An error occurred."  
If error = 123 Then t = "An error occurred." Else t = "Success"  
If error = 103 Then Break
```

## If Operator

For situations where you need to simply return a result based on a boolean expression, you can use the [If operator](#).

```
If(condition, resultIfTrue, resultIfFalse)
```

The condition is evaluated and if it is True, the resultIfTrue is returned, otherwise resultIfFalse is returned.

For example, this code outputs "Big":

```
Dim result As Text  
Dim myInteger As Integer = 41  
result = If(myInteger > 40, "Big", "Small")
```

Because this returns a result, the types of resultIfTrue and resultIfFalse must match (or be able to be converted between each other) and must match the destination type.

## Select...Case

When you need to test a property or variable for one of many possible values and then take action based on that value, use a Select...Case statement

Consider the following example that uses If...ElseIf..End If to test a variable (dayNumber) and display the day of the week:

```
Dim dayName As String  
If dayNumber = 2 Then  
    dayName = "Monday"  
ElseIf dayNumber = 3 Then
```

```

    dayName = "Tuesday"
ElseIf dayNumber = 4 Then
    dayName = "Wednesday"
ElseIf dayNumber = 5 Then
    dayName = "Thursday"
ElseIf dayNumber = 6 Then
    dayName = "Friday"
Else
    dayName = "the weekend."
End If
Dim result As Text = "It is " + dayName

```

No two of these conditions can be True at the same time. While this method of writing the code works, it's not that easy to read.

This next example uses a Select...Case statement to achieve the same result. It is far easier to read:

```

Dim dayName As Text
Select Case dayNumber
Case 2
    dayName = "Monday"
Case 3
    dayName = "Tuesday"
Case 4
    dayName = "Wednesday"
Case 5
    dayName = "Thursday"
Case 6
    dayName = "Friday"
Else
    MsgBox "the weekend."
End Select
Dim result As Text = "It is " + dayName

```

The Select...Case statement compares the variable or property passed in the first line to each value on the Case statements. Once a match is found, the code between that case and the next is executed.

Select...Case statements can contain an Else statement to handle all other values not explicitly handled by a Case.

You can create local variables using the Dim statement inside a Case statement. However, such variables go out of scope at the conclusion of the Case statement. For example:

```

Select Case dayNumber
Case 2
    Dim day As Text
    day = "Tuesday"
Else
    Dim day As Text ' new scope
    day = "It is NOT Tuesday!")
End Select

```

```
' day is now out of scope
```

The variable “day” should be declared prior to the Select...Case statement so that it is available after the End Select statement executes.

The Select...Case statement works with variables of any data type, including strings, integers, singles, doubles, booleans, and colors. For example, you can compare colors, as in the following example:

```
Dim c As Color
c = &cFF0000 ' pure red

Select Case c
Case &c00FF00 ' green
  MsgBox("Green")
Case &cFF0000 ' red
  MsgBox("Red")
Case &c0000FF ' blue
  MsgBox("Blue")
End Select
```

A Case statement can accept more than one value, with different values separated by commas. For example, the following is valid:

```
Dim c As Color
c = &cFF0000 ' red

Select Case c
Case &c00FF00, &cFF0000 ' green, red
  MsgBox("Green or Red")
Case &cFF0000 ' red
  MsgBox("Red")
Case &c0000FF ' blue
  MsgBox("Blue")
End Select
```

In the preceding example, the first Case statement is True, so its code executes. Although the color passed to Select...Case is Red, the code for the second case does not execute because it is not the first matching case.

The Select Case statement accepts an Else clause. The code in the Else clause executes only if none of the preceding cases match. The Else clause can be written as either "Else" or "Case Else". In the following example, the Case Else clause executes because the color FFFF00 does not match any of the Case statements:

```
Dim c As Color
c = &cFF0000 ' pure red

Select Case c
Case &c00FF00 ' green
  MsgBox("Green")
```

```

Case &cFF0000 ' red
  MsgBox("Red")
Case &c0000FF ' blue
  MsgBox("Blue")
Case Else
  MsgBox("None of the above")
End Select

```

The Case statement can also accept a range of consecutive values using the “To” keyword. For example:

```

Dim i As Integer = 53
Select Case i
Case 1 To 25
  MsgBox("25 or less")
Case 26 To 50
  MsgBox("26 to 50")
Case 51 To 100
  MsgBox("51 to 100")
End Select

```

In this example, the third case, “51 to 100”, is true.

You can combine ranges with nonconsecutive values, by separating them with commas, such as:

```
Case 0, 26 to 50, 75, 100 to 200
```

You can write inequalities with the “Is” keyword and an inequality operator. The syntax is:

```
Is inequalityOperator <value>
```

For example:

```

Dim i As Integer = 10
Select Case i
Case Is <= 10
  ' this case selected
Case Is > 10
  ' this case not selected
End Select

```

You can combine inequalities with values, as in:

```

Dim i As Integer = 75
Select Case i
Case 0, Is <= 10, 100
  ' case not selected
Case Is > 10, Is < 99
  ' case selected
End Select

```

You can even use functions that return a value of the specified data type in a Case statement. Here is a simple example:

```
Dim i As Integer = 4
Dim a As Integer = 2

Select Case i
Case CalcSquare(a)
  ' case 1
Case a
  ' case 2
Else
  ' no match
End Select
```

The function in the first Case statement is:

```
Function CalcSquare(a As Integer) As Integer
  Return a * a
End Function
```

In this example, the function squares the value passed to it, so the first Case statement matches.

In the case of a simple function like this, you can write the expression in the Case statement itself. That is, the following is an equivalent matching Case statement:

```
Case a*a
```

The Select Case statement can also compare variables that are Objects. The following example uses a Select...Case statement to determine which button the user pressed in a MessageDialog box. The Select Case statement compares objects of type MessageDialogButton to determine which of three possible dialog buttons was pressed.

```
Dim d As New MessageDialog
Dim b As MessageDialogButton

d.Icon = MessageDialog.GraphicCaution
d.ActionButton.Caption = "Save"
d.CancelButton.Visible = True
d.AlternateActionButton.Visible = True
d.AlternateActionCaption = "Don't Save"
d.Message = "Save changes before closing?"
d.Explanation = "If you don't save your changes, you will lose your work."

b = d.ShowModal
Select Case b
Case d.ActionButton
  // user pressed Save
Case d.AlternateActionButton
```

```
// user pressed Don't Save
Case d.CancelButton
  // user pressed Cancel
End Select
```

You can also use the IsA operator to determine whether an object is of a particular class. The syntax is:

```
Case IsA ClassName
```

Here is a simple example. The code in a PushButton Action event handler in a window:

```
Select Case Me
Case IsA PushButton
  MsgBox("I'm a PushButton.")
Case IsA TextField
  MsgBox("Nope!")
End Select
```

The term “Me” refers to the PushButton, so the first Case statement returns true.

## Nesting

All of these commands can be nested within each other in order to achieve the desired effect.

# Repeating Code

## Table of Contents

- [While...Wend](#)
- [Do...Loop](#)
- [Endless Loops](#)
- [Lengthy Loops](#)
- [For...Next](#)
- [For...Each](#)
- [Adding Loops Using the Code Editor](#)

There may be times when one or more lines of code need to be executed more than once. If you know how many times the code should execute, you could simply repeat the code that many times, but that is not an efficient way to write code. For example, if you wanted a PushButton to display a message three times when clicked, you could simply put the MsgBox method in your code three times like this:

```
MsgBox("One")  
MsgBox("Two")  
MsgBox("Three")
```

But suppose you need it to count fifty times or perhaps until a certain condition is met? Simply repeating the code over and over in these cases will either be just tedious or not possible. How do you solve this problem? The answer is a loop.

Loops execute one or more lines of code over and over again.

The following types of loop structures are available:

- **While...Wend:** The loop runs until the condition specified in the While statement is satisfied.
- **Do...Loop:** The loop runs until the condition specified in the Do or Loop statements are satisfied.
- **For...Next:** The loop runs a specified number of times given in the For statement. A local counter variable controls the execution of the loop.
- **For...Each:** The loop runs repeatedly for each element in an array.

You can declare local variables inside a loop structure. When you define a local variable inside a loop structure, its scope is local to the structure itself, not the entire method. It goes out of scope after the condition of the loop is satisfied. If you need to use the variable after the loop executes, define it outside the loop, so that it is local to the method.

## While...Wend

A While loop executes one or more lines of code between the While and the Wend statements. The code between these statements is executed repeatedly, provided that the condition passed to the While statement continues to evaluate to True.

Consider the following example:

```
Dim i As Integer
```

```
While i < 10
  i = i + 1
Wend
```

The variable “i” will be zero by default when it is created by the Dim statement. Because zero is less than ten, execution will move inside the While...Wend loop. The variable i is incremented by one and the loop returns to the top where the While statement checks to see if the condition is still True and if it is, then the code inside the loop executes again. This continues until the condition is no longer True. If the variable i was not less than ten in the first place, the contents of the loop are skipped entirely and execution would continue at the line of code after the Wend statement.

## Do...Loop

Do loops are similar to While loops but a bit more flexible. Do loops continue to execute all lines of code between the Do and Loop statements until a particular condition is True. While loops on the other hand execute as long as the condition remains True. Do loops provide more flexibility than While loops because they allow you to test the condition at the beginning or end of the loop. The example below shows two loops; one testing the condition at the beginning and the other testing it at the end:

```
Dim i As Integer
Do Until i = 10
  i = i + 1
Loop
```

```
i = 0
Do
  i = i + 1
Loop Until i = 10
```

The difference between these two loops is this: In the first case, the loop will not execute if the variable i is already equal to ten. The second loop executes at least once regardless of the value of i because the condition is not tested until the end of the loop.

It is possible to create a Do loop that does not test for any condition. Consider this loop:

```
Do
  i = i + 1
Loop
```

Because there is no test, this loop will run endlessly. You use the Exit command to force a loop to exit without testing for a condition. However, this is generally considered poor design because you have to read through the code to figure out what will cause the loop to end.

```
Do
  i = i + 1
  If i > 10 Then Exit
```

## Loop

# Endless Loops

Much like the situation with a Do...Loop without a condition, you should be sure that the code inside your While and Do loops eventually causes the condition to be satisfied. Otherwise, you will end up with an endless loop that runs forever. Should you do this accidentally, you can click the Stop button in the Debugger. If this doesn't work, you can always "force-quit" your running application using Task Manager on Windows, the Force Quit window on OS X or the Tasks window in Linux.

# Lengthy Loops

When a loop starts running, its process "takes over" and doesn't allow the user to interact with interface elements such as menus, buttons, and scroll bars. On modern computers and reasonably short loops, this isn't a problem because the loop executes faster than the user can think of another button to push or menu item to select. If this is not true, there are a couple of things you can do:

- If the user should wait until the loop is finished before doing anything else (e.g., if a user action might invalidate the results of the loop), you can signal that a lengthy operation is in progress by changing the mouse cursor to a "wait" cursor until the loop ends.
- If the user is permitted to do other tasks while the loop is running, you should consider using a thread. A thread runs code in the background, allowing the main application to handle user operations in the foreground.

# For...Next

While and Do loops are perfect when the number of times the loop should execute cannot be determined because it is based on a condition. A For loop is for cases in which you can determine the number of times to execute the loop. For example, suppose you want to add the numbers one through ten to a List Box. Since you know exactly how many times the code should execute, a For loop is the right choice. For loops also differ from While and Do loops because For loops have a loop counter variable, a starting value for that variable and an ending value. The basic construction of a For loop is:

```
Dim Counter As Integer
For Counter = 0 To 100
    // [your code goes here]
Next
```

Notice that the Dim statement declares the counter as an Integer. Although an Integer is the most common way to define the counter variable, you can also declare it as a Single or Double.

In this example, the counter variable was declared in the usual way, via the Dim statement. Since counter variables are rarely needed outside the For loop, you can also declare the counter variable right inside the For statement. In other words, you can redo this example like this:

```
For Counter As Integer = 0 To 100
```

```
// [your code goes here]
Next
```

Notice that the Dim statement has been removed from the example. If you declare the counter variable this way, you can use it only within the For loop. It goes out of scope after the For loop is finished. This is the recommended way to declare a counter variable. Of course, if you need to read or change the value of the counter variable outside the For loop, which is rarely necessary, you should use the Dim statement instead.

In the prior examples, the starting value and the ending value are specified as numbers. You can also use variables, as shown in this example:

```
Dim startingValue, endingValue As Integer
startingValue = 0
endingValue = 100

For counter As Integer = startingValue To endingValue
  // [your code goes here]
Next
```

The first time through the loop, the counter variable will be set to StartingValue. When the loop reaches the Next statement, the counter variable will be incremented by one. When the Next statement is reached and the counter variable is equal to EndingValue, the counter will be incremented and the loop will end.

Look back at the example mentioned earlier. You want to add the numbers one through ten to a List Box. The following code accomplishes that:

```
For i As Integer = 1 To 10
  ListBox1.AddRow(Str(i))
Next
```

The counter variable (i in this case) is passed to the Str function to be converted to a string so that it can be passed to the AddRow method of ListBox1.

 Note: The letter “i” is commonly used as the loop counter for historical reasons. In FORTRAN, the letters I through N are integers by default. Therefore, FORTRAN programmers began the practice of using those letters as counters, and in the order they appear in the alphabet. That is, if a FORTRAN programmer needed to nest one loop in another, he would use j as the counter for the inner loop. This convention made it easy for FORTRAN programmers to follow the logic of code that processed multi-dimensional arrays.

By default, For loops increment the counter by one. You can specify another increment value using the Step statement. In this example, the Step statement is added to increment the counter variable by 5 instead of 1:

```
For i As Integer = 5 To 100 Step 5
  ListBox1.AddRow(Str(i))
Next
```

In this example, the For loop starts the counter at 100 and decrements by 5:

```
For i As Integer = 100 DownTo 1 Step 5
  ListBox1.AddRow(Str(i))
Next
```

## Performance Considerations

So far, you have seen examples where StartingValue and EndingValue are integer numbers. If either StartingValue or EndingValue are expressions that must be evaluated to integers, the For loop will perform the evaluation each time it increments the counter — even if the expression always evaluates to the same integer.

Therefore, for performance reasons it is advisable to perform any evaluations before entering the loop. For example, consider a loop that needs to process all the fonts that are installed on the user’s computer. This number cannot be known in advance but there is a built-in function, FontCount, that you can use to obtain the total number of fonts. If you use it in the For statement to compute EndingValue (like so):

```
For i As Integer = 0 To FontCount-1
  .
  .
Next
```

The loop will run more slowly than if you calculate the value only once:

```
Dim numFonts As Integer = FontCount-1
For i As Integer = 0 To numFonts
  .
  .
Next
```

However, the difference in speed may be of no practical value unless it is an incredibly lengthy loop. On a typical computer the difference between these two loops is only small fractions of a second—not enough to lose sleep over.

## Nested Loops

A For loop (as well as any other kind of loop) can have another loop inside it. In the case of a For loop, the only thing you will have to watch out for is making sure that the counter variables are different so that the loops won’t confuse each other. The example below uses a For loop embedded inside another For loop to go through all the cells of a multi-column ListBox counting the number of cells in which the word “Hello” appears:

```
Dim count As Integer
For row As Integer = 0 To ListBox1.ListCount-1
  For column As Integer = 0 To ListBox1.ColumnCount-1
    If ListBox1.Cell(row, column) = "hello" Then
      count = count + 1
    End If
  End For
End For
```

```
Next
Next
MsgBox(Str(count))
```

One way to help keep this readable is to include the counter variable with the Next statement so you can more easily see which loop is which:

```
Dim count As Integer
For row As Integer = 0 To ListBox1.ListCount-1
  For column As Integer = 0 To ListBox1.ColumnCount-1
    If ListBox1.Cell(row, column) = "hello" Then
      count = count + 1
    End If
  Next row
Next count
MsgBox(Str(count))
```

## For...Each

Another situation in which you want to loop through a group of values is array processing. Rather than looping through a set of statements for each value of a counter, the For...Each statement processes each element of an array that is passed to it.

Take a look at an example to sum the values of an array using a counter variable:

```
Dim values() As Double
values = Array(2.2, 1.1, 3.3, 4.4)

Dim sum As Double
Dim element As Double

For i As Integer = 0 To values.Ubound
  sum = sum + values(i)
Next
```

Here is the same example using For...Each:

```
Dim values() As Double
values = Array(2.2,1.1,3.3,4.4)

Dim sum As Double
Dim element As Double

For Each element In values
  sum = sum + element
Next
```

In the For...Each statement, instead of a counter variable, there is a variable that automatically gets the value of

each element in the array. In the above example, the element variable gets the next value in the values array each time through the loop. When you are working with arrays, For...Each allows you to write simpler code.

Since the array doesn't necessarily have to be numbers, this statement enables you to process a group of objects of any type. They could be pictures, colors, documents, sets of database records, and so forth. As is the case for the For...Next loop, you can declare the data type of the element variable inside the For...Each statement rather than in a separate Dim statement. For example the previous example could be rewritten like this:

```
Dim values() As Double
values = Array(2.2, 1.1, 3.3, 4.4)

Dim sum As Double

For Each element As Double In values
  sum = sum + element
Next
```

## Adding Loops Using the Code Editor

The Code Editor's contextual menu offers an especially convenient way of adding loops to your code. The last three items in the contextual menu wrap the selected lines of code inside a type of loop. To use these menu items, simply write the code that goes inside the loop, select the lines, and then choose the type of loop from the contextual menu. Your choices are If...End If, Do...Loop, and While...Wend.

The Code Editor can wrap the selected lines in the loop but it doesn't know what condition should terminate the loop. Therefore, it inserts the placeholder text `_condition_` in the loop statement. This condition is selected for you so all you need to do is start typing to replace it with the condition you want.

# Properties

A property is a value of a class or module. You can think of it as a variable that belongs to a class or module. Most of the built-in classes have properties. For example, the `Xojo.Core.Date` class has a property to get the Year.

## Creating Your Own Properties

There are two types of properties you can add: a (instance) property and a computed property.

### Adding a Property

Properties can be added to project items, including classes (including Windows, Web Pages, Views) and modules.

There are several way to add a property:

- Insert button on the main toolbar
- Add button on the Code Editor toolbar
- Insert → Property from the menu
- The contextual menu
- Keyboard shortcut (Option-Command-P on OS X or Ctrl+Shift+P on Windows and Linux)

To quickly create a property, you can enter both its name and type on one line in the Name field

- like this: **PropertyName As DataType**. When you leave the field, the type will be set in the Type field.

Any of these adds the property and displays a blank editor (where you can type notes) and displays the Inspector where you can set the values for the property:

- **Name**  
The name of the property. Just like variables, properties are given names to describe them and the same naming rules apply.
- **Type**  
The Type can be any valid data type, including classes.
- **Default**  
The default value for the type. For example, for an Integer property you can specify 42 to have its default value be 42.
- **Scope**  
Scope indicates what parts of your code can call the method. Choices are Public, Protected and Private.
  - Public methods can be called from anywhere in your code with no restrictions.
  - Protected methods have some restrictions, which vary depending on where the method is located (class or module).
  - Private methods can only be called by the module or class that contains the method.

The property signature (its name, and type) also appears at the top of the Editor for reference.

Properties are referenced by their name when used within the module or class that owns them:

```
TeamName = "Flying Squirrels"
```

For usage outside of the class or module that owns them, they are often referenced differently. Refer to the [Modules](#) and [Classes](#) sections for specifics.

## Adding Computed Properties

A computed property is a property that does not store its value but instead calculates it each time it is accessed.

There are several ways to add a computed property:

- Insert button on the main toolbar
- Add button on the Code Editor toolbar
- Insert → Property from the menu
- The contextual menu

Like with a property, you have to specify the values for the computed property:

- **Name**

The name of the property. Just like variables, properties are given names to describe them and the same naming rules apply.

- **Type**

The Type can be any valid data type, including classes.

- **Scope**

Scope indicates what parts of your code can call the method. Choices are Public, Protected and Private.

- Public methods can be called from anywhere in your code with no restrictions.
- Protected methods have some restrictions, which vary depending on where the method is located (class or module).
- Private methods can only be called by the module or class that contains the method.

The property signature (its name, and type) also appears at the top of the Editor for reference.

A computed property has two additional parts to it: a getter and a setter. A computed property can be expanded in the Navigator to display the Get and Set sections below the property name. You add code to the Get section to get a value for the property. You add code to the Set section to set a value for the property. If you do not include code in Set then the property becomes a read-only property. You can also leave out code in the Get section to make the property write-only, but that is rarely useful.

An example of a computed property might be a TotalPrice property of a Product class. You could implement TotalPrice with this code in Get:

```
Return Price * Quantity
```

If your computed property needs to retain a value, often you would create a matching private property to do so (usually prefixed with an “m”). For a computed property called "TeamName As Text", this means you might do something like this:

Get:

```
Return mTeamName
```

Set:

```
mTeamName = Value
```

Which is referring to a companion property:

mTeamName As Text

Code can access the computed property by its name:

```
Dim tName As Text = myClass.TeamName
```

Using a computed property in this manner is not much different than a regular property, but it does have at least one benefit: you can set a breakpoint in the Set section so you can track when the property value changes.

# Methods

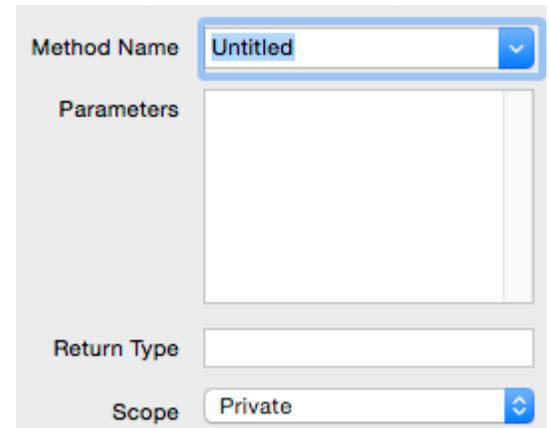
Now that you understand the various coding concepts, now it is time to learn more about the place where you write your code: methods.

Methods are the building blocks of your application. The code you write most often exists in a method. A method is one or more instructions that are performed to accomplish a specific task; an action of some sort.

There are many built-in methods. For example, the Quit method causes your application to quit. Most classes have built-in methods. For example, the ListBox class has a method called AddRow for adding rows to it (as the name implies). And you can of course, create your own methods.

## Creating Your Own Methods

Methods can be added to project items, including classes (including Windows, Web Pages, Views) and modules. To add a method, choose Insert -> Method from the menu or toolbar when the appropriate project item is selected. This adds the method, displays a blank code editor and displays the Inspector where you can set the properties for the method:

A screenshot of the Xojo Inspector window. It has a light gray background. At the top, there is a label 'Method Name' followed by a text box containing 'Untitled' and a blue dropdown arrow. Below that is a label 'Parameters' followed by a large, empty white rectangular area. Further down is a label 'Return Type' followed by an empty white text box. At the bottom is a label 'Scope' followed by a text box containing 'Private' and a blue dropdown arrow.

- **Method Name**  
The name of the method. Just like variables, methods are given names to describe them and the same naming rules apply.
- **Parameters**  
Parameters are values that you pass to the method that it can then use within the method as if they were variables. Parameters are separated by commas and are declared similarly to how you use the Dim statement:  
`value As Integer, name As Text</code>`
- **Return Type**  
Methods that do not specify a return type are called Subroutines (or Procedures in some other languages). Methods that specify a return type are called Functions. The Return Type can be any valid data type, including classes.
- **Scope**  
Scope indicates what parts of your code can call the method. Choices are Public, Protected and Private.
  - Public methods can be called from anywhere in your code with no restrictions.
  - Protected methods have some restrictions, which vary depending on where the method is located (class or module).

- Private methods can only be called by the module or class that contains the method.

The method signature (its name, parameters and return type) also appears at the top of the Code Editor for reference.

## Using Methods

To use a method you specify its name, optionally prefixed by the module or class instance name (depending on how the method is defined). Classes are discussed in the Object-Oriented Programming section and Modules are discussed in the Modules section.

An example of a method you have used before is the `ToText` method on the number data types:

```
Dim i As Integer = 42
Dim t As Text = i.ToText
```

## Passing Parameters to Methods

Some methods, such as the `ToText` method shown above, are called simply with their name. But some methods require additional information or values. This information that is passed to a method is called a parameter. A method can have any number of parameters.

If you have a Dictionary containing a value for key "Name", then this method call removes the key and value from the Dictionary:

```
myDictionary.Remove("Name")
```

The value "name" was passed as a parameter to the `Remove` method.

Methods define their parameters and specify the types for them. A method can have any number of parameters. When passing multiple parameters, separate each one with a comma.

The `Lookup` method on a Dictionary takes two parameters and can be used to get a value for a key and if not available, provide a default:

```
Dim value As Text = myDictionary.Lookup("Name", "Unknown")
```

In these examples, the parameters have been passed as text literals, but you can also pass variables and constants as parameters instead:

```
Dim key As Text = "Name"
Dim default As Text = "Unknown"
```

```
Dim value As Text = myDictionary.Lookup(key, default)
```

As with variable assignment, the parameters you pass to the method must match the types of the parameters as declared on the method.

## Passing Arrays as Parameters

An array can be passed as a parameter in a call to a method or function. You can pass both one and multi-dimensional arrays. To specify that a parameter is a one-dimensional array, put empty parentheses after its name in the declaration. For example:

```
names() As Text
```

can be used in the declaration when you want to pass an array of strings to the method. Since you do not need to specify the number of elements in the array to be passed, you can pass a different number of elements at different places in your code. When you pass an array to the method or function, omit the parentheses in the array. For example:

```
PrintLabels(allNames)
```

PrintLabels is the name of the method that accepts the text array as its parameter.

You can pass multi-dimensional arrays without specifying the number of elements in each dimension, but you need to indicate the number of dimensions. Do this by placing one fewer commas in the parentheses than dimensions. For example, if names were a two-dimensional text array, you would declare the array in the following manner:

```
names(,) As Text
```

When you pass a multi-dimensional array to a method or function, you can include the parentheses but not any commas:

```
PrintLabels(allNames())</code>
```

## Returning Values from Methods

Some methods return values. These methods are called Functions. When a method returns a value, the value is passed back from the method to the line of code that called the method. For example, the method Ticks returns the number of ticks (a tick is 1/60th of a second) that have passed since you turned on your computer. You can assign the value returned by a method the same way you assign a value to a variable. In the example below, the value returned by Ticks is assigned to the variable *elapsed*:

```
Dim elapsed As Double  
elapsed = Ticks
```

Some methods require parameters and return a value. You saw this earlier with the Lookup method for a Dictionary:

```
Dim value As Text = myDictionary.Lookup("Name", "Unknown")
```

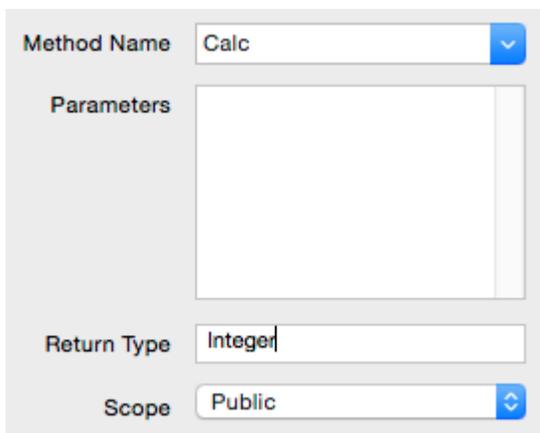
The method takes two parameters and returns a value.

You can also directly pass methods that return values to other methods:

```
Dim t1 As Text = "Hello, how are you?"
Dim t2 As Text = "Hello, I am fine."

Dim b As Boolean
b = t1.BeginsWith(t2.Left(5)) ' True</code>
```

For a method to return a value, you just have to specify a Return Type in the method properties of the Inspector. For example, this method definition returns an Integer:

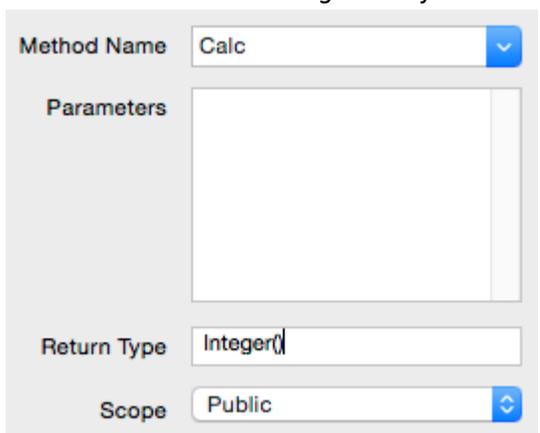


The Inspector window shows the following properties for a method named 'Calc':

- Method Name: Calc
- Parameters: (Empty list)
- Return Type: Integer
- Scope: Public

A method can return any type, from the intrinsic data types to any class or other type you have created yourself.

Methods can also return arrays of any type. To indicate that the return type is an array, add the double parenthesis to it. This returns an Integer array:



The Inspector window shows the following properties for a method named 'Calc':

- Method Name: Calc
- Parameters: (Empty list)
- Return Type: Integer()
- Scope: Public

## Passing Parameters by Value and by Reference

By default, you pass values to a method by value. When you do so, the method receives a copy of the data that you pass allowing the method to modify it without affecting the original value.

## Passing by Value

Parameters passed by value are essentially treated as local variables inside the method — just like variables that are created using the Dim statement. This means that you can modify the values of the parameters themselves rather than first assigning the parameter to a local variable. For example, if you pass a value to the Integer parameter “x”, you can increment or decrement the value of x rather than assigning the value of x to a local variable that is created using Dim.

The screenshot shows a configuration window for a method named 'SquareIt'. The 'Method Name' field contains 'SquareIt'. The 'Parameters' field contains 'num As Integer'. The 'Return Type' field contains 'Integer'. The 'Scope' field contains 'Public'.

This example modifies the parameter and returns the result:

```
Function SquareIt(num As Integer) As Integer
    num = num * num
    Return num
End Function
```

The original value passed to the method is not changed, as shown in this code:

```
Dim value As Integer = 10
Dim result As Integer
result = SquareIt(value)
' value remains 10
' result is 100
```

## Passing by Reference

When you write your own methods, you have the option of passing information by reference. The practical advantage of this technique is that the method can change the values of each parameter and replace the values of the parameters with the changed values. When you pass parameters by value, you can't do this because the parameter only represents a copy of the data itself.

- i When you pass information by reference, you actually pass a pointer to the object containing the information.
- i All arrays and any class types are always passed by reference regardless of what is specified in the method declaration.

Here the SquareIt example from above written to use ByRef:

The screenshot shows a configuration window for a method named 'SquareIt'. The 'Method Name' field contains 'SquareIt'. The 'Parameters' field contains 'ByRef num As Integer'. The 'Return Type' field is empty. The 'Scope' dropdown menu is set to 'Public'.

```
Sub SquareIt(ByRef num As Integer)
  num = num * num
End Sub
```

Since this modifies the parameter value, it is called like this:

```
Dim value As Integer = 10
SquareIt(value)
' value is now 100
```

## Optional Parameters and Default Values

There are two ways to make a parameter optional, both of which you do at the time you write the method declaration:

- Assign a value to it in the declaration
- Use the Optional keyword in the declaration

The way to make the parameter's explicit status clear is to use the Optional keyword. This modifier precedes the parameter name and may be used with or without a default value. Should caller omits passing this parameter, the parameter will receive the default value for its data type if no default value was passed in the declaration. For example:

```
MyMethod(a As Integer, b As Integer, Optional c As Integer)
```

The parameter c is optional. If it is omitted in the method call, then it gets the default value of Integer, which is 0.

If you want a different default value, you assign it to the parameter:

```
MyMethod(a As Integer, b As Integer, Optional c As Integer = 5)
```

Now if the parameter c is omitted in the method call, then it gets the value 5.

Lastly, you can omit the Optional keyword from the declaration if you specify a default value:

```
MyMethod(a As Integer, b As Integer, c As Integer = 5)
```

With any of the the above method declarations, you can call the method like this:

```
MyMethod(3, 4)
```

## Identifying the Currently Executing Method Name

For logging purposes, it can often be useful to know the name of the currently running method. Use the `CurrentMethodName` method in your code to get the name of the currently running method:

```
MsgBox(CurrentMethodName)</code>
```

## Setters

A value passed to a method is normally supplied in parenthesis following the method name, such as:

```
MyMethod(10)
```

Sometimes it is preferred to have the method call behave differently so that you assign the parameter like this:

```
MyMethod = 10
```

This is called a Setter because it looks like you are setting the `MyMethod` value to 10 rather than passing 10 as a parameter to `MyMethod`. You can enable this alternative syntax by using the `Assigns` keyword in the parameter declaration:

```
Sub MyMethod(Assigns value As Integer)
```

You can have more than one parameter with this technique but only the first parameter can have the `Assigns` keyword. The other parameters are passed as before. For example:

```
Sub MyMethod(Assigns value As Integer, value2 As Integer)
```

To call this method:

```
MyMethod(5) = 10
```

## Events

Events are a special type of method commonly used with control classes. But they can also be implemented in your own non-control classes as well. Events can be called only by the class that declares the event. Generally speaking, events are used to provide a way for subclasses to provide additional functionality.

You don't need to worry about Events right now. Both subclasses and events are described in more detail in the Object-Oriented Programming chapter.

# Enumerations

An enum or [enumeration](#) is a data type consisting of a set of named values, which are called elements.

## Creating Your Own Enumerations

Enumerations can be added to project items, including classes (including Windows, Web Pages, Views) and modules. To add an enumeration, choose Insert -> Enumeration from the menu or toolbar when the appropriate project item is selected. This adds the Enumeration, displays a blank Enumeration Editor and displays the Inspector where you can set the properties for the enumeration:

- Name  
The name of the enumeration.
- Type  
Enumerations are always an Integer type and default to “Integer”. You can change the type to other Integer types (such as UInt64) should you need to use larger values for the enumeration elements.
- Scope  
Scope indicates what parts of your code can call the method. Choices are Public, Protected and Private.
  - Public methods can be called from anywhere in your code with no restrictions.
  - Protected methods have some restrictions, which vary depending on where the method is located (class or module).
  - Private methods can only be called by the module or class that contains the method.

In the Enumeration Editor, you use the “+” icon to add a new enumeration element to the set. Use the “-” button to remove an element from the set. When you add an element, you also give it a name. The name is used to refer to the value. You can edit the name by clicking on it once to select it and a second time to edit it.

 Note: The enumeration name cannot be left blank.

Although enumeration elements are integers behind the scenes, you don’t normally worry about the integer values. However, should you need to assign a specific integer value to an enumeration item, you can do so by assigning it as part of the name like this:

```
EnumElement = 10
```

To use an enumeration in code, you refer to the container of the enumeration, the enumeration name and then the enumeration element name:

```
Dim pictureSize As Module1.Size ' Size is an enumeration
pictureSize = Module1.Size.Large ' Large is an element on Size
```

If you find you need the actual integer value stored by the enumeration, then you have to cast it to an integer first:

```
Dim pictureSize As Integer
pictureSize = Integer(Module1.Size.Large)
```

# Modules

A module is a collection of project items, usually methods, properties and constants. But a module can also contain other project items such as classes or even other modules. In fact, modules can pretty much contain anything except layouts such as Windows, Web Pages, Views and Container Controls.

## When to use a Module

In general, modules should be used sparingly in an object-oriented language such as Xojo. In most cases, you will probably be better served by a class. With that said, here are some common (and valid) uses for modules:

- Global constants, particularly for localization
- Grouping project items into namespaces
- Class extensions
- Global properties and methods, but these should be minimal

Modules are covered here first since they are easier to use and learn and transition nicely to classes (which are covered in the next chapter).

## Adding a Module to Your Project

You can add a new module to your project by clicking the Insert button on the toolbar and selecting Module (or by using the menu Insert → Module or the contextual menu). The new module appears in the Navigator with a default name (the first module you add will be named **Module1**, for example). You can use the Inspector to rename the module to something more appropriate. For example, if the module contains financial functions, you might name it **Financial**.

You edit modules with the Code Editor: simply click the module's name in the Navigator. Modules are identified by their special icon in the Navigator.

Modules primarily contain properties, constants and methods so those are covered first.

### Adding Properties

Properties are variables that belong to the entire module rather than a single method. There are several ways to add a property to a module:

Name	Untitled
Type	Integer
Default	
Scope	Global

- Insert button on the main toolbar
- Add button on the Code Editor toolbar
- Insert → Property from the menu
- The contextual menu
- Keyboard shortcut (Option-Command-P on OS X or Ctrl+Shift+P on Windows and Linux)

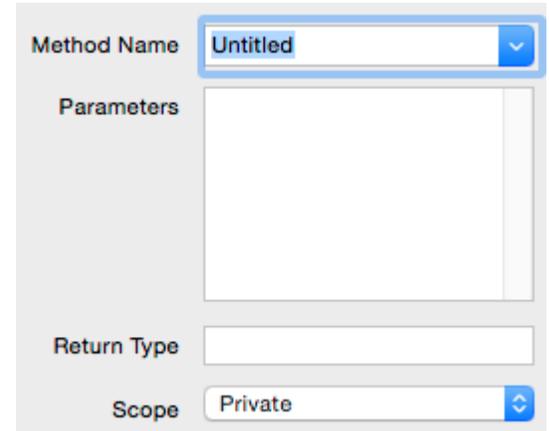
You can set the property Name, Type, Default value and Scope using the fields in the Inspector.

To quickly create a property, you can enter both its name and type on one line in the Name field like this: **PropertyName As DataType**. When you leave the field (or press return), the type will be set in the Type field.

You can also add computed properties to modules.

## Adding Methods

Methods were discussed in the previous section. There are several ways to add a method to a module:



The image shows a screenshot of the Inspector panel for adding a method. It contains the following fields:

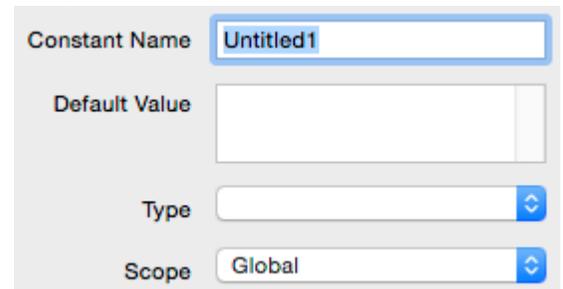
- Method Name:** A text field containing "Untitled" with a dropdown arrow on the right.
- Parameters:** A large empty text area for defining method parameters.
- Return Type:** An empty text field for specifying the return type.
- Scope:** A dropdown menu currently set to "Private".

- Insert button on the main toolbar
- Add button on the Code Editor toolbar
- Insert → Method from the menu
- The contextual menu
- Keyboard shortcut (Option-Command-M on OS X or Ctrl+Shift+M on Windows and Linux)

You can set the Method Name, Parameters, Return Type and Scope using the Inspector.

## Adding Constants

Constants added to a module work like constants added to a method except they are accessible anywhere in the module and possibly to code not in the module as well (depending on the scope).



The image shows a screenshot of the Inspector panel for adding a constant. It contains the following fields:

- Constant Name:** A text field containing "Untitled1".
- Default Value:** An empty text field for specifying the default value.
- Type:** An empty text field with a dropdown arrow on the right.
- Scope:** A dropdown menu currently set to "Global".

There are several ways to add a constant to a module:

- Insert button on the main toolbar
- Add button on the Code Editor toolbar
- Insert → Method from the menu
- The contextual menu
- Keyboard shortcut (Option-Command-C on OS X or Ctrl+Shift+C on Windows and Linux)

You can set the Constant Name, Default Value, Type (Number, String, Boolean, Color, Text) and Scope using the Inspector.

## Adding Other Items

You can also add other items to a module, including: Enumerations, External Method, Note, Structure, Using Clause.

### Enumeration

An Enumeration is a data type consisting of a set of named values.

### External Method

Add a reference to an external method (also called a Declare), which can be a method on a DLL (on Windows), a dylib (on OS X) or a shared library (on Linux).

### Note

A Note is essentially a text field that is added to a project item. Use a Note to add comments, description or any other text to the module that is not part of the code. Think of a Note as a giant comment.

### Structure

Adds a Structure, which is an advanced data type typically used with Declares.

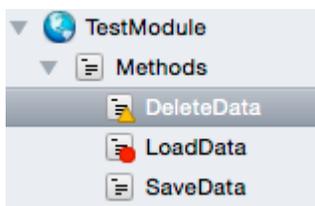
### Using Clause

Adds a Using statement that is applied to the entire module. For more about Using, see [Namespaces](#) below.

## Other Project Items

### Scope of Items

When you add an item to a module, you need to set its Scope using the Scope property in the Inspector. The Scope is also indicated in the Navigator. The Scope of an item determines its accessibility to other items in the project.



There are three choices:

- **Global:** A Global item can be called from code anywhere in the project. For example, you can use a global property to store a piece of information that needs to be available from several different Windows (or Web Pages or Views). To reference a global item in your code, just use its name. Global can only be used for items

added to a top-level module. If you add a global property called **MyGlobalProperty As Text**, then you can refer to it any code just by its name:

```
Dim value As Text = MyPublicProperty</code>
```

- **Public:** A Public item can also be called from code anywhere in the project. To reference a public item in a module, you use "dot" notation and prefix it with the name of the module. For example, if in Module1 you declare a Public property called **MyPublicProperty As Text**, then you call it like this in code outside of Module1:

```
Dim value As Text = Module1.MyPublicProperty
```

For code within Module1 you can just refer to it using **MyPublicProperty**.

- **Private:** A Private item is available only within the module. It is "invisible" to the rest of the application. To reference the Private item inside the module in which it was created, you simply use its name:

```
Dim value As Text = MyPrivateProperty</code>
```

## Extension Methods

An extension method is a special method definition that can be called using syntax that indicates that it belongs to another object. For example, you can add a method that is called from any FolderItem object to save something in a particular format. After you add the class extension method to a module, you can call it as if it were built into the FolderItem class. Using these methods does require an understanding of Classes, which are covered in the Object-Oriented Programming chapter.

This feature is useful when you would like to add a method to a class for which you do not have access to the source code. In particular, it allows you to add functionality to built-in framework classes.

To define a method as a class extension method, use the [Extends](#) keyword prior to the first parameter. The data type of the first parameter is the object type from which the method must be called. In other words, the use of the Extends keyword indicates that the parameter is to be used on the left side of the dot (".") operator in a calling statement. When you use the Extends keyword, you do not have to pass an object of that type to the method. You can add "normal" parameters to the parameter declaration that follow the Extends parameter.

The Extends keyword can be used only with Global methods that reside in modules.

### Creating and Using an Extension Method

As an example, say you want to add a Clear method that can be called from a TextArea control to clear all its text. If you don't already have a module in your project, add one.

In the module, add a method and name it Clear. It's parameter is:

```
Extends ta As TextArea
```

And its code is:

```
ta.Text = ""
```

You can now call this method on any TextArea control in the project. For example, a button could have this code to clear TextArea1 on the Window:

```
TextArea1.Clear
```

Extension methods for for all classes, including the ones included with Xojo and any that you create.

## Namespaces

Modules can also contain other project items, including classes, class interfaces and even other modules.

**i** Modules cannot contain Windows, Web Pages, iOSViews or Containers.

When you add other project items to a module, you have created what is called a Namespace. A Namespace is a collection of (typically related) items. The Xojo framework itself is an example of a nested namespace. The main Xojo namespace is a module and within it are other modules such as Core, IO, etc and those modules contain classes.



## The Using Statement

You can use the Using statement to avoid having to write the full namespace name each time you want to access an item within it. For example, you can write code like this to avoid having to use full dot notation to access classes within Module1:

```
Using Xojo.Core
Dim d As New Dictionary ' instead of Xojo.Core.Dictionary
d.Value("test") = "Hello"
```

Alternatively, you can add a Using Clause to a module to apply a namespace to all the methods of the module. To do so, choose Insert → Using Clause and enter the namespace name in the Inspector field.

# External Methods and Structures

## External Methods

An External Method adds a reference to an external method which can be a method on a DLL (on Windows), a dylib (on OS X) or a shared library (on Linux).

You can also use the Declare command to create an external method.

The image shows a screenshot of the Xojo Inspector for an External Method. The form is titled 'External Method' and contains the following fields:

- Method Name:** center
- Parameters:** windowRef As Integer
- Return Type:** (empty)
- Scope:** Protected
- Lib:** Cocoa
- Selector:** center
- Objective-C:**

External methods are added using Insert->External Method for a class or module. In the Inspector for the External Method, you can provide the necessary information, including:

- **Method Name:** The name of the method. This is the name you will use to call the method.
- **Parameters:** The parameters required by the method.
- **Return Type:** The return type of if the method is a function.
- **Scope:** The scope of the method matches the scoping rules for items in a module or class.
- **Lib:** The library containing the method.
- **Alias:** This is optional and refers to the actual method name in the library. You only need to provide it if you are using a different Method Name than what the method is called in the library.
- **Soft/Objective-C:** Select Soft to only check if the method exists at run-time. Check Objective-C to add an external method to an Objective-C method. This changes the Alias field to say "Selector" where you can specify the selector to call the method.

For more information on creating External Methods, refer to the [Declare](#) command in the Reference Guide.

## Structures

A Structure is a compound value type. It consists of a series of fields that are grouped together as a single block. You can control the size and order of the fields. A structure can provide a convenient alternative to a MemoryBlock as they are also often used to group together information for external function calls.

The values of structures have specific sizes giving the structure its own specific size.

Structures are added to project items such as modules, classes and windows. To add a structure, select the Add

button on the toolbar and select Structure from the menu (or use the Insert menu in the main menu bar). This displays the Structure Editor where you specify the fields in the structure and their sizes.

Use the “+” button to add a new field. Fields are added by entering their name followed by the type in this form:

```
Name As Type
```

So to enter an Integer you would do:

```
Age As Integer
```

Note that when you enter a field, it shows the size. This is the amount of bytes that the field uses in the structure. Integers use 4 bytes (in 32-bit apps), for example.

## Array of Bytes

The preferred way to deal with strings of text in a Structure is to create an array of Bytes, which you can do like this:

```
Name(50) As Byte
```

## Strings

 Not supported on iOS. Use an Array of Bytes instead.

Strings are a special case because you have to specify the exact size of the string in bytes. Unlike normal strings, a string in a structure always takes up the specified size in the structure and you cannot exceed it. Also, strings in structures do not contain encoding information. The syntax is:

```
Name As String*size
```

So to have a String with a size of 50:

```
Name As String*50
```

When you are creating a structure to pass to an external method, be sure that your sizes precisely match the sizes specified by the API of the method.

## Usage

When you are creating a structure to pass to an external method, be sure that your sizes precisely match the sizes specified by the API of the method.

### TestStructure

	Declaration	Offset	Size
⊖	Age As Integer	0	4
⊖	Name(50) As Byte	4	51
+			55

You can reference a structure using dot notation:

```
Dim cust As CustomerStruct  
cust.Age = 60
```

Structures are useful for organizing data, but they are not object-oriented and are limited in many ways (such as with string sizes). For your project's internal data management, you should almost always use a class over a structure.

However, structures are small and memory efficient. In addition to being useful when used with external methods, they may also be useful in specific situations where memory must be managed carefully.

# Overview

Object-oriented programming (OOP) is a style of programming based on "objects", which are data structures that contain data (properties) and code (methods). These data structures are called "classes". Classes are the fundamental building blocks of all Xojo applications.

This chapter covers object-oriented concepts, including classes and concepts that apply to classes: Encapsulation, Overloading, Inheritance and Polymorphism.

## Overview

The concepts you have previously learned about (in the [Xojo Language](#) chapter) are shared with OOP, including

- [Variables and Constants](#)
- [Data Types](#)
- [Properties](#)
- [Methods](#)
- [Enumerations](#)

And the entire Xojo language.

With object-oriented programming, you'll learn how classes are used to create objects that are then used to create reusable code in your projects.

If you are familiar with object-oriented program, Xojo uses an object model that is similar to C#, Java and other single-inheritance OOP languages.

## OOP Topics

- [Classes](#)
- [OOP Design Concepts](#)
- [Properties, Methods and Events](#)
- [Constructors and Destructors](#)
- [Interfaces](#)
- [Subclassing Examples](#)
- [Advanced Features](#)

## Additional Learning

The User Guide does not cover object-oriented programming in its entirety; it's a large subject. Here are some resources to help you learn more:

- [Video: Introduction to Object-Oriented Programming 101](#)
- [Video: Introduction to Object-Oriented Programming 201](#)
- [Stanford Programming Methodology Course on iTunes U](#)

# Classes

## Table of Contents

- [Overview](#)
- [Understanding Classes, Interfaces and References](#)
- [Adding Classes to Your Projects](#)
- [Adding Properties to Classes](#)
  - [Shared Properties](#)
- [Adding Methods to Classes](#)
  - [Shared Methods](#)

## Overview

In its simplest form, a class is a container of code and data much like a module. But unlike a module, a class provides better code reuse. Classes are the fundamental building blocks of object-oriented programming.

Classes offer many benefits, including:

- **Reusable Code**

When you directly add code to a PushButton on a Window to customize its behavior, you can only use that code with that one PushButton. If you want to use the same code with another PushButton, you need to copy the code and then make changes to the code in case it refers to the original PushButton (since the new PushButton will have a different name than the original).

Classes store the code once and refer to the object (like the PushButton) generically so that the same code can be reused any number of times without modification. If you create a class based on the PushButton control and then add your code to that class, any usage (instances) of that custom class will have that code.
- **Smaller Projects and Applications**

Because classes allow you to store code once and use it over and over in a project, your project and the resulting application is smaller in size and may require less memory.
- **Easier Code Maintenance**

Less code means less maintenance. If you have basically the same code copied in several places of your application, you have to keep that in mind when you make changes or fix bugs. By storing one copy of the code, when you need to make changes, you'll spend less time tracking down all those places in your project where you are using the same code. Changes to the code in a class are automatically used anywhere where the class is used.
- **Easier Debugging**

The less code you have, the less code there is to debug.
- **More Control**

Classes give you more control than you can get by simply adding code to the event handlers of a control in a window. You can use classes to create custom controls. And with classes, you have the option to create versions that don't allow access to the source code of the class, allowing you to create classes you can share or sell to others.

## Understanding Classes, Instances and References

Before you can use a class in your project, it is important to understand the distinction between these three

concepts: the **class** itself, the **instance** of the class and the **reference** to the class.

The Class	The Instance
Think of the class as a template for a container of information (code and data), much like a module. And like a module, each class exists in your project only once. But unlike a module, a class can have multiple instances that each contain different data.	Classes provide better code reuse because of a concept called instances. Unlike a module, which exists only once in your application, a class can have multiple instances. Each instance (also called an object) is a separate and independent copy of the class and all its methods and properties.

There are many built-in classes for user interface controls such as PushButton, WebButton, iOSButton, Label, WebLabel, iOSLabel, TextField, WebTextField, iOSTextField, ListBox, WebListBox and iOSTable.

By themselves, control classes are not all that useful; they are just abstract templates. But each time you add a control class to a layout (a window, web page or view) you get an instance of the class. Because each button is a separate instance, this is what allows you to have multiple buttons on a window, with each button being completely independent of each other with its own settings and property values.

These instances are what you interact with when writing code on the window and is what the user interacts with when they use your application.

For example, when you drag a TextArea from the Library to a window, you create a usable instance of the TextArea on the window. The new instance has all the properties and methods that were built into the TextArea class. You get all that for free — styled text, multiple lines, scroll bars, and all the rest of it. You customize the particular instance of the TextArea by modifying the values of the instance's properties.

When you add a control to a window (or web page), the Layout Editor creates the reference for you automatically (it is the name of the control).

When you write code, you create instances of classes using the New keyword. For example, this create a new instance of a Vehicle class that is in your project:

```
Dim car As New Vehicle
```

## The Reference

A reference is a variable or property that refers to an instance of a class. In the above code example, the car variable is a reference to an instance of the Vehicle class.

You interact with properties and methods of the class using dot notation like this:

```
Dim car As New Vehicle
car.Brand = "Ford"
car.Model = "Focus"
```

Here the Brand and Model were defined as properties of the Vehicle class and they are given values for the specific instance. If you try to access a property or method of a class without first have an instance, you will get a NilObjectException. This is is one of the most common programming errors that people make and is typically caused

by forgetting the New keyword. For example, this code might look OK at first glance:

```
Dim car As Vehicle
car.Brand = "Ford"
car.Model = "Focus"
```

But the 2nd line will result in a NilObjectException because car is not actually an instance. It is just a variable that is declared to contain a Vehicle instance, but is currently empty (Nil).

If you are not sure if a variable contains an instance, you can check it before you use it:

```
If car <> Nil Then
    car.Brand = "Ford"
    car.Model = "Focus"
End If
```

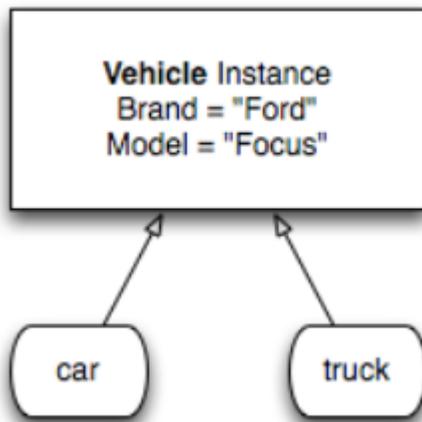
When the variable is not Nil, then it has a valid reference. And since it is a reference, it has important considerations when assigning the variable to another variable.

 When you do an assignment from one reference variable to another, the second variable points to the same reference as the first variable.

This means the second variable refers to the same instance of the class as the first variable and is not a copy of it. So if you change a property of the class with either variable, then the property is changed for both. An example might help:

```
Dim car As New Vehicle
car.Brand = "Ford"
car.Model = "Focus"
Dim truck As Vehicle
truck = car
' truck.Model is now "Focus"
car.Model = "Mustang"
' truck.Model is now also "Mustang"
truck.Model = "F-150"
' car.Model is now also "F-150"
```

In this diagram you can see that the variables for both car and truck point to the same instance of Vehicle. So changing either one effectively changes both.

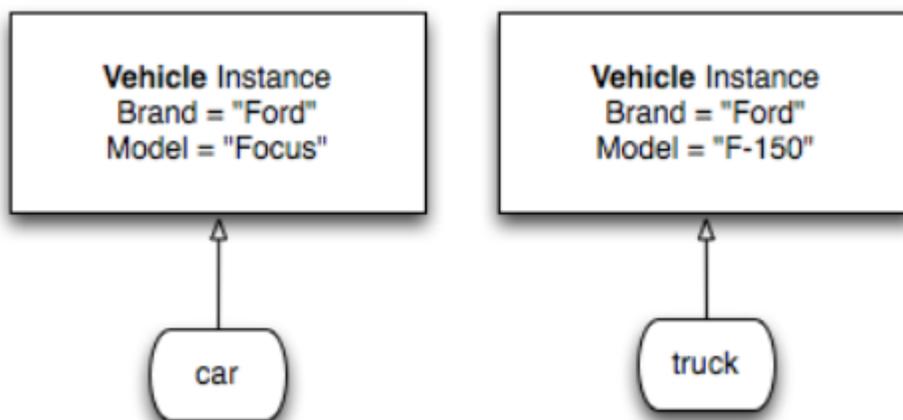


If you want to create a copy of a class, you need to instead create a new instance (using the New keyword) and then copy over its individual properties as shown below:

```

Dim car As New Vehicle
car.Brand = "Ford"
car.Model = "Focus"
Dim truck As New Vehicle
truck.Brand = car.Brand
truck.Model = car.Model
' truck.Model is now also "Focus"
truck.Model = "F-150"
' car.Model remains "Focus"
  
```

When you do it this way, you get two separate instances. Changes to one do not affect the other as you can see in



these diagrams:

## Adding Classes to Your Projects

Adding a class to a project is easy. To add a new class, click the Insert button on the toolbar and choose Class or select Class from the Insert menu. This adds a new class to the Navigator with the default name (Class1 for the first class).

Use the Inspector to change the name of the class. Like modules, classes primarily contain properties and methods.

But they can also contain many other things such as constants, enumerations, events and structures.

## Adding Properties to Classes

Properties are variables that belong to an entire class instance rather than just a single method. To add a property to a class, use the Add button on the Code Editor toolbar, Insert → Property from the menu, the contextual menu or the keyboard shortcut (Option-Command-P on OS X or Ctrl+Shift+P on Windows and Linux). You can set the property name, type, default value and scope using the Inspector.

To quickly create a property, you can enter both its name and type on one line in the Name field like this: `PropertyName As DataType`. When you leave the field, the type will be set in the Type field.

Properties added in this manner are sometimes called Instance Properties because they can only be used with an instance of the class. You can also add properties that can be accessed through the class itself without using an instance. These are called Shared Properties.

### Shared Properties

A shared property (sometimes called a Class Property) is like a “regular” property, except it belongs to the class, not an instance of the class. A shared property is global and can be accessed from anywhere its scope allows. In many ways, it works like a module property.

It is important to understand that if you change the value of a shared property, the change is available to every usage of the shared property.

Generally speaking, shared properties are an advanced feature that you only need in special cases. For example, if you are using an instance of a class to keep track of items (e.g., persons, merchandise, sales transactions, and so forth) you can use a shared property as a counter. Each time you create or destroy an instance of the class, you can increment the value of the shared property in its constructor and decrement it in its destructor. (For information about constructors and destructors, see the section Constructors and Destructors.) When you access it, it gives you the current number of instances of the class.

## Adding Methods to Classes

To add a method to a class, use the Add button on the Code Editor toolbar, Insert → Method from the menu, the contextual menu or the keyboard shortcut (Option-Command-M on OS X or Ctrl+Shift+M on Windows and Linux). You can set the method name, parameters, return type and scope using the Inspector.

When typing the method name, the field will autocomplete with the names of any methods on its super classes.

Methods added in this manner are called Instance Methods because they can only be used with an instance of the class.

You can also add methods that can be accessed through the class itself. These are called Shared Methods.

## Shared Methods

A shared method (sometimes called a Class Method) is like a normal method, except it belongs to the class, not an instance of the class. A shared method is global and can be called from anywhere its scope allows. In many ways, it works like a module method.

 Shared Methods do not know about an instance so its code can only access other shared methods or properties of the class.

Generally speaking, shared methods are an advanced feature that you only need in special cases. A common usage of shared methods is to create an instance (rather than relying on a Constructor).

# OOP Design Concepts

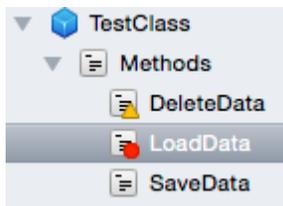
This section covers some of the important object-oriented design concepts that apply to classes: Encapsulation, Overloading, Inheritance and Polymorphism.

## Table of Contents

- [Encapsulation](#)
- [Overloading](#)
- [Inheritance](#)
- [Virtual Methods](#)
- [Polymorphism](#)

## Encapsulation

Encapsulation is the process of hiding information that does not need to be exposed to other parts of the code. With classes, you can control this using the scope of your properties and methods. In order from least visible to most visible, there are: Private, Protected and Public.



### Private

Setting the scope to private means that the item is only accessible from within the current class instance (or class for shared items). This makes it inaccessible to the rest of your code.

Private is the most restrictive setting. It is good coding practice to get into the habit of setting your items to Private if they are not needed outside the class instance.

### Protected

A scope of protected means the item is accessible from the current class instance (or the class itself for shared items) and its subclasses.

A Protected method, property, or constant is available only to other code within the class and subclasses based on the class. It

is inaccessible to the rest of your code. When other code in the class needs to access a Protected method, property, or constant, you simply reference it by name. If you try to access a Protected method, property, or constant outside of the class, you will get an error message indicating that the item is out of scope when you try to run your project.

### Public

A Public method, property, or constant is available to code throughout the project. You reference it using dot

notation using the class reference and the name, such as:

```
car.Model
```

## Overloading

Overloading is the term for methods that have the same name, but have different method signatures (such as a different number of parameters or different data types for the parameter).

 You cannot overload methods based on return values.

To overload a method in your own classes, simply add the new method to the class and give it different parameters. The method appears in the Navigator with the overloaded parameters shown below the method so that you can tell them apart.



To overload methods in built-in framework classes, you first need to create a subclass using inheritance.

You can also overload operators using a variety of specially implemented functions. A good example of a built-in overloaded operator is the “+” operator. If its arguments are numbers, it computes the sum; if the arguments are text, it concatenates the text. The Advanced Class Features section in this chapter has an example of this.

## Inheritance (Subclassing)

Inheritance a feature of object-oriented programming where you create a new class that is based on an existing class. The new class is called the subclass and the existing class is called the super class. The subclass "inherits" and can thus use all the public and protected properties and methods of its superclass.

Why is this useful? You may find situations where you would like to have an object that is a slightly altered version of one of the built-in classes. For example, you might want a version of the TextArea control that disables the Cut and Copy items on the Edit menu, preventing the user from putting sensitive data on the Clipboard. You might want to create a List Box that, by default, has the months of the year in it. You can create your own versions of these built-in classes by creating subclasses that you add to your project.

There's an important difference between adding an instance of TextField to a window versus adding a subclass based on TextField to your project. In the latter case, you can customize the subclass based on TextField itself and use instances of the customized subclass in several places in the application. You can also save the customized subclass so that you can reuse it in other projects.

### What is a Subclass?

A subclass is simply a class that has a super class. A super class is a class the subclass is based on, also sometimes

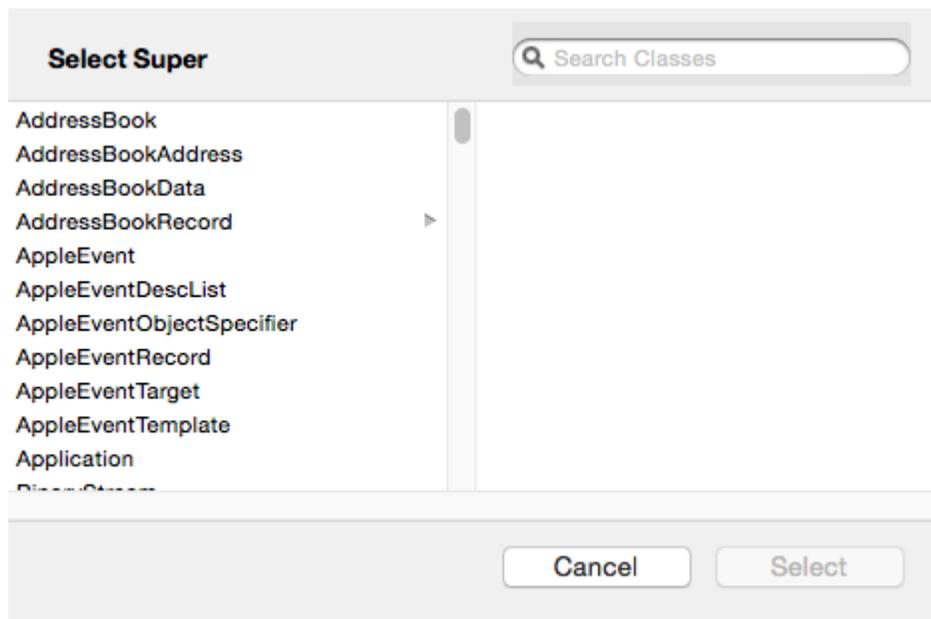
called the parent class. Subclasses inherit all of their super's (public or protected) properties, methods and constants. The subclass can also implement any event handlers that were not implemented on the super class or it can implement event definitions on the super class.

In fact, a subclass works identically to its super class until you start modifying it. After that, it's different from its super class only in the ways you make it different by adding properties, modifying events, or adding or modifying methods.

## Creating a Subclass from an Existing Class

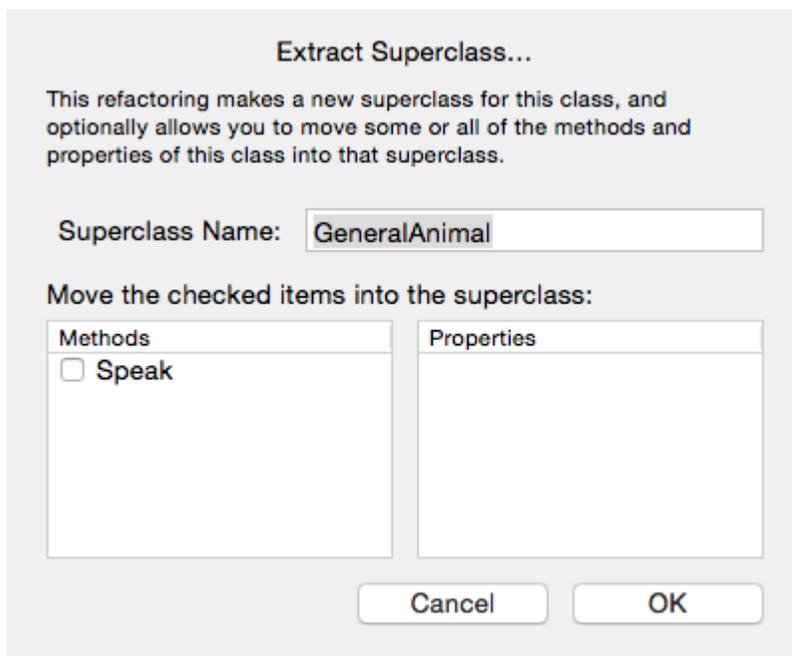
There are a several ways to create a subclass from an existing class in your project or from an existing framework class.

1. If want to subclass a built-in control class, you can simply drag the control from the Library to the Navigator.
2. You can select a class in the Navigator and use the contextual menu option "New Subclass" to create a new subclass based on the selected class.
3. You can add a class to your project (using the Insert button on the toolbar or the Insert menu) and then manually change its Super property to the name of the class that it is based on. You can just type the name of the superclass in the Super field or you can click the small "pencil" button icon next to the field to display the Select Super Class window where you can choose a class.



## Creating a Superclass from an Existing Class

If you have an existing class that you have determined should really be a subclass, you can create a super class from it. Use the contextual menu for the class and select "Extract Superclass". This opens a window that allows you to specify the name of the super class and to choose the method and properties to move from the class to the super class.



## Virtual Methods (Overriding)

Virtual methods provide a way for a subclass to override its own version of a method that its super class has. Ordinarily, a subclass inherits the methods belonging to its parent.

When a subclass has a method that has the same name (and parameters) as a method on the super class, the subclass method is instead called. This is referred to as “overriding”. You can still call the method on the superclass by calling it specifically using the Super prefix with the method call:

```
Super.MethodName
```

## Polymorphism

Polymorphism is the ability to have completely different data types (typically classes) behave in a uniform manner. You can do this using inheritance and overriding, inheritance and events or class interfaces (described later in this chapter).

A common example of this is the Animal, Cat and Dog relationship. If you want to have a generic animal object that knows how to speak “Meow!” or “Woof!” depending on whether it contains a Cat or Dog, you could implement it using inheritance and overloading.

First, create a new class called Animal. Add to it a Speak method that returns Text. Now create a subclass of Animal, called Cat. Add to it a Speak method that returns Text. This means you are overriding the unimplemented Speak method on Animal. The code for this method is:

```
Return "Meow!"
```

Next, create another subclass of Animal, called Dog. Add to it a Speak method that returns a String with this code:

Return "Woof!"

To test this, create add a button to a layout. In the Action event of the button add code to create an array of Animals and then to loop through the array calling the Speak method:

```
Dim animals() As Animal
animals.Append(New Cat)
animals.Append(New Dog)
For Each a As Animal In animals
    MsgBox(a.Speak)
Next
```

When you run this code, you will see "Meow!" and then "Woof!". Because Cat and Dog are both subclasses of Animal, they are allowed to be assigned to a variable (the animals array) with a type of Animal.

And when you call the Speak method of Animal (in the loop), because of polymorphism, your code calls the Speak method of the actual subclass type that was added to the array. Later sections will also show you how you can accomplish the same thing using inheritance with events and with class interfaces.

# Properties, Methods and Events

## Table of Contents

- [Properties](#)
- [Methods](#)
- [Events](#)

## Properties

Properties are a trait that tells you something about an object. You use them as a way for class instances to store or calculate values. You can add four types of properties to classes: properties, computed properties, shared properties and shared computed properties.

You can add properties to a subclass to store values that its super class doesn't store. For example, you might want to create a subclass of the TextField control that stores the last value the user entered. This would allow you to selectively reject the current entry and restore the last entry.

### Properties

Properties are variables that belong to an entire class instance rather than just a single method. To add a property to a class, use the Add button on the Code Editor toolbar, Insert → Property from the menu, the contextual menu or the keyboard shortcut (Option-Command-P on OS X or Ctrl+Shift+P on Windows and Linux). You can set the property name, type, default value and scope using the Inspector.

To quickly create a property, you can enter both its name and type on one line in the Name field like this: `PropertyName As DataType`. When you leave the field, the type will be set in the Type field.

Properties added in this manner are sometimes called Instance Properties because they can only be used with an instance of the class.

You can also add computed properties to a class. These are properties whose values are calculated rather than stored.

### Shared Properties

A shared property (sometimes called a Class Property) is like a "regular" property, except it belongs to the class, not an instance of the class. A shared property is global and can be accessed from anywhere its scope allows. In many ways, it works like a module property.

It is important to understand that if you change the value of a shared property, the change is available to every usage of the shared property.

Generally speaking, shared properties are an advanced feature that you only need in special cases. For example, if you are using an instance of a class to keep track of items (e.g., persons, merchandise, sales transactions, and so forth) you can use a shared property as a counter. Each time you create or destroy an instance of the class, you can increment the value of the shared property in its constructor and decrement it in its destructor. (For information

about constructors and destructors, see the section Constructors and Destructors.) When you access it, it gives you the current number of instances of the class.

Shared Computed Properties work exactly like you expect. They are shared versions of computed properties.

## Methods

Methods provide functionality for your classes. They usually perform some action, such as loading data or calculating values.

Chapter 3, Section 8 discusses methods in more detail.

### Methods

Methods (Instance Methods) were described in Section 1. They are methods that belong to a class instance.

### Shared Methods

As discussed in Section 1, Shared Methods (also called class methods) are methods that belong to the class rather than an instance.

## Events

Events are a type of method that is called by some action (or event) that has occurred. These events have nothing to do with the “events” in “event-driven programming”. Many classes and controls have their own built-in events (which are covered in the User Interface and Framework books), but you can also create your own events using Event Definitions.

Event Definitions gives you a way to add your own Event Handlers to your classes. Event Definitions can be called only from the class itself, but they can be implemented only by its subclasses. To add an event definition to a class, use the Add button on the Code Editor toolbar, Insert → Event Definition from the menu or the contextual menu.

For example, if you have a Save method on a class you may want to give subclasses a way to do processing before and after the save. One way you could do this is by overriding the Save method on the subclass like this:

```
Sub Save
  PreSave
  Super.Save
  PostSave
End Sub
```

This works but it relies on you defining and implementing everything properly. If you do this with events, then there is no room for error. With events, you create two event definitions on the super class: PreSave and PostSave.

You would create two event definitions on the super class: PreSave and PostSave. In the Save method on the super class, you then call PreSave at the beginning and PostSave at the end. When you create a subclass, you will see that it has two Event Handlers you can add to it: PreSave and PostSave. Simply add those event handlers and implement

them as needed.

The built-in control classes all contain a wide variety of event handlers (such as Open), all of which work in this manner.



Note that when you implement an event handler, it no longer appears in subclasses. If you want the event handler to still be available to additional subclasses, you need to create a new event definition in the subclass (matching the name and parameters) and then call it in from event handler you implemented.

## Animal Example

Looking back at the Animal example, this is how you would do it using events.

Create a new class called Animal. Add to it a Speak method that returns Text with this code:

```
Return SpeakSound
```

Now add to the Animal class an Event Definition called SpeakSound and set its return value to Text.

Next, create a subclass of Animal, called Cat. Click on the button “+” button on the toolbar and select “Add Event Handler”. In the dialog, you will see the SpeakSound event handler. Select and and press OK. The code for this event handler is:

```
Return "Meow!"
```

Create another subclass of Animal, called Dog, and also add the SpeakSound event handler with this code:

```
Return "Woof!"
```

To test this, create a button on a window. In the Action event of the button add code to create an array of Animals:

```
Dim animals() As Animal
animals.Append(New Cat)
animals.Append(New Dog)
```

```
For Each a As Animal In animals
    MsgBox(a.Speak)
Next
```

When you run this code, you will see “Meow!” and then “Woof!”.

Because Cat and Dog are both subclasses of Animal, they are allowed to be assigned to a variable (the animals array) with a type of Animal. When you call the Speak method of Animal (in the loop), it in turn calls the SpeakSound event handler of each subclass that was added to the array.

# Constructors and Destructors

When you create a new object, you will sometimes want to perform some sort of initialization on the object. The constructor is a mechanism for doing this. An object's constructor is the method that is executed automatically when an instance of the class is created. Conversely, when an object is removed from memory, its destructor is called.

## Constructors

You add a constructor to a class by adding a method to the class called Constructor. For convenience, "Constructor" is a choice in the Method Name popup menu in the Code Editor.

The constructor will be called automatically when an instance is created via the New operator. If your constructor accepts parameters, you must pass the required number of parameters and they must be of the correct data type.

Whenever you create a constructor for a subclass, the Code Editor automatically adds a call to the super class's constructor for you in the Code Editor and adds comments that explains its purpose.

```
' Calling the overridden superclass constructor.
Super.Constructor
```

This is added because the constructor that you are writing overrides its Super class's constructor, but the new object may not be initialized correctly unless its Super class's constructor executes. The Super method is how you call a method of a Super class that is being overridden by a method of the subclass.

```
*** This needs to change since Xojo.Core.Date is immutable ***
```

Here is a simple example of a constructor. Suppose you are managing a service that sells monthly subscriptions. You want a custom version of the Date class that automatically takes the value of the expiration date when an instance is instantiated. First, add a new class to the project and set its Super class to Xojo.Core.Date and rename it "ExpirationDate". Now you can add a method called "Constructor". The constructor should take one integer parameter, numMonths, which is the number of months the customer has signed up for. So the constructor has only one line of code:

```
Self.Month = Self.Month + numMonths
```

When an instance of a "regular" Date object is created, it is initialized to the current date, so this line increments the current month number by the number of months that is passed in as a parameter.

When you need to get the expiration date for a customer who signs up for 6 months, you can get it like this:

```
Dim expDate As New ExpirationDate(6)
MsgBox(expDate.ShortDate)
```

The number months is passed in as a parameter and the new instance holds the expiration date instead of the current date.

## Initializing Control Subclasses

If you need to initialize an instance of a control subclass, don't use the constructor. Instead, put the code that does the initialization in the Open event handler for the control.

When you access Inspector properties in the Constructor, they will not have the default value specified in the Inspector and will instead have the default value for the data type. If you assign values to these public properties, the values will get overridden with what is specified in the Inspector.

## Destructors

You can also create a destructor, although they are not usually necessary. The destructor is called automatically when an instance of the parent class is deleted or goes out of scope — for example, when the user closes the window. Name the new method “Destructor”. Destructors take no parameters and do not return a value.

Destructors are called when the last reference to an object is removed, even if execution is in a destructor for another object. Note that this means you can cause a stack overflow if your destructor triggers other destructors in a deep recursion. However, such overflow will not happen as long as properties of the object are being cleaned up automatically. So, it is generally preferable to not set properties to Nil in your destructor, but instead allow them to be automatically cleaned up for you.

## Memory Management

Memory is managed for you automatically using something called reference counting. A counter is used for each instance (object) that you create and it records references to the object. Each additional reference increments the counter; removing a reference to the object decrements the counter. You remove a reference, for example, by setting it to Nil, by letting it go out of scope such as when a method containing the instance returns to its caller. When the reference counter reaches zero, the object is removed from memory immediately.

This means that instances of classes are removed from memory automatically when they are no longer used. Suppose you create a class based on a ListBox. You then create an instance of that class in a window. When the window is opened, the instance of the class is created in memory automatically. When the window is closed, the instance of the class is automatically removed from memory. If you store the reference to a class in a local variable, when the method or event handler is finished executing, the instance of the class is removed from memory. If you store a reference to an instance of a class in a property, the instance will be removed from memory when the object owning the property is removed from memory.

# Interfaces

A class interface is a construct that you can use to tie together classes that do not share a super class but have something in common in your application. Class interfaces are used to specify what an object does without specifying how it does it.

In order to understand class interfaces better, it's helpful to think of a class as consisting of two components, the (public) interface to the class and the implementation. The interface consists of the class's Public method calls and the implementation is the code that implements the methods. The interface says what the class does and the implementation says how it does it.

In the object hierarchy, a subclass inherits both the interface and the implementation from its super class. That is, it gets both the method calls and the specific implementation of the method.

Class interfaces enable you to separate the two constructs. If two or more classes need to do the same thing but do it in different ways, you use an interface instead of a super class.

A class interface operates as a "spec" that contains a list of methods that custom classes in your project use. It does not actually contain any code for the methods themselves.

The methods in the class interface are placeholders for methods that are actually contained in each custom class that "implements" the class interface. Also, a custom class can implement more than one class interface.

The term "implement" simply means that the class has methods of the same names and declarations that are found in the class interface. The class interface specifies the methods and their declarations but not the code.

Several class interfaces are built-in to the framework. You can implement any of these in your classes or add and implement your own. For example, the Readable and Writeable interfaces specify methods for reading and writing data. Each class that uses the Readable or Writeable interfaces supplies the implementations. For example, a class that reads data from the Serial port would use a different implementation than a class that reads data from a binary file.

When you specify that a class implements a class interface, the class must implement all the methods in the class interface and the method declarations must match. However, the classes are free to implement the methods in different ways. For example, a method that changes the font in a Label would be implemented in a different way than a method that changes the font in a Canvas control that displays text via calls to the Graphics class.

The process involves three basic phases:

- Creating the class interface
- Creating the classes that implement the class interface
- Adding the classes to your project and calling the class interface methods in your program. Typically, that means writing generic code that tests whether a class implements a class interface and executing class interface methods where appropriate

To add a new class interface, click the Insert button on the toolbar and choose Class Interface or select Class Interface from the Insert menu. This adds a new class interface to the Navigator with the default name (Interface1 for the first class interface).

Use the Inspector to change the name of the class interface.

You can only add methods to class interfaces. To add a method to a class interface, use the Add button on the Code Editor toolbar, Insert → Method from the menu, the contextual menu or the keyboard shortcut (Option-Command-M on OS X or Ctrl+Shift+M on Windows and Linux). You cannot specify any code in the class interface, so the Code Editor is disabled. Use the Inspector to specify method names and parameters.

## Implementing the Interface Methods in a Class

To implement the methods of the class interfaces, you need to assign the class interface to a class. To do so, select the class in the Navigator and in the Inspector select the Choose button next to the Interfaces label. You can also use the “Implement Interface” option in the contextual menu of the method in the Navigator.

When you click Choose, the Choose Interfaces dialog displays showing the names of all the class interfaces available to you. Some of the interfaces are ones that are built-in and others are ones you created yourself. Select one or more interfaces to assign to the class.

You can also select the “Include #pragma error” in the source of each method” to force a compiler error to be generated with the message “Don't forget to implement this method!”. Remove the pragma after you have added code to implement the interface method.

Press OK to have the Code Editor automatically adds the methods of the interface to your class with a comment telling you the interface to which it belongs (along with the optional pragma).

If you modify or change the class interface after you have already applied it to a class, you will need to manually update the class.

If you attempt to compile a project that contains a class with an interface, but the class does not implement all the interface methods, you get a compile error: “This class is missing one or more methods of an interface it implements.”

### Modifying and Deleting Interfaces

If you change your mind and want to delete or replace an interface, you do so the same way you added the interface. When the Interface dialog appears, uncheck the interfaces you no longer want.

Note: Any methods that were added to the class while the interface was implemented are not modified or deleted when you remove the interface that they belonged to. That is, if you add an interface via the Implement Interfaces dialog, the methods that are specified by that interface remain as part of the class even if you delete the interface itself. You must take care of any “clean up” activities.

### Specifying the Interface Being Implemented

In rare cases you may find that a class implements more than one method with the same name but from different class interfaces. You may have decided that a method implements two interfaces that both have a method that shares the same name and declaration. Normally, there is no way to tell them apart. However, there is a way to specify the class interface to which each method belongs. You can do so using the optional Interfaces field in the method declaration dialog box, available using the “Show Implements” contextual menu of the method.

To use it, suppose you have two class interfaces, Foo and Bar, that both contain a method named “wahoo” that has no parameters. You can have a class called (Baz) that implements both methods separately.

Add the wahoo method to Baz. Then right-click (control+click on OS X) on the method name and select “Show Implements”. This displays an additional field in the Inspector called “Implements”. Here you can specify the name of the interface and method that this method actually implements: Foo.wahoo.

Now add another method called wahoo and select “Show Implements” for it. Specify “Bar.wahoo”. Now you have told the class exactly which interface methods it implements.

## Creating a new Class Interface from an Existing Class

If an existing class in your project contains methods that you want to extract as a class interface, you can generate the new class interface from the Navigator. To do so, select “Extract Interface” from the contextual menu of the method in the Navigator.

The new class interface is added to the project and the current class is made an implementor of the new interface.

## Polymorphism

Polymorphism is the ability to have completely different data types (typically classes) behave in a uniform manner. This can be done using class interfaces.

Here is how to implement the Animal, Dog and Cat example using class interfaces.

First, create a new class interface (called Animal). Add to it a Speak method that returns Text.

Now create a new class (Cat) and select Animal as its interface. The Code Editor automatically adds the Speak method for you. Add this code to it:

```
Return "Meow!"
```

Add another new class (Dog) and select Animal as its interface. The Code Editor adds the Speak method where you can put this code:

```
Return "Woof!"
```

To test this, create a button on a window. In the Action event of the button add code to create an array of Animals:

```
Dim animals() As Animal  
animals.Append(New Cat)  
animals.Append(New Dog)
```

```
For Each a As Animal In animals  
    MsgBox(a.Speak)  
Next
```

When you run this code, you will see “Meow!” and then “Woof!”.

Because Cat and Dog both implement Animal, they are allowed to be assigned to a variable (the animals array) with a type of Animal. And when you call the Speak method of Animal (in the loop), because of polymorphism, your code calls the Speak method of actual type that was added to the array.

# Subclassing Examples

## Example SelectablePopupMenu

As an example, you can create a `PopupMenu` subclass that adds a method that can be used to select an existing item in the `PopupMenu`.

Create a new class, called `SelectablePopupMenu`, and set its `Super` to `PopupMenu`. On this class, add a new public method and call it `SelectRow` with the parameter `value As String`.

This method will loop through all the rows in the `PopupMenu` and check to see if the text for a row matches the supplied value. If the text matches, it will select the row and return. If no matches are found, then nothing is selected.

This is the code for the `SelectRow` method:

```
For i As Integer = 0 To Self.ListCount-1
  If Self.List(i) = value Then
    ' Found a row with matching
    ' text, so select it
    Self.ListIndex = i
    Return
  End If
Next

' Select nothing
Self.ListIndex = -1
```

## Example: MonthPopup

Here is another example. Suppose you want to create a `PopupMenu` that, by default, displays the names of the months of the year, with the current month selected. You create a new class, call it `MonthPopup`, and choose `PopupMenu` as its super class.

Now you can add an `Open` event handler to `MonthPopup` and add the month names, the heading, and code that selects the appropriate month in the list. In this case, the `Open` event handler is:

```
Self.HasHeading = True
Self.InitialValue = "Months"
Self.AddRow("January")
Self.AddRow("February")
Self.AddRow("March")
Self.AddRow("April")
Self.AddRow("May")
Self.AddRow("June")
Self.AddRow("July")
Self.AddRow("August")
```

```
Self.AddRow("September")  
Self.AddRow("October")  
Self.AddRow("November")  
Self.AddRow("December")
```

```
// Select the current month  
Dim d As New Date  
Self.ListIndex = d.Month-1)
```

To add an instance of the MonthPopup class to a window, switch to the window's Window Editor and then drag MonthPopup from the Navigator to the Window Editor. When you run the application, the Open event handler will run and populate the PopupMenu with the months of the year, selecting the current month.

Note: Once created, custom control classes can be exported as self-contained objects that can be used in other projects. Just right-click (Control-click on OS X) on the custom control class in the Navigator and choose Export... from the contextual menu.

## Subclasses are Classes

Remember, subclasses are classes. They are called subclasses to differentiate them from the original classes and emphasize the point that they inherit the properties, events, and methods of their parent class. Because subclasses are classes, they can be the super class to other subclasses. For example, suppose you had a NumbersOnlyTextField subclass, but now you need a TextField that allows only numbers within a certain range. You could duplicate the NumbersOnlyTextField subclass and then modify its code.

However, this would make your project larger and more difficult to maintain. If you found a bug in the code of the NumbersOnlyTextField, you would have to remember that you used that code in other places as well, track them down, and fix them. A more efficient way is to create a new subclass and choose the NumbersOnlyTextField as its super class. The new subclass (let's call it "NumberRangeTextField") would utilize all of the properties, events, and methods of its super class. However, you can add code to the TextChange event handler that allows only numbers within a specific range.

# Advanced Features

This section covers more advanced features of classes. You will not need these features often, but they are definitely useful.

## Table of Contents

- Operator Overloading

## Assigning a Value to a Method

Sometimes it is helpful, from a syntax standpoint, to have a method that gets a value using an assignment operator.

Rather than doing this:

```
order.Quantity(2)
```

You would write:

```
order.Quantity = 2
```

which looks very much like a property. You can get the syntax behavior of a property using a method by using the `Assigns` keyword in the parameter list.

The declaration looks like this:

```
Sub Quantity(Assigns amount As Integer)
```

You can also have more than one parameter, something that you can not do with a simple property:

```
Sub Quantity(productName As String, Assigns amount As Integer)
```

This command would be used like this:

```
order.Quantity("Book") = 2
```

When you use this technique, it will look like you are setting a property, even though you are really dealing with a method. This can be handy when designing classes and can make your code easier to read.

## Operator Overloading

You can easily overload methods on a class, but how do you overload an operator on a class? For example, what if you have two instances of a `SalesOrder` class and want to tell if one is greater than the other?

You do this by implementing one of the Operator overload methods (Figure 5.10) on the `SalesOrder` class, in this case `Operator_Compare`.

```
Sub Operator_Compare(rightOrder As SalesOrder)
    If Self.TotalAmount > rightOrder.TotalAmount Then
        Return 1
    ElseIf Self.TotalAmount < rightOrder.TotalAmount Then
        Return -1
    End If
End Sub
```

```

Else
    Return 0
End If
End Sub

```

These are the operator overload methods:

Operator	Functions
+	Operator_Add, Operator_AddRight
-	Operator_Subtract, Operator_SubtractRight
*	Operator_Multiply, Operator_MultiplyRight
/	Operator_Divide, Operator_DivideRight
\	Operator_IntegerDivide, Operator_IntegerDivideRight
Mod	Operator_Modulo, Operator_ModuloRight
And	Operator_And, Operator_AndRight
Or	Operator_Or, Operator_OrRight
Not	Operator_Not
=, <, >, <=, >=	Operator_Compare
(lookup)	Operator_Lookup
(negation)	Operator_Negate
(convert)	Operator_Convert
Subscript	Operator_Subscript
Redim	Operator_Redim

For more information about operator overloading for custom classes, refer to [Operator Overloading](#) in the Reference Guide.

## Operator Lookup

Dot notation is used to access methods and properties of a class instance. If you try to access a method or property that does not exist, you get a compilation error.

Operator Lookup is a special class method that is called when anything following the "." that is not an actual method or property of the class is used. One use for this feature is to look up a value that may not be known until run-time. Suppose you are creating a Preferences class that will be used to save your application preferences. Normally, you would have to create a property for each preference you want to save. If you find you have a new preference value, you'll need to go back to the class and add the property before you can use it.

But you could use operator lookup instead.

With operator lookup, you use the method to check what was typed after the "." and have that be the name of the preference. The value that is assigned can then be stored (perhaps using a Dictionary).

Your code to save a value might look like this:

```
Sub Operator_Lookup(name As String, Assigns value As String)
  If mPreferenceDict = Nil Then
    mPreferenceDict = New Dictionary
  End If
  mPreferenceDict.Value(name) = value
End Sub
```

And the code to get a value might look like this:

```
Function Operator_Lookup(name As String) As String
  If mPreferenceDict = Nil Then Return ""

  If mPreferenceDict.HasKey(name) Then
    Return mPreferenceDict.Value(name)
  End If
End Sub
```

With this in place, you can now write code to save preference values even though you have never defined specific properties or methods for the preference. In the App class, add a property called Prefs as Preferences. In the App.Open event, initialize this property:

```
Prefs = New Preferences
```

Now, anywhere in your project you can write code like this to store a preference value:

```
App.Prefs.UserName = "Bob Roberts"
App.Prefs.UserEmail = "bob@gmail.com"
And code like this to get a preference value:
Dim userName As String
userName = App.Prefs.UserName ' Not an actual property, but calls Operator_Lookup
```

## Attributes

Attributes are compile-time properties. They can be added to project items and code items such as method and properties. An attribute consists of its Name and its Value. The Name is required, but the value is optional.

Attributes are added using the advanced tab of the Inspector. Attributes can be added to classes, modules, windows, containers, interfaces and toolbars. A subclass inherits attributes from its super class. These inherited values can be overridden by the subclass by simply redefining them.

## Creating an Attribute

You create an attribute using the Attribute Editor in the advanced tab of the Inspector for the item. Use the “+” or “-” buttons to add or remove attributes. Specify the Name and Value for the attribute in the list.

There are two ways to specify the value. If it is a literal value, such as “ID”, then the value must be enclosed in quotes. If it is a constant, you can just use the name of the constant, for example kID. If you forget the quotes for a literal value, you will get a compiler error if the constant is not found.

## Accessing an Attribute

Attributes are accessed in your code using Introspection. The `AttributeInfo` class is used to fetch the Name-Value attribute pairs for a particular object.

This code gets the attribute values of the default window and displays them in a List Box:

```
Dim myAttributes() As Introspection.AttributeInfo =  
Introspection.GetType(Window1).GetAttributes  
  
For i As Integer = 0 To UBound(myAttributes)  
    ListBox1.AddRow(myAttributes(i).Name)  
    If myAttributes(i).Value.IsNull Then  
        ListBox1.Cell(ListBox1.LastIndex, 1) = "No Value"  
    Else  
        ListBox1.Cell(ListBox1.LastIndex, 1) = Str(myAttributes(i).Value)  
    End If  
Next
```

## Using Deprecated and Hidden Attributes

The `Deprecated` and `Hidden` attributes are reserved and perform special actions when used on a project item (such as a class or module) or a method, property (or constant, etc.) that you can add to a project item.

The `Deprecated` attribute allows you to indicate that an item “has a replacement”. Set the attribute value to the name of the class/method/property that should be used instead (be sure to enclose the name with quotes).

A deprecated item appears in the Errors Pane when you use Analyze Project.

The `Hidden` attribute hides the specified item from introspection, the debugger and auto-complete. You do not need to specify a value.

Both of these attributes may be useful on frameworks that are used by other developers.

## Casting

An extremely powerful way of creating generic, reusable code is to take advantage of the ability to convert an instance of a class to an instance of its subclass. The idea is best illustrated by an example. Since the objects you create are subclasses of base classes, you can always test to see whether an object is a member of a specific

subclass. The IsA operator does this.

With the IsA operator, you test whether an object is of a specific subclass and, if it is, cast it as that type to do something specific with it. You cast an object by using the classname as a function that operates on the instance.

When you cast an instance, compile-time error-checking cannot guarantee that you are casting to a legal object type. The instance that you are casting has to be of the type that you specify. Casting just tells your code to treat the object as a instance of the class to which it is cast. It doesn't convert it from one class to another. If you cast incorrectly, you will get an `IllegalCastException` at run-time.

Here is an example that uses a For loop to cycle through all the controls in a window. It checks each control to see if it is a Label and if it is, it casts the control and displays its text:

```
Dim labelText As String
' Loop through controls in window
For i As Integer = 0 To Self.ControlCount-1
  If Self.Control(i) IsA Label Then
    ' Cast the control to a Label
    ' and gets its Text property
    labelText = Label(control(i)).Text
    MsgBox(labelText)
  End If
Next
```

As you can see, the code does not refer to any specific windows, Labels or anything else. Therefore, you can write this routine once and use it in any window in any project.

## Static Variables

A Static variable is a global variable that has the scope of a local variable.

It is declared in a method and, unlike with a normal local variable, the value is retained the next time the method in which it is declared is called. This value is retained for the method in all instances of the class, whether they already exist or are created later.

You declare a Static variable by using the Static keyword in place of Dim:

```
Static myValue As Integer
```

The most common use of a Static variable is when you need set a value using a method that might take a noticeable time to execute. By using a Static variable, you can ensure the initialization is only done once, but the value is available for re-use each time the method containing it is called.

# Overview

Every project has an App object that is automatically added to it when the project is created. This App object gives you access to events and properties that are specific to the app type. If you look at the Inspector for the App object, you'll see that it has a super:

iOS	<a href="#">iOSApplication</a>
Desktop	<a href="#">Application</a>
Web	<a href="#">WebApplication</a>
Console	<a href="#">ConsoleApplication</a>

The next sections cover the specifics for each app type.

## Accessing the App object

You can rename the App object. If you wish, you can even subclass App. When you do this, your new subclass gets the properties for the default window, menu bar and icon.

You can set the scope of properties, methods and constants of the App subclass using the same rules as for any class.

The global App function gives you a reference to the App class in your project so that you can access its public properties, methods and constants. If you have subclassed App, the then App function gives you a reference to your subclass instead.

To access a public property somewhere in your project, you would write it like this:

```
value = App.MyProperty</code>
```

 You use the App method to get a reference to the App object even if you have renamed or subclassed the App object.

iOS Applications runs on iOS devices such as iPhones and iPads.

## Table of Contents

- [Project Overview](#)
- [App Object](#)
- [Screens](#)
- [Build Settings](#)

## iOS Project Overview

When you create your first iOS project, you get the following project items added automatically:

- **App:** The App object works similarly to how it works for desktop and web projects. In the App Inspector, you can set the Screen to use for iPhone and iPad-sized devices. Leave a device screen blank if you do not want it to have a native UI for the device. For example, to prevent an app from working on iPhone, do not assign an iPhone Screen. If you do not want the app to run natively on an iPad, leave off an iPad Screen. The app will still

run on the iPad, but will do so by running the iPhone app in the OS "scaled" mode.

- **iPhoneScreen:** Specifies the initial screen displayed when the app is started on iPhone-sized devices.
- **iPadScreen:** Specifies the initial screen displayed when the app is started on iPad-sized devices.
- **View1:** This the main layout where you place controls. It is roughly equivalent to a Window or WebPage in desktop and web projects.
- **App Icon:** This item is where you place the various size app icons for the app. You can use the ImageMaker utility (in the Extras/iOS Utilities folder of the Xojo installation) to create the image files and then drag them into the various sized areas.
- **Launch Images:** This item contains the various size launch images for the app. Launch images are used kind of like a "splash screen" while the app is actually launching. Typically these are screen shots of your empty View so that it makes it look like your app is starting quickly, but they can be almost anything. There are a variety of sizes for all the devices. In particular, the iPhone 5 size is required in order for an iOS app to fill the screen on iPhone 5 and larger devices.

## App Object

The App object work similarly to how it works for desktop and web projects. For an iOS project, the App object is a subclass of `iOSApplication`. You use it to specify the default screen layouts for the devices. You can add events, properties and methods to the App object. Use the App prefix to refer to public properties and methods elsewhere in your code. For example, to refer to the property `UserName` added to the App object in a class of the project, you would write:

```
App.UserName = "Billy"</span>
```

## Event Handlers

The Application class for an iOS application has event handlers. They are:

- **LowMemoryWarning:** Called when iOS is running low on memory.
- **Open:** Called when your application first starts.
- **UnhandledException:** Called when an exception was raised in any of your code and the exception was not handled by its own `Try...Catch` block.

## Properties

- **Default iPhone Screen:** The Screen to display when an iPhone-sized device is used.
- **Default iPad Screen:** The Screen to display when an iPad-sized device is used.

## Usage

The LowMemoryWarning event handler gets called when iOS is running low on memory. Since iOS does not use virtual memory, it has to purge apps from memory when the OS detects memory is running low. Implement this event handler if your app reserves a lot of memory (giant arrays, for example) so that you can release the memory. If you do not release the memory quick enough, iOS will just terminate the app.

The Open event handler is called when the app is launched (or relaunched if it was removed from memory). It is not called if your app remains in memory and the user switches back to it from another app. The Open event handler is a great place for app initialization code.

The UnhandledException event handler is called when an exception was raised in any of your code and the exception was not handled by its own `Try...Catch` block. Return True to allow your app to continue, essentially ignoring the exception (this may cause instability depending on the cause of the exception). This event is commonly used to log error information or maybe save state before the app quits. You might find it useful to put a Break statement in this event handler so that the debugger is displayed when an unhandled exception occurs, along with code to save the stack to an array so you can review it in the debugger:

```
Dim</span> stack() As</span> StackFrame = exc.CallStack  
Break
```

## Screens

In the Inspector for App, you can specify the default screens to use when the app is run on an iPhone or an iPad. A Screen is a project item that specifies the type of layout and supported orientations. Screen1 is added to your project by default. When you click on the Screen you are shown a preview of how it might look on an iPhone in portrait mode. Refer to [Screens](#) for more information.

## Build Settings

The Build Settings section of the Navigator contains the build-specific settings for your application. You can check the box next to each target in order to build an application for that target.

## Shared

The Inspector for Shared settings contains these properties:

- **Major Version** (*MajorVersion*): The Major version for your app. Version numbers are usually written as 1.2.3.4, where “1” is the major version.
- **Minor Version** (*MinorVersion*): The Minor version for your app. Version numbers are usually written as 1.2.3.4, where “2” is the minor version.
- **Bug Version** (*BugVersion*): The Bug version for your app. Version numbers are usually written as 1.2.3.4, where “3” is the bug version.
- **Stage Code** (*StageCode*): Used to indicate the type of app you are building (Development, Alpha, Beta, Final).
- **Non Release Version** (*NonReleaseVersion*): The Non Release (build) version for your app. Version numbers are usually written as 1.2.3.4, where “4” is the non release version.
- **Auto Increment Version Info**: When ON, the Non Release Version is increased by one each time you do a Build (but not when you Run).
- **Short Version** (*ShortVersion*): A short text description for your app. Usually this contains just the version number (such as 1.2.3.4) and is displayed by some operating systems in Get Info or Property windows for the app.
- **Long Version** (*LongVersion*): A longer text description for your app. Usually this contains the app name, copyright, version and other information. This is displayed by some operating systems in Get Info or Property windows for the app.
- **Package Info** (*PackageInfo*): A text description for your app that may be displayed by some operating systems in Get Info or Property windows for the app.
- **Use Builds Folder**: When ON, a separate folder is placed alongside the project file. Each platform that is built (OS X, Windows, Linux) also gets its own subfolder within this build folder.
- **Include Function Names**: When ON, the actual names of your function calls are embedded in the built app. This is useful for debugging purposes and for getting stack traces.
- **Language**: Allows you to specify the language to use to resolve dynamic constants.
- **Simple Reference**: When ON, you do not have to use namespace prefixes with any of the new framework classes used by iOS projects.
- **Command Line Arguments**: These are the command-line arguments that are passed to your app when you run it in Debug mode.
- **Destination**: Specifies the path where the app is located when you run it in Debug mode. If not specified, then the app is placed alongside your project file (or in a folder alongside the project file on Windows and Linux).

## iOS

Contains iOS-specific settings, including:

<b>ID</b>	
iOS App Name	My iOS App
<b>Build</b>	
Bundle Identifier	com.example.myapp
<b>Code Signing</b>	
Team	None
Build For App Store	<input type="radio"/> OFF
<b>iOS Debugging</b>	
Simulator Device	iPhone 4s (iOS 8.4)
Architecture	32 Bit

- **iOS App Name:** The actual name for your iOS app. This is the name that appears on the Springboard in iOS.
- **Bundle Identifier:** The bundle identifier is used by iOS as a unique descriptor for your app. It is usually specified as a reverse domain name, such as com.xojo.myapp. A bundle identifier is required for iOS apps.
- **Team:** This is the Team to use for certificates when code signing the iOS app. You have to code sign the app before it can be installed to a device or submitted to the App Store.
- **Build for App Store:** When ON, this creates a signed package file that can be submitted to the App Store using Xcode's Application Loader.
- **Simulator Device:** When debugging, this specifies the type of Simulator device to run your app on.
- **Architecture:** Currently only 32-bit is available for debugging.

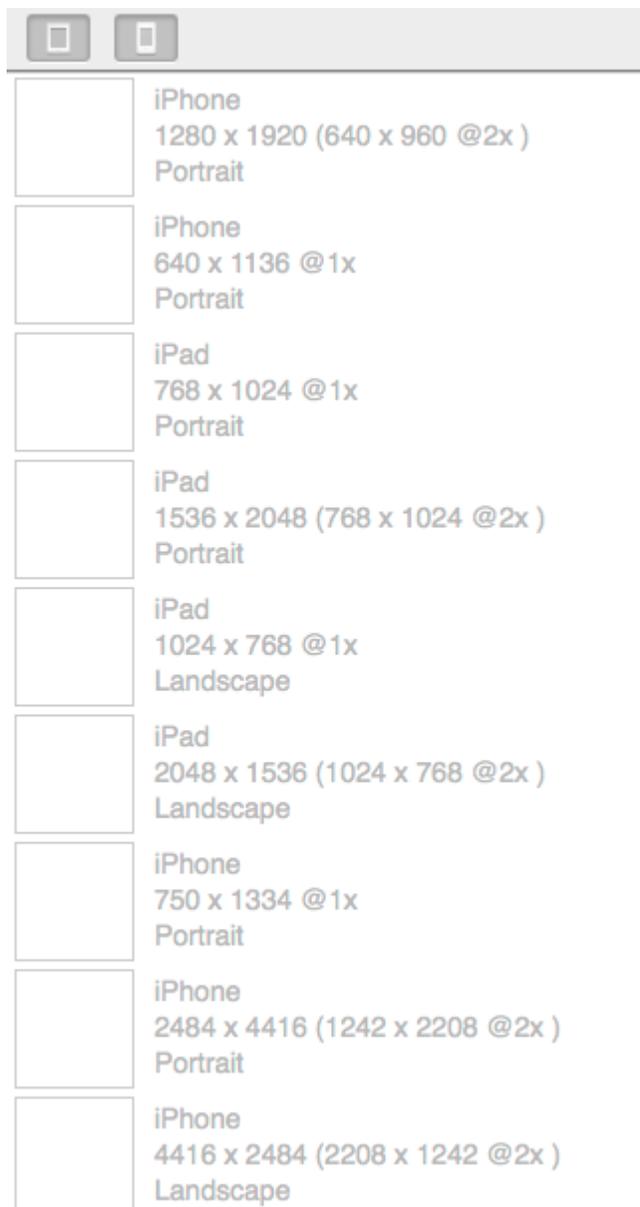
# Launch Images and App Icons

## Table of Contents

- [Launch Images](#)
- [App Icons](#)
- [iOS ImageMaker Utility](#)

## Launch Images

A Launch Image is a placeholder image that appears while the app is loading. When you create a new iOS project, an empty Launch Images Image Set is added to your project. When you click on it, you'll see a list of available launch image sizes that you can provide:



Drag the images you want to use to the appropriate item. Note that @2x images are used on retina devices and are

double the resolution of @1x images.

You can also use the [iOS ImageMaker utility](#) to create new launch images (in different colors) for inclusion in your projects (see below).



You must include the iPhone 640x1136 size launch image if you want your app to launch in full screen on an actual iPhone 5-sized device (including 5c, 5s and some iPod touch models). Without this image, the app will have black bars at the top and bottom when run on iPhone 5, 6 and 6 Plus screen sizes.



If you created a Launch Images folder using earlier versions of Xojo, you should migrate it to a Launch Image item by right-clicking on it and selecting "Convert" or by deleting the Launch Images folder, selecting Insert->Launch Image from the menu and then dragging in your images to the appropriate sections.

For more information about Launch Images, refer to the [iOS HIG on Launch Images](#) from Apple.

## App Icons

There are also a wide variety of app icon sizes for the various iOS devices. When you create a new iOS project, an empty App Icon item is added to your project. When you click on it, you'll see a list of available app icons sizes:



Drag the images you want to use to the appropriate item. Note that @2x images are used on retina devices and are double the resolution of @1x images.

You can use the [iOS ImageMaker utility](#) to create all the icon files for you from a single image or you can create them yourself (see below).

Your app icons should be opaque with no transparency.

If you created an App Icons folder using earlier versions of Xojo, you should migrate it to an App Icon item  by right-clicking on it and selecting "Convert" or by deleting the App Icons folder, selecting Insert->App Icon from the menu and then dragging in your images to the appropriate sections.

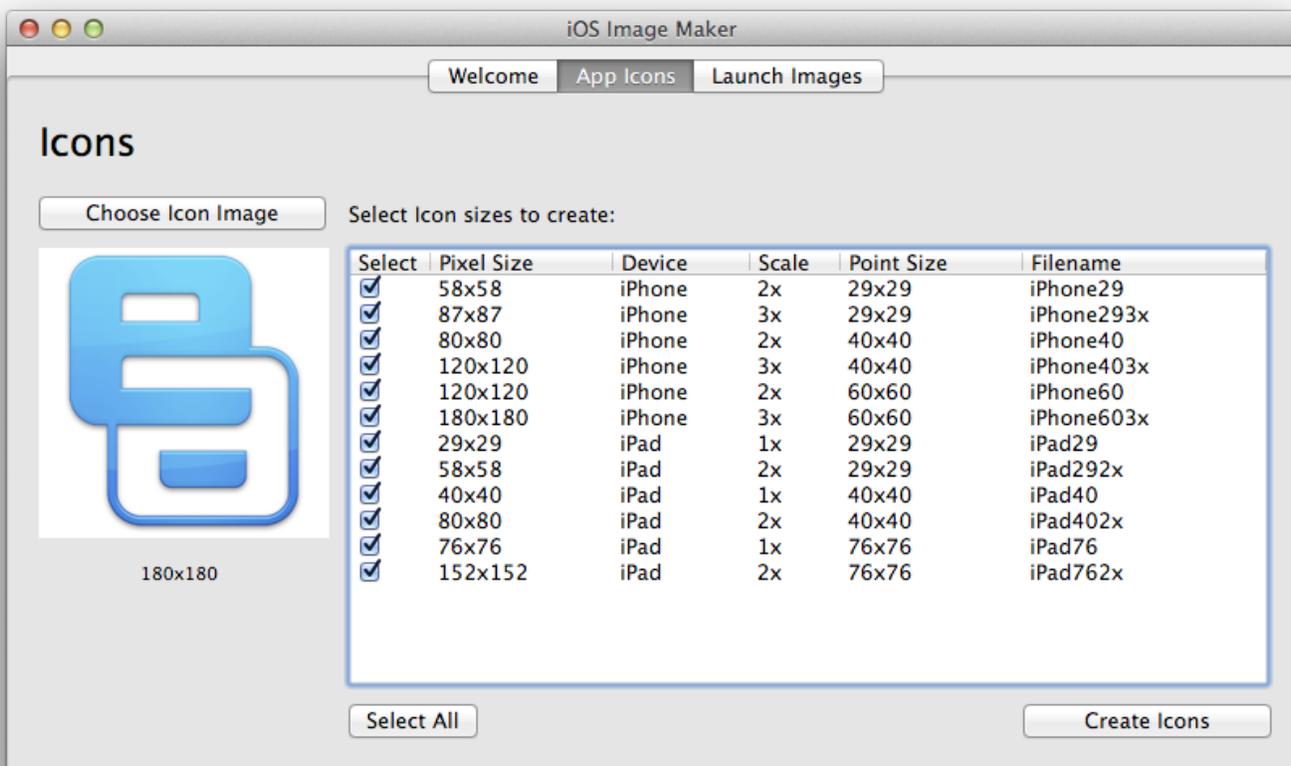
For a description of other App Icon recommendations from Apple, refer to the [iOS Human Interface Guidelines](#)

[section on App Icons.](#)

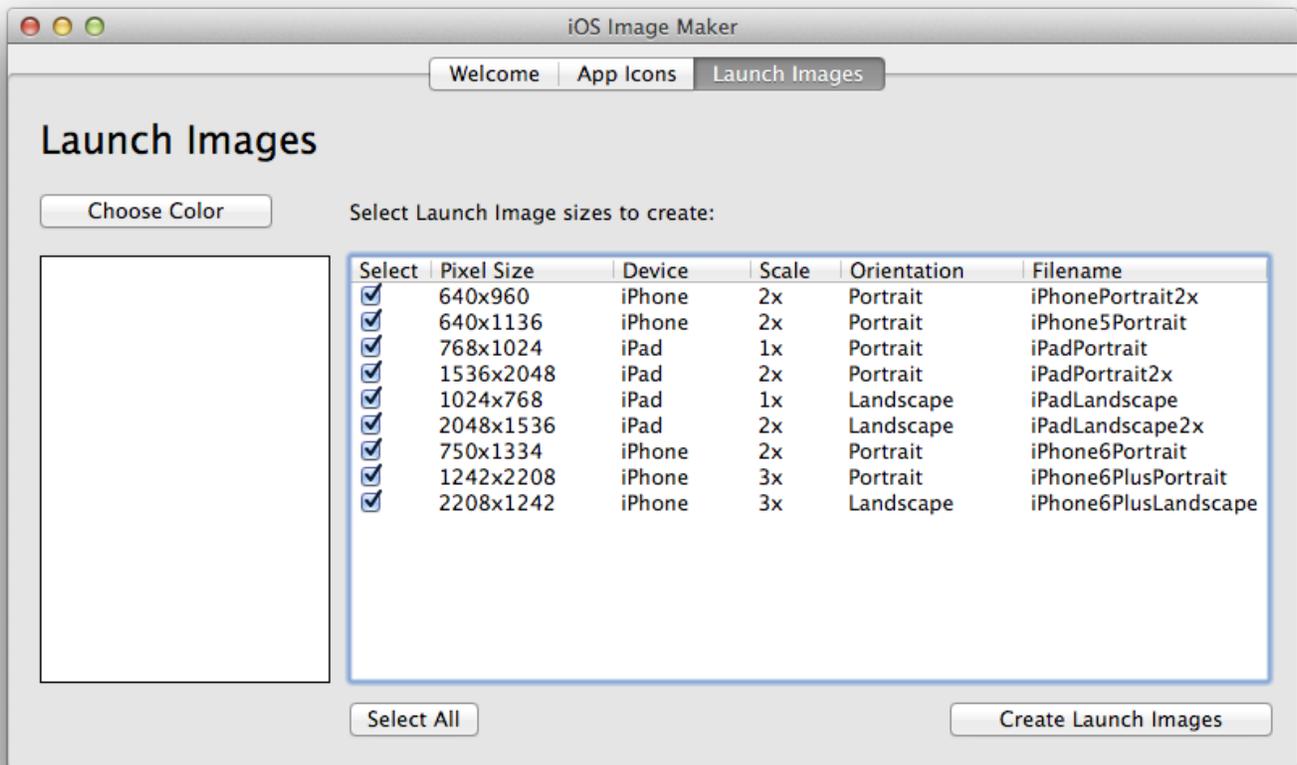
## iOS ImageMaker Utility

The iOS ImageMaker utility included with Xojo can be used to quickly create all the file sizes you need for your App Icon and Launch Images.

On the App Icons tab, choose the icon image to scale to the other sizes. For best results, select an icon that is 180x180 or larger. Then select the icons sizes you want and click the Create Icons buttons. The tool will create a folder containing the icon files. You can then drag each icon to the appropriate image size of the App Icon Image Set in your iOS project.



Creating Launch Images works similarly, but only a blank image of the selected color is created. These can serve as placeholders that you can update in image editing software if necessary. Simply choose the color for the launch image, the sizes you want and click Create Launch Images to get a folder of icons that you can drag into the Launch Images Image Set in your iOS project.



---

# Retina Images

When you add an image to your project (Insert->Image) and Image Set is created. An Image Set provides a way for you to manage all the different sizes of an image that are needed for iOS retina screens.

The Image Set editor displays three boxes for three different image sizes: 1x, 2x and 3x. Images sized at 1x are the original image as it appears on non-Retina devices. Images sized at 2x have double the horizontal and vertical resolution and are used on most retina devices. Images sized at 3x are triple the horizontal and vertical resolution and are currently only used on iPhone 6+.

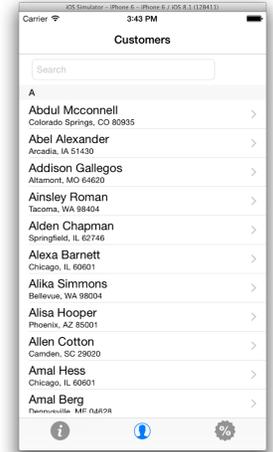
To add the images, simply drag them from Finder onto the appropriate box. For best results, iOS prefers PNG images. These images are not copied to your project, but are referenced as external items. This means if you move the external files or copy the project, the images may not be found. It is often convenient to create a ProjectImages folder next to your project where you place your images files and then drag them from there.

If you drag an image file onto the Navigator, a new Image Set with the image populated at the 1x size is created.

If you do not provide images in all three sizes, iOS will scale from the images you have provided. Scaling up from 1x images results in a blurry image, so at the very least you will want to specify images in 2x sizes.

# Running and Debugging in the iOS Simulator

Xojo can easily run your iOS apps in the iOS Simulator, which is included with Xcode. This means you'll need to have the latest version of Xcode installed (which is 6.1 as of this writing).



The easiest way to install Xcode is to [download it from the Mac App Store](#). Before you try to use the iOS Simulator with Xojo, you should be sure to start Xcode at least once so that you can agree to its license screen and install necessary components. If you do not do that, then the iOS Simulator will not start. However, you do not need to keep Xcode running in order to use the iOS Simulator.

**Note:** Xojo does not use Xcode to do its builds. It only needs Xcode to be installed so that the iOS Simulator and some necessary iOS frameworks are available for use by Xojo.

To run your iOS app in the iOS Simulator, you simply click the Run button on the Xojo toolbar. By default, your app will run on an iPhone 4S, but you can change the type of device in the Simulator Device property in the Inspector for the iOS Build Setting. Note that for iPhone 5 and larger, you will also want to have the appropriate [Launch Images](#) in place in order for you app to use the entire screen.



When you run the app in the Simulator, you have full access to the Xojo debugger. Breakpoints will stop at the line of code and everything should work the way you would expect.

Running in the Simulator is fast and convenient, but you should also always test your apps on actual devices. Some differences when running in the Simulator include:

- Apps built for the Simulator use the x86 compiler, not the ARM compiler.
- Not all device features are available in the Simulator.
- Keyboard may work differently in the Simulator.

 You may have noticed that the iOS entry in Build Settings has a built step called "Sign". This build step is used when you build your iOS apps for deployment. You cannot change its properties and you should not remove it.

## iOS Simulator Tips

For complete information about the iOS Simulator, refer to the [iOS Simulator User Guide](#). The tips below are from the section [Interacting with the iOS Simulator](#):

### Useful Gestures

Two Finger Drag	<ol style="list-style-type: none"> <li>1. Place the pointer where you want the two-finger drag to occur.</li> <li>2. Hold down the Option key.</li> <li>3. Move the circles that represent finger touches to the start position.</li> <li>4. Move the center of the pinch target by holding down the Shift key, moving the circles to the desired center position, and releasing the Shift key.</li> <li>5. Hold down the Shift key and the mouse button, move the circles in the direction you want to drag, and release both the Shift key and the mouse button.</li> </ol>
Pinch	<ol style="list-style-type: none"> <li>1. Place the pointer where you want the pinch to occur.</li> <li>2. Hold down the Option key.</li> <li>3. Move the circles that represent finger touches to the start position.</li> <li>4. Move the center of the pinch target by holding down the Shift key, moving the circles to the desired center position, and releasing the Shift key.</li> <li>5. Hold down the mouse button, move the circles in and out to the end position, and release the Option key.</li> </ol>
Rotate	<ol style="list-style-type: none"> <li>1. Place the pointer where you want the rotation to occur.</li> <li>2. Hold down the Option key.</li> <li>3. Move the circles that represent finger touches to the start position.</li> <li>4. Move the center of the pinch target by holding down the Shift key, moving the circles to the desired center position, and releasing the Shift key.</li> <li>5. Hold down the mouse button, rotate the circles to the end position, and release the Option key.</li> </ol>

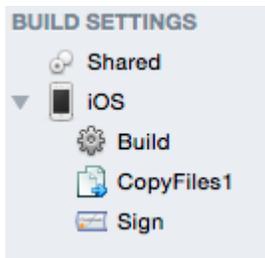
# Copying Files to the Device

There are two ways to copy files to your iOS device. You can use a [Copy Files Build Step](#) or you can use iTunes after you set your app up for [File Sharing](#).

## Using Copy Files Build Step

This process uses [Build Automation](#) to copy files into the Resources folder of your iOS app. When the app is running on the device, you can use the files directly or copy them out of the resources folder to another location (this is required to write to the files as the Resources folder is read-only).

1. A [Copy Files Build Step](#) is added to your project by selecting Insert->Build Steps->Copy Files. In the center area you can drag the files (or folders) you want to copy to the device.
2. In the Inspector, change the Destination to "Resources Folder".
3. You can also give this Copy Files step a name.
4. Drag the Copy Files Step from its original location onto the the iOS item in the Build Settings section of the Navigator. Be sure that it is after the Build item, but before the Sign item.



Any Copy File Build Steps that you add should be before the "Sign" step so that the files you have copied into the app bundle are properly signed for submission to the App Store.

When you Run or Build your project, these files are now copied to the Resources folder. Your app can access the files using [SpecialFolder.GetResources](#), which returns a FolderItem that points to the specified file in Resources. For example, if you've copied a file called MyFile.html, you can access it like this:

```
Dim myFile As FolderItem = SpecialFolder.GetResource("MyFile.html")</code>
```

Now that you have a FolderItem, you can use the file or you can copy it to Documents so that it can be modified using the [CopyTo](#) method:

```
Dim myFile As FolderItem = SpecialFolder.GetResource("MyFile.html")
```

```
If myFile.Exists Then
```

```
    Dim destFile As FolderItem = SpecialFolder.Documents.Child(myFile.Name)
```

```
    myFile.CopyTo(destFile)
```

```
End If
```

You can also copy files to a folder within Resources by specifying a value for the Subdirectory property in the

Inspector for the Build Step.

You still use `GetResource` to get the folder, which you can then iterate through to access the files:

```
Dim myFolder As FolderItem = SpecialFolder.GetResource("MyFolder")
```

```
If myFolder.Exists Then
  For Each c As FolderItem In myFolder.Children
    // use c as necessary to access or copy the file
  Next
End If
```

## iTunes File Sharing

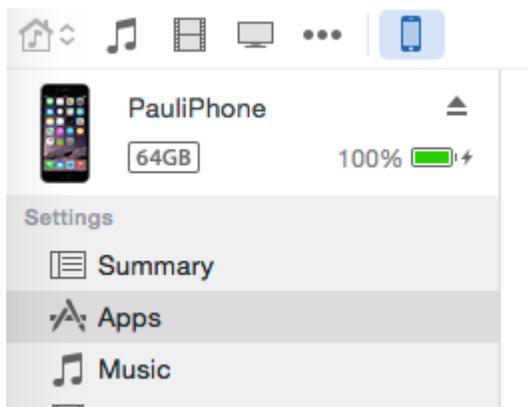
When you enable File Sharing, you can use iTunes to sync files to or from the applications Documentation folder on the device.

In order to enable File Sharing, you'll need to add a custom plist to your project. Copy the text below into a file and name it `Info.plist` and then drag it into your project:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>UIFileSharingEnabled</key>
<true></true>
</dict>
</plist></code>
```

Any files that your app creates in the [SpecialFolder.Documents](#) folder are now accessible via iTunes when your device is connected to the computer via USB.

1. In iTunes, click on the device and select Apps in the Settings navigator on the left.



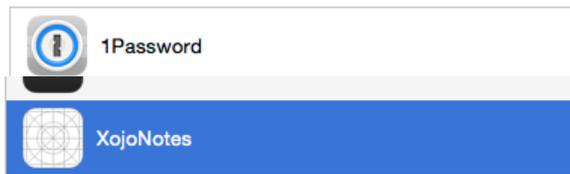
2. Scroll to the File Sharing section and you'll see a list of all the apps on your device that support file sharing. Find your app and click on it to see a list of the files in the Documents folder.

3. You can click on individual files to save them to your computer or you can choose to add files to the list. Files you add are copied to the device when you sync it using iTunes.

### File Sharing

The apps listed below can transfer documents between your iPhone and this computer.

#### Apps



#### XojoNotes Documents



To see this in action, watch [Webinar: iOS File Sharing with iTunes](#).

---

# iOS App Deployment

There are several ways to deploy iOS apps to devices, which are briefly outlined below.

## Ad-Hoc Deployment to Devices

With Ad-Hoc Deployment, you build your iOS apps using an iOS Development Profile and manually copy the built app to the device. To create this profile you need to be a member of the [Apple Developer Program](#) (currently \$99/year from Apple). You can deploy apps built in this manner on up to 100 devices.

For details on the steps to do this, refer to the section [Ad-Hoc Device Deployment](#).

## TestFlight

For beta testing purposes, you can also use Apple's TestFlight server to allow up to 1000 users to install a beta version of your app. Use of TestFlight also requires that you are a member of the [Apple Developer Program](#). Refer to Apple for more [information about TestFlight](#).

## App Store

To get your app in the hands of end users, you will typically want to use the App Store. To do so, you will build your app using an iOS Distribution Profile. You will have to be a member of the [Apple Developer Program](#).

For details on the steps to submit to the App Store, refer to the section [Submitting to the App Store](#).

## Volume Purchase Program

The Volume Purchase Program provides a way for you to distribute apps to businesses or educational institutions without having to publish your app to the App Store.

- [Volume Purchase Program for Business](#)
- [Volume Purchase Program for Education IT](#)

## Enterprise Deployment

Businesses can use the iOS Developer Enterprise Program to get tools and resources for developing proprietary, in-house iOS apps that you can distribute to your employees. These apps are distributed outside of the App Store.

For more information, visit the [iOS Developer Enterprise Program](#).

# Ad-Hoc Device Deployment for Testing

This section describes the steps to deploy your iOS apps onto iOS devices for testing. You can also watch the webinar: [Deploying iOS Apps](#).

## Table of Contents

- [Getting Started](#)
- [Configuring Provisioning Profile and Installing on your Devices](#)
  - [Add Profile to Devices](#)
- [Build the App](#)
- [Install the App on the Device](#)
  - [Using Xcode Devices Window](#)
  - [Using Apple Configurator](#)
- [Getting Crash Logs](#)

## Getting Started

In order to deploy your app to an iOS device for testing, you need to be a member of the [Apple Developer Program](#). There are two main steps to this. First, you need to create your certificates and provisioning profiles which are required to deploy onto a device. Then you need to install the profile onto the device. For these steps you will need to use Xcode. Some steps can also be done using the Apple Configurator tool. Generally speaking, you only need to do these steps once. But you may have to revisit them if you add devices or are setting up another Mac for deployment testing.

1. Start Xcode and open Preferences.
2. Select Accounts.

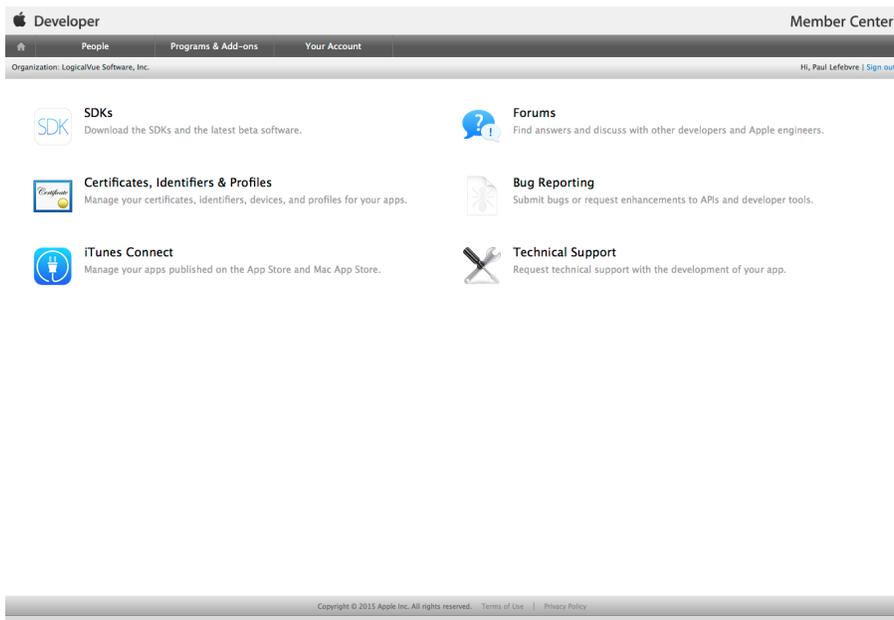


3. Add the Apple ID you used for your iOS developer account.
4. Select the account for the Apple ID and click the "View Details" button.
5. Click the "+" button and select "iOS Development" to request and create necessary certificates.
6. Click the Refresh button at the bottom of this window to load other files.

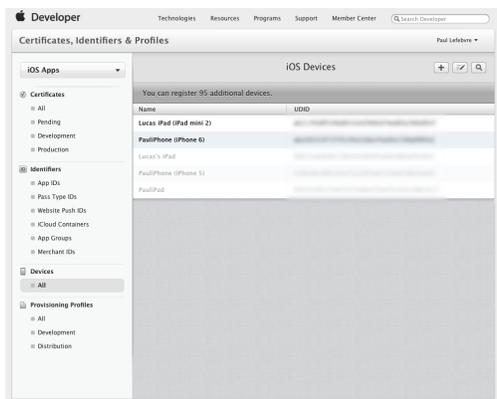
## Configuring Provisioning Profile and Installing on your Devices

Now you need to add any devices on which you will do testing.

1. Log into the Apple Developer Portal at <https://developer.apple.com> and select "Certificates, Identifiers & Profiles".



2. In the iOS Apps section, add your devices to the Devices section. Click on "All" in the Devices section and then click "+" to add the device. You'll need to supply the UDID any devices you want to deploy to. This UDID is displayed in the Xcode Devices window (Windows->Devices) where you can copy it to the clipboard.



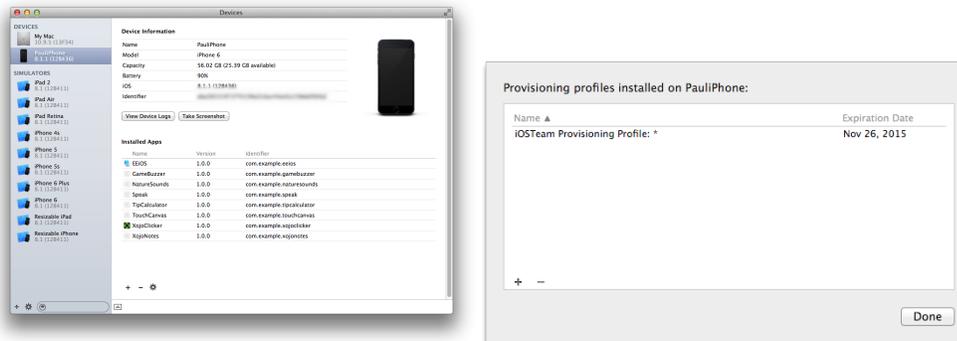
With the devices added, you can now request the provisioning profile from Xcode, which will also create a default wildcard App ID.

1. Go back to Xcode Preferences and select your account, click "View Details" and then Refresh. This loads the provisioning profile for Xcode, which is usually called "iOSTeam Provisioning Profile" and followed by the wildcard App ID. You may be prompted by Xcode to request a distribution certificate, but you can click Cancel on that for now.
2. Wildcard App IDs are typically of the form "com.company.\*". If you are having trouble getting Xojo to recognize the provisioning profile, you may want to instead create a wildcard ID of just "\*".

## Add Profile to Devices

Xcode should copy the provisioning profile to the device for you, but this is how you can do it manually should you run into problems:

1. Go back to the iOS Dev Portal and select Development in the Provisioning Profiles section.
2. Click on the profile and then click Download. Note the location of the downloaded file.
3. Plug in the device via USB.
4. Select Windows->Devices, right-click on the device and select "Show Provisioning Profiles".



5. In the dialog, click "+" and select the profile file that you just downloaded.
6. Click Done.

If you would rather use [Apple Configurator](#) instead of Xcode, you can click the "Install Profiles" button on the Prepare Settings screen to add the profile you downloaded and then click the Prepare button.

## Build the App

Now that your Mac and iOS device are configured with the provisioning profile, you can now build your iOS app using Xojo.

1. Click on iOS Build Settings and select your Code Signing Team/Company/Name in the Inspector.
2. Click the Build button in Xojo to build the app. This code-signs your apps and compiles it for ARM using LLVM.  
Be patient and let it finish as it may take a little longer than running in the Simulator.

When your app is built you will get the app bundle (this is what gets installed on the device) and a .dSYM file (the debugging symbols).

It is important that you retain both of these files for builds you deploy. The dSYM file is used for analyzing crash reports that you may receive from your users and without it, it will be difficult to determine the cause of the crash. For more information, refer to the Apple document [Understanding and Analyzing iOS Application Crash Reports](#).

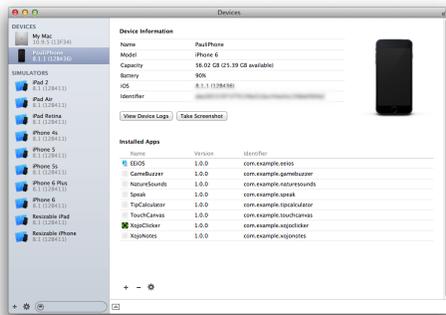
if you submit device crash logs in a Xojo Feedback case, you will also need to include the dSYM file in order for us to interpret them.

# Install the App on the Device

Now you are ready to install the app onto your device. You can do this using Apple Configuration or the Xcode Devices window.

## Using Xcode Devices Window

1. Ensure your iOS device is plugged into the Mac via USB.
2. If the Devices window is not displayed, select it from the Window menu.
3. Click on the iOS device in the sidebar.



4. Click the "+" button (In the Installed Apps section) and select the app you previously built (or drag the app into the Installed Apps section). This immediately installs it on the iOS device.
5. On the device, tap on the app to launch it.

## Using Apple Configurator

[Apple Configurator 2](#) is a free download from the Mac App Store. This tool is also used for enterprise deployment of apps, so it can do much more than just deploy built apps to devices. Some people might prefer using it instead of the Xcode Devices window (you'll still have to set up your device using Xcode as described above, however):

1. Ensure your iOS device is plugged into the Mac via USB.
2. Run Apple Configurator.
3. On the "All Devices" view, double-click your iOS device.
4. Click on Apps in the Navigator on the left. This displays all the apps on the device.
5. Click the Add button in the toolbar, select "Apps" from the menu and in the dialog, click "Choose from my Mac...".
6. Select the app you previously built. Click OK to copy the app to the device.
7. On the device, tap on the app to launch it.

For more information about the specifics of certificates, profiles and deployment, be sure to read the Apple docs in the iOS Developer Portal.

---

## Getting Crash Logs

You can retrieve crash logs from your iOS device using the Devices window in Xcode. Plug your device in via USB and select it in the Devices window. Then click the "View Device Logs" button. You'll see a list of crash logs (it may take a few minutes to download if you have a lot of them). Find your app in the list and click it to see the log. When submitting Feedback cases with crash logs, wait to ensure that the symbols for your log are loaded before you send the log.

This tool is also useful to reading iOS console logs:

- [iOS Console](#)

---

# Submitting to the App Store

When you have finished developing and testing your app, you can submit it to the App Store to make it available to everyone. These steps should help get you started, but most of the specifics are defined by Apple, so you may want to also review the official Apple [App Distribution Guide](#).

This section describes the steps to deploy your iOS App to the App Store. You can also watch the webinar: [Deploying iOS Apps](#).

To build for the App Store, you need to be a member of the [Apple Developer Program](#) and have the correct certificates, identifiers and a distribution provisioning profile. You will be using Xcode and the iOS Development Portal to set this up.

## iOS Distribution Certificates

1. In Xcode Preferences, select your account and "View Details".
2. Click the "+" button and choose "iOS Distribution" to request the necessary certificate files.
3. Click the Refresh button to load any other necessary files.

Next, you have to create an App ID at the iOS Dev Center.

## Identifiers

This step is optional, since you can use the same App ID you have used for device deployments.

- Log in to the iOS Developer Center at <http://developer.apple.com>.
- Select "Certificates, Identifiers & Profiles" in the iOS Developer Program sidebar.
- In the Identifiers section, click App ID and then the "+" to register a new Application ID. It is easiest to specify a "Wildcard App ID" of the format "com.mycompany.\*", which can be used with all your apps.
- Click Continue and then Submit.

Now you can create your Distribution Provisioning Profile.

## Generate Distribution Provisioning Profile

In the Provisioning Profile section, click Distribution.

- Click "manually generate profiles".
- Click "App Store" in the Distribution section and then Continue.
- Follow the steps to select the identifier and create the profile.
- Once the profile is created, go the Xcode Preferences and select the Accounts tab.
- Select your Apple ID, choose the team name and select "View Details".
- Click the Refresh button in the lower left to load the provisioning profile.

Now you are ready to build your Xojo app.

## Build Xojo App

- In Xojo, set "Build for App Store" property to ON in the iOS Build Settings. Note: This setting is always set to False when the project is opened.
- Select the appropriate Code Signing team and optionally specify any entitlements (as a plist file).
- Verify that the bundle ID matches your wildcard app ID. For example if your wildcard ID is "com.mycompany.\*", then your bundle ID must start with "com.mycompany" and be followed with the app name. So a final bundle ID might be "com.mycompany.myapp".
- In Shared Build Settings, ensure that all version fields are correct and that "Short Version" has a value. If you have to submit an update to the App Store, you will need to increase the version information.
- Click Build. This creates an IPA (iOS App Archive) file.

### If you use Build Steps



On iOS, there is an additional step added by default that is called "Sign". Any Copy File Build Steps that you add should be before the "Sign" step so that the files you have copied into the app bundle are properly signed for submission to the App Store.

## iTunes Connect

If you have not already created the app details in iTunes Connect, you'll need to go do that now.

- Visit <http://itunesconnect.apple.com> and log in.
- Click "My Apps" and click the "+" button to add a new iOS app.
- Here you will need to fill in all the information on the various pages, such as app name, description, screen shots, icon (1024x1024), rating, price, etc. You can save and go back to make updates.
- When everything is correct, the app status should show as "Prepare for Submission".

## Upload App

When everything is filled out, you are now ready to upload the build you created.

- Launch Application Uploader (from Xcode, in the Xcode menu, select Open Developer Tools->Application Loader).
- Follow the steps to select the IPA file and send it to iTunes connect for verification.
- If there were errors, you'll need to correct them and resubmit. Don't forget to update the version information!

## Submit for Review

Once everything is verified properly, your final step is to go back to iTunes Connect online.

- Select your app from the list.
- Select the build that was just uploaded.
- Click "Submit for Review"

Be patient. It can take several days before an app is approved. If your app reviewers finds issues, you'll need to fix them, rebuild, upload and resubmit.

<http://appreviewtimes.com> is a good place to go to get estimates on current review times.

---

# iOS Deployment Using Free Provisioning Profile

Starting with Xcode 7 and OS X 10.11 El Capitan, Apple now provides a free provisioning profile you can use to deploy your apps to an iOS device without requiring a paid Apple Developer membership.

Setting this up is a bit of work and you'll have to do it for each and every app you want to deploy on the device.

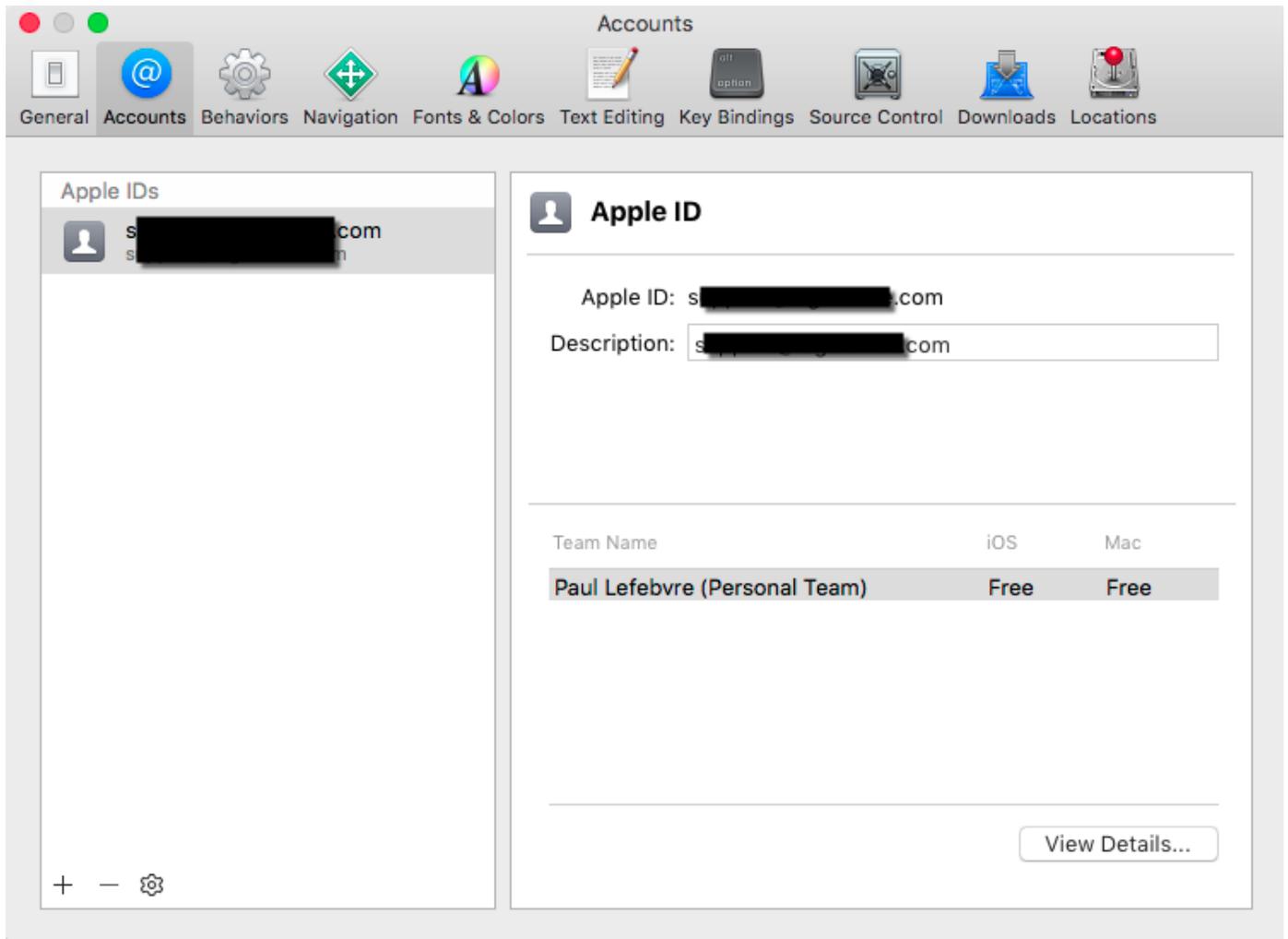
## Requirements

You will need the following in order to use the free provisioning profile:

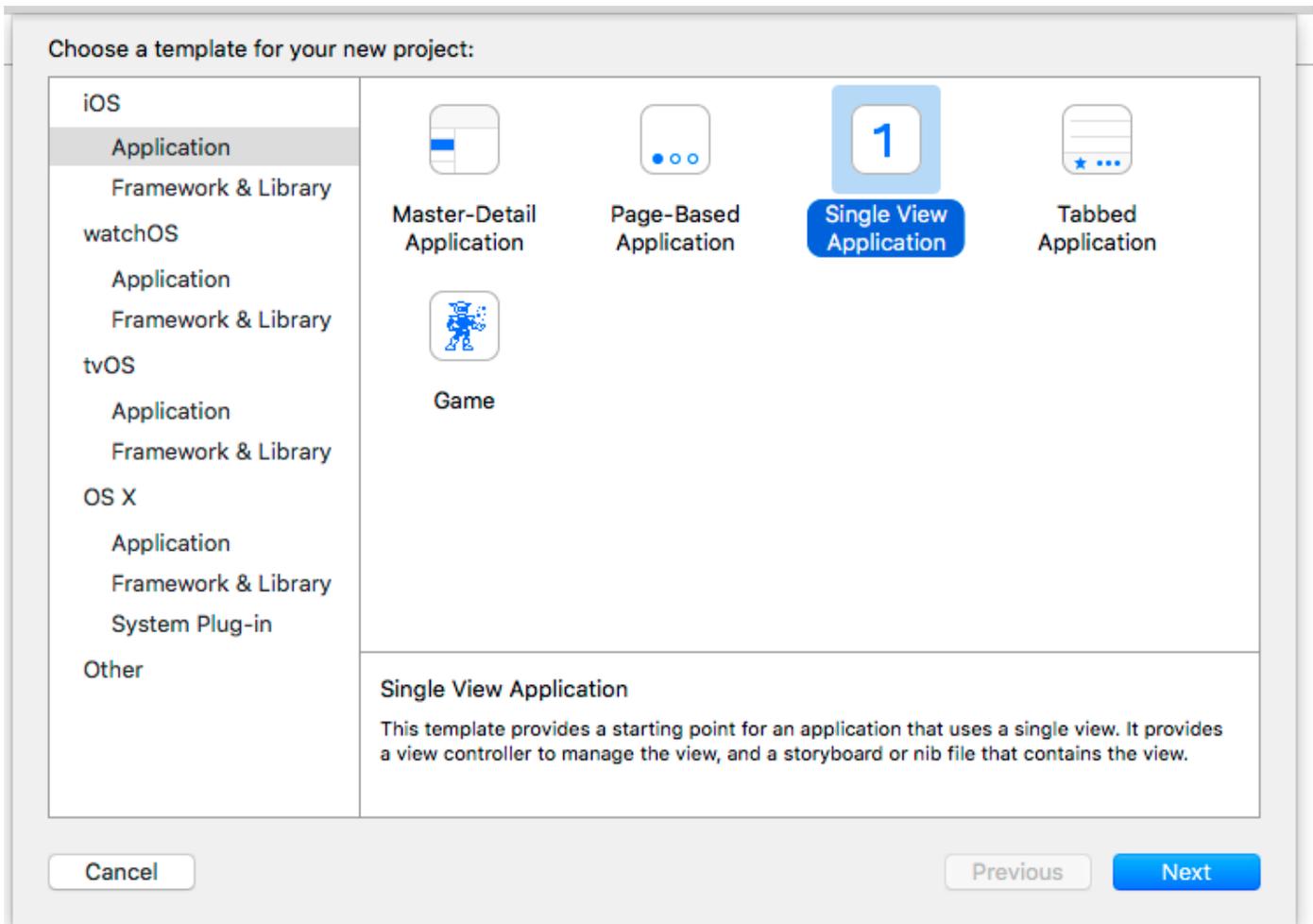
- Xcode 7
- An Apple ID that is not associated to an Apple Developer account

## Xcode Configuration

1. If you do not have an Apple ID you can use, you can create a new one at [appleid.apple.com](http://appleid.apple.com).
2. Start Xcode and open Preferences (Xcode menu -> Preferences).
3. Select the Accounts tab and click the "+" button to add your Apple ID. This adds a Team Name which should be selected by default.



4. Click View Details for the selected Team Name. In the dialog that appears, click the Create button next to iOS Development. This creates the Signing Identity. When it has finished, click the Done button.
5. Plug in the iOS device (via USB) that you will deploy you app to.
6. Create a new Xcode project: Choose File->New Project and select "Single View Application" in the dialog and the click the Next button.



7. For the Product Name, enter the name you will use (or have used) for your Xojo app.
8. For the Organization identifier, enter the first part of the Bundle Identifier (i.e. com.example). Click the Next button.

Choose options for your new project:

Product Name: XojoTest

Organization Name: Paul Lefebvre

Organization Identifier: com.example

Bundle Identifier: com.example.XojoTest

Language: Objective-C

Devices: Universal

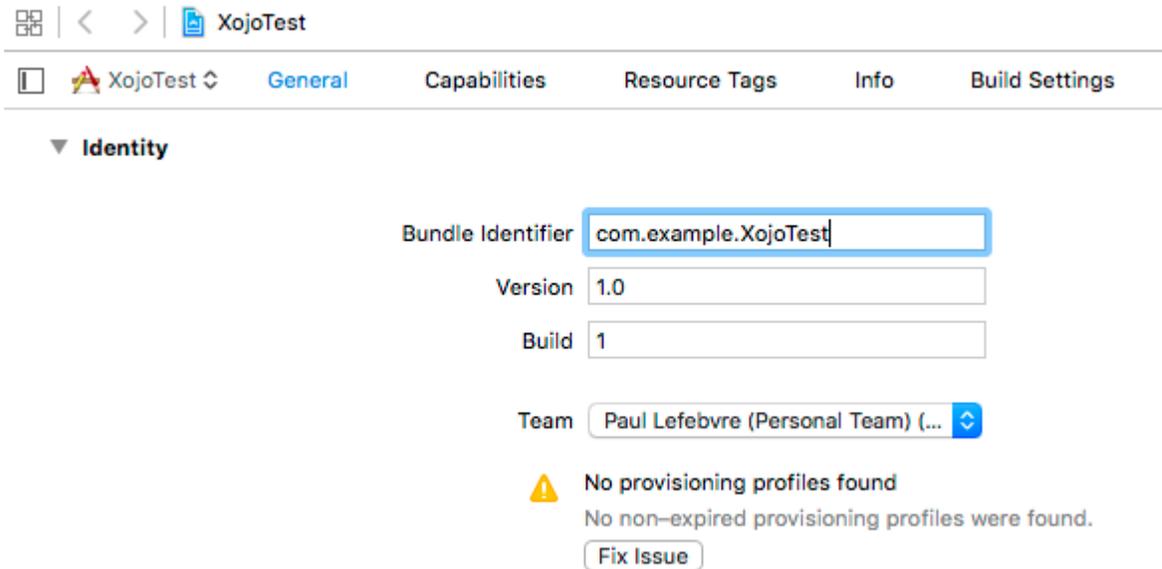
Use Core Data

Include Unit Tests

Include UI Tests

Cancel Previous Next

9. Select a folder to save the project. You will want to uncheck the Source Control option since you're not really going to use this project for anything. Click the Create button to create the project.
10. In the General section that appears, verify that the Bundle Identifier exactly matches what you will use (or have used) for your Xojo app. The next steps will only create a provisioning profile for a single app, so this Bundle Identifier is important. And you'll need to follow steps 5 and later for each new app you want to deploy.
11. In the Team drop-down, make sure your Apple ID is selected.



12. Check the Project Name gadget in the toolbar at the top to make sure that the iOS device you have plugged in is selected. If it is not, click the gadget and select your device from the list.
13. Now click the Fix Issue button that is in the General area.

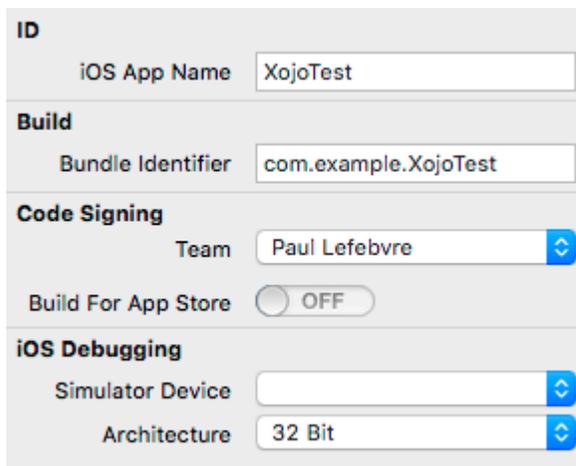
This will send the profile over to the device. You are now done with this Xcode project, so you can close it. Keep Xcode running as you'll want to use its Devices window to transfer the app you build with Xojo to the device.

**i** You will have to repeat steps 5 to 13 for each new app that you want to deploy to the device.

## Xojo Setup

With Xcode configured, you can now configure your Xojo project to use the Bundle Identifier and Team.

1. Start Xojo and open your iOS project (or create a new one).
2. In the iOS Build Settings, enter the Bundle Identifier that you used in the Xcode project. It must match exactly.
3. In the Team drop-down, select your Apple ID you used with the Xcode project.



4. Click Build to build the app.

---

## Deploy to Device

You can now transfer the built app to your device using the Xcode Devices window:

1. Ensure your iOS device is plugged into the Mac via USB.
2. If the Devices window is not displayed, select it from the Window menu.
3. Click on the iOS device in the sidebar.
4. Click the "+" button (In the Installed Apps section) and select the app you previously built (or drag the app into the Installed Apps section). This immediately installs it on the iOS device.
5. On the device, tap on the app to launch it.

## Troubleshooting

- In order to run the app on the device, you may have to tell the device to trust the "developer". To do so, go to Settings->General and scroll to Profile. There you should see your Apple ID, which can click. If you see a message about "Trust" and your Apple ID, then you can click that button and click Trust in the dialog to allow the app to run.
- If you are unable to Run/Build from Xojo, check your Xcode Command Lines Tools setting. Start Xcode, open Preferences and click the Locations tab. Check the Command Lines Tools setting to make sure it has a selection that matches the version of Xcode you have installed.

[Reference](#)

# Desktop Apps

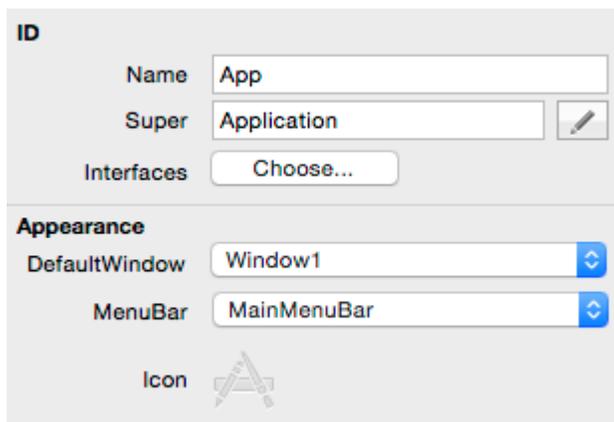
Desktop apps have a graphical user interface and are run directly on the user's computer.

## Table of Contents

- [App Object](#)
  - [Event Handlers](#)
  - [Properties](#)
- [Build Settings](#)
  - [Shared](#)
  - [OS X](#)
  - [Windows](#)
  - [Linux](#)
  - [This Computer](#)
- [Building and Deployment](#)
  - [Build Folder Names](#)
  - [OS X](#)
  - [Windows](#)
  - [Linux](#)

## App Object

For a desktop project, the App object is a subclass of [Application](#). You use it to specify the default window that opens when your app starts, the default menu bar that is used by the app and the app icon (in various sizes).



## Event Handlers

The App object has several event handlers:

- **Open:** This event handler is called when your app first starts, either by running the app directly (in debug mode) or by running a built app.
- **Close:** The close event is called when the user quits the app.
- **NewDocument:** This event is called when the built app is run without supplying a document.
- **OpenDocument:** This event is called when one of the app's documents is double-clicked from the desktop causing the app to start. It is also called if you drop a document on the app.
- **EnableMenuItems:** This event is called when the user clicks in the menu bar but before any menu items are

displayed. The `EnableMenuItems` event handler executes after the `EnableMenuItems` event handler of any classes with instances in the frontmost window and after the window's `EnableMenuItems` event handler. This is the event handler that should be used to enable menu items that should be enabled regardless of whether there is a window open or not (this possibility exists on OS X app). Note that if a menu item should always be enabled, you should use its `AutoEnable` property instead of an `EnableMenuItems` event handler.

- **HandleAppleEvent:** Executes when an `AppleEvent` is received by the application.
- **Activate:** The application is being activated. This occurs when the application is opening and when it is being brought to the front.
- **Deactivate:** The application is being deactivated. This occurs when another application or a desktop window is being brought to the front or when the application quits.
- **UnhandledException:** Called when a runtime error occurs that is not caught by your code. This event gives you a “last chance” to catch runtime errors before they cause your application to quit.

## Properties

- `DefaultWindow`: This is the window that is opened automatically when the app starts.
- `MenuBar`: This is the primary menu bar for the app. It is used as the default menu bar for newly created windows. This property is accessible by the `App` function (see next topic).
- `Icon`: This opens the Icon Editor and lets you specify the various size icons that are used by the app.

## Build Settings

The Build Settings section of the Navigator contains the build-specific settings for your app. You can check the box next to each target in order to build an app for that target.

 Note: Properties in this section that are accessible by the `App` function are in italics.

## Shared

The Inspector for Shared settings contains these properties:

The screenshot shows the Shared settings Inspector with the following sections and controls:

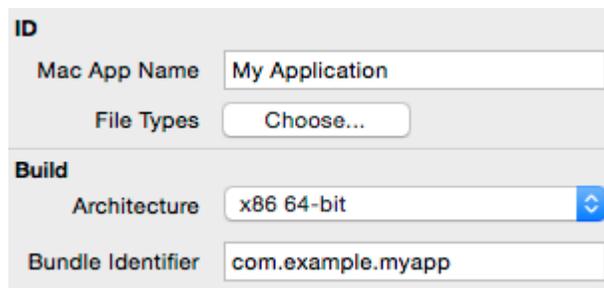
- Version Section:**
  - Major Version: Text input field containing "1"
  - Minor Version: Text input field containing "0"
  - Bug Version: Text input field containing "0"
  - Stage Code: Dropdown menu showing "Development"
  - Non Release Version: Text input field containing "0"
  - Auto Increment Version: Toggle switch set to "OFF"
  - Short Version: Text input field (empty)
  - Long Version: Text input field (empty)
  - Package Info: Text input field (empty)
- Build Section:**
  - Use Builds Folder: Toggle switch set to "ON"
  - Include Function Names: Toggle switch set to "ON"
  - Language: Dropdown menu showing "Default"
- Debug Section:**
  - Command Line Arguments: Text input field (empty)
  - Destination: Text input field containing "N/A" with a pencil icon for editing

- **Major Version** (*MajorVersion*): The Major version for your app. Version numbers are usually written as 1.2.3.4, where "1" is the major version.
- **Minor Version** (*MinorVersion*): The Minor version for your app. Version numbers are usually written as 1.2.3.4, where "2" is the minor version.
- **Bug Version** (*BugVersion*): The Bug version for your app. Version numbers are usually written as 1.2.3.4, where "3" is the bug version.
- **Stage Code** (*StageCode*): Used to indicate the type of app you are building (Development, Alpha, Beta, Final).
- **Non Release Version** (*NonReleaseVersion*): The Non Release (build) version for your app. Version numbers are usually written as 1.2.3.4, where "4" is the non release version.
- **Auto Increment Version Info**: When ON, the Non Release Version is increased by one each time you do a Build (but not when you Run).
- **Short Version** (*ShortVersion*): A short text description for your app. Usually this contains just the version number (such as 1.2.3.4) and is displayed by some operating systems in Get Info or Property windows for the app.
- **Long Version** (*LongVersion*): A longer text description for your app. Usually this contains the app name, copyright, version and other information. This is displayed by some operating systems in Get Info or Property windows for the app.
- **Package Info** (*PackageInfo*): A text description for your app that may be displayed by some operating systems in Get Info or Property windows for the app.
- **Use Builds Folder**: When ON, a separate folder is placed alongside the project file. Each platform that is built (OS X, Windows, Linux) also gets its own subfolder within this build folder.
- **Include Function Names**: When ON, the actual names of your function calls are embedded in the built app. This is useful for debugging purposes and for getting stack traces.
- **Language**: Allows you to specify the language to use to resolve dynamic constants.
- **Command Line Arguments**: These are the command-line arguments that are passed to your app when you run it in Debug mode.
- **Destination**: Specifies the path where the app is located when you run it in Debug mode. If not specified, then

the app is placed alongside your project file (or in a folder alongside the project file on Windows and Linux).

## OS X

The OS X section specifies settings used when building the OS X app.

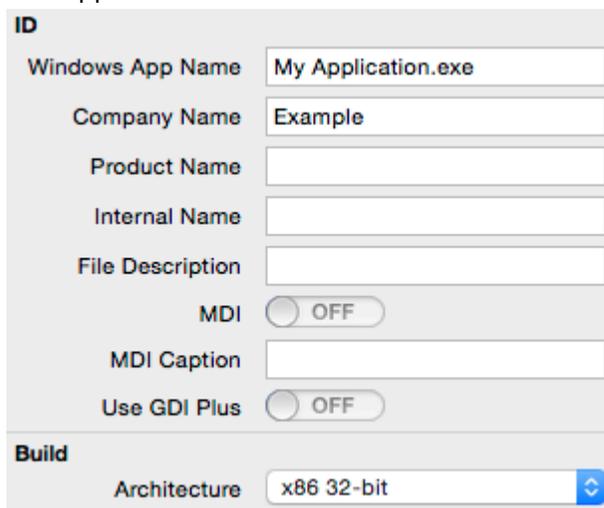


The screenshot shows the OS X settings panel. Under the 'ID' section, there is a text field for 'Mac App Name' containing 'My Application' and a button for 'File Types' labeled 'Choose...'. Under the 'Build' section, there is a dropdown menu for 'Architecture' set to 'x86 64-bit' and a text field for 'Bundle Identifier' containing 'com.example.myapp'.

- **Mac App Name:** The actual file name for your OS X application. If not specified, the ".app" suffix is automatically added when the app is built.
- **File Types:** Opens the Accept File Types dialog used to specify the acceptable file types that the app supports. For each file type, you can specify its role as None, Viewer, Editor or Shell.
  - None: Your app cannot handle the file type.
  - Viewer: Your app can open and read the file type, but cannot save it.
  - Editor: Your app can open, read, edit and write using the file type. This is what you should normally choose.
  - Shell: Not documented by Apple.
- **Architecture:** The CPU architecture for the app. Choices are x86 32-bit and x86 64-bit. Currently only 32-bit apps can be debugged.
- **Bundle Identifier:** The bundle identifier is used by OS X as a unique descriptor for your application. It is usually specified as a reverse domain name, such as com.xojo.myapp. This is added to the CFBundleIdentifier section of the app's plist. A bundle identifier is required for OS X apps.

## Windows

The Windows section specifies settings used when building the Windows app.



The screenshot shows the Windows settings panel. Under the 'ID' section, there are text fields for 'Windows App Name' (My Application.exe), 'Company Name' (Example), 'Product Name', 'Internal Name', and 'File Description'. There are also two toggle switches: 'MDI' (OFF) and 'Use GDI Plus' (OFF). Under the 'Build' section, there is a dropdown menu for 'Architecture' set to 'x86 32-bit'.

- **Windows App Name:** The actual file name for the Windows app. If not specified, the ".exe" suffix is automatically added when the app is built.
- **Company Name:** On Windows 7 and later, the "Company Name" appears in the Copyright section of the app properties in the Details tab.
- **Product Name:** The name of the product as installed in the Windows Start → All Programs menu. This also

appears in "Product name" on the app properties Details tab.

- **Internal Name:** This is useful when your product has a different internal name than its external name. This is not shown on the app properties Details tab.
- **MDI:** When ON, the app's windows will be enclosed in the "parent" MDI window. MDI stands for Multiple Document Interface. Although still supported by Windows, it is not commonly used.
- **MDI Caption:** If MDI is ON, you can enter the caption that appears in the title bar of the MDI window.
- **Use GDI Plus** (*UseGDIPlus*): When ON, enables the newer GDI+ graphics subsystem that has support for transparency.
- **Architecture:** The CPU architecture for the app. Choices are x86 32-bit and x86 64-bit. Currently only 32-bit apps can be debugged.

## Linux

The Linux section specifies settings used when building the Linux app.

The screenshot shows a settings panel for a Linux app. It is divided into two sections: 'ID' and 'Build'. Under 'ID', there is a text field for 'Linux App Name' containing the text 'MyApplication'. Under 'Build', there is a dropdown menu for 'Architecture' currently set to 'ARM 32-bit'.

- **Linux App Name:** The actual file name for the Linux app.
- **Architecture:** The CPU architecture for the app. Choices are ARM 32-bit, x86 32-bit and x86 64-bit. Currently only X86 32-bit apps can be debugged.

## This Computer

The **This Computer** section shows you the appropriate section (OS X, Windows or Linux) for the platform you are currently using. For example, if you are running Xojo on OS X, then This Computer shows the OS X settings.

# Building and Deployment

To create a stand-alone build for deployment, you create a build by clicking the Build button on the main toolbar or choose Project → Build. This creates apps for each of the OS platforms you selected, with separate folders for each.

## Build Folder Names

OS	Architecture	Folder Name
OS X	x86 32-bit	Mac OS X (Cocoa Intel)
OS X	x86 64-bit	OS X 64 bit
Windows	x86 32-bit	Windows
Windows	x86 64-bit	Windows 64 bit
Linux	x86 32-bit	Linux
Linux	x86 64-bit	Linux 64 bit
Linux	ARM 32-bit	Linux ARM

Desktop apps are deployed by installing them on a computer and running them on the computer. Once you have created your builds, you'll need to test them on the target OS and possibly package them into a format for distribution (such as an installer).

 Always test your apps on the specific target platform!

## OS X

OS X apps consist of a single App file, called the Application Bundle. You can embed other files into the Application Bundle because it is technically a folder that OS X treats as a file. In Finder, you can right-click on any App and select "Show Package Contents" to see the actual contents of the Application Bundle as a folder. Of course, your app can still have separate files not in the Bundle, in which case you want to make sure that your application is in a folder of its own.

Since apps are technically a folder, you have to include them in some sort of container in order to distribute them, such as disk images (DMG), installers and even simple Zip files.

### Disk Images

A disk image is a file that the user downloads. After downloading the file (and double-clicking it), it appears in the sidebar of the Finder as a drive. This is called "mounting" the disk image. When the user clicks on the drive in the Finder sidebar, they see your application and typically drag it to the Applications folder to install it.

To create a disk image, you can use the Disk Utility application included with OS X or try one of the many specialized disk image creation tools such as DMG Canvas. A disk image is probably the most common way to install OS X applications, but keep in mind that some users (especially those new to OS X) may find the concept of mounting a drive and dragging a file to the Applications folder very confusing.

Also keep in mind that if the user mistakenly tries to run the Application directly from the disk image, it may not behave as expected because the disk image is read-only.

### Installers

OS X can also use an actual installer to install your application, but it is not common for applications distributed outside the Mac App Store. To create an installer you can use the PackageMaker tool (included with the OS X Development tools) or you can use the free [Packages](#) installer.

An installer gives you more control over permissions and other settings, but they can be much more difficult to create than a simple disk image. Also, remember that an installer on OS X (a pkg file) is actually a bundle so you have to distribute it in a disk image or a Zip.

### Zip

A Zip file is an archive of your application. A zip is easy to download and most users understand what they are.

They can usually be unzipped by simply double-clicking on them, which reveals the application itself. The application can then be copied to the Applications folder. You can create a Zip by right-clicking on your application in the Finder and selecting Compress.

## Code Signing for GateKeeper

OS X 10.8 introduced a new feature called GateKeeper. GateKeeper is designed to provide a level of security for users installing applications. Essentially, if the application (or its installer) is not digitally signed using an Apple-provided certificate then OS X will not allow the application to be installed by default.

This can be overridden by the user right-clicking on the application (or installer) and selecting Open, but not many users will know about this.

This means you are probably going to want to digitally sign your OS X applications. To get a certificate, you have to join the Mac Developer Program (\$99). Your certificate is good for five years.

You use Certificate Utility in the Mac Dev Center to create a certificate (you can use the same certificate for all your applications). Once you have followed the instructions to create the certificate and have installed it on your computer, you can use it to code sign your application using this command:

```
codesign -f -s "Developer ID Application: YourName" "YourXojoApp.app"
```

Code signing must be done as the absolute last step. If you modify anything inside your application bundle (such as Info.plist) after you code sign, you will invalidate the signature and have to code sign again.

 Note: If you are using an installer then you have to sign the installer separately using a special installer certificate.

## Mac App Store

The Mac App Store is a great way to distribute your OS X apps. People find it easy and convenient to purchase applications from the Mac App Store. Unfortunately, getting applications into the Mac App Store is not easy or convenient for the developer.

Apple requires that all apps submitted to the Mac App Store are sandboxed. Sandboxing is used to restrict what your app is able to access. This serves as a security feature, because if an app were to become compromised for some reason then it would be unable to do as much damage as if it had full control of the computer.

To be able to submit an application to the Mac App Store for Apple to review, you need to first create your Mac App Store certificates using the online Certificate Tool that is part of the Mac Dev Center. You also need to create a bundle ID for your application. When your application is complete, you then need to code sign your application (and all its dynamic libraries). You also need to create the installer and code sign that as well.

Finally you can create a submission using iTunes Connect, fill in all the required information and then upload your application using the Application Loader tool that is part of the OS X Development Tools. Once you have submitted something to the Mac App Store, it can take several weeks before your application is approved by Apple and ready for sale.

For free apps, you'll only have to worry about sandboxing and code signing. But if you are selling your app, you will want to prevent people from copying the purchased app to another computer and running it there (without having to log into their Apple ID). To do this you also need to verify the Apple ID.

Verifying an AppleID requires calling a Cocoa API. The open-source MacOSLib library has classes and methods to do

this.

## Windows

Windows apps have a minimum of these parts: the main executable, its resources (contained in a Resources folder) and its libraries (contained in a corresponding Libs folder). All parts are required in order for the app to run.

**Note:** The Libs and Resources folder can be renamed to "AppName Libs" or "AppName Resources", where AppName is the name of your app. This can be useful should you want to package several apps into a single folder.

Generally you should use an installer to deploy Windows applications. With an installer you can create shortcuts in the Start menu or Desktop and copy the files to the appropriate locations. There are many products available for creating installers for Windows applications. Two that work well with Xojo applications are InnoSetup and Advanced Installer.

For simple needs or for testing purposes, you can also just zip the Windows application and its related folders for distribution. You can create a Zip by right-clicking on the parent folder in Windows Explorer and selecting Send To->Compressed (zipped) Folder. Once unzipped, the application can be run from any location. Although this can be useful for testing purposes, it is not recommended for proper Windows application deployment.

**Note:** When you compile an OS X on Windows, the resulting application is created as a tar archive file. You can transfer this file to OS X and expand the archive in order to run it.

### Installers

When you create your installer, you need to tell it to include all the files necessary to run the application. At a minimum, this includes the EXE file, the Resources folder and the Libs folder. For example, an application called Sliders would create a file called "Sliders.exe" and a folder called "Sliders Libs". The Libs folder contains DLLs for libraries, plugins and other associated files needed by your application.

If your application has other support files or folders, such as a Resources folder, then make sure that your installer includes them as well.

Most installer tools allow you to create your installer as a Setup.exe file or as an MSI (Microsoft Installer) file. Either work fine, but MSI files have the advantage of being the current recommended method from Microsoft and can be used by IT departments for better control of installations. Choose what works best for your customers.

Windows applications are installed to the Program Files folder. On 64-bit systems, your 32-bit applications are installed to the Program Files (x86) folder (because they are 32-bit).

Windows users expect to have easy access to your application, so this means you should create easily accessible shortcuts. Your installer tool should provide you with the option of creating a shortcut for the user on the Desktop and in the Start Menu.

### Microsoft Redistributable Files

Your applications require the Microsoft Visual C++ Runtime. The required files are included in the Libs folder

(msvcp100.dll and msvcr100.dll) of your applications for your convenience.

However, for increased Windows compatibility, you should instead consider including the Microsoft Visual C++ Runtime Redistributable Installer as part of your installer. To do this you do not include the msvc100.dll and msvcr100.dll files that are in the Libs folder and instead embed the Microsoft Visual C++ Runtime Redistributable as part of the installer. Most installer tools have a way to do this automatically.

The advantage of doing it this way is that the Visual C++ Runtime Redistributable will install its files into the Windows system folder, which allows them to be updated by Microsoft Windows Update for security and other reasons. If you leave your Visual C++ Runtime files in the Libs folder then they cannot be updated by Windows Update.

As of this writing, you can download the Visual C++ Runtime Redistributable from here:

- [Microsoft Visual C++ 2010 Redistributable Package \(x86\)](#)
- [Microsoft Visual C++ 2010 Redistributable Package \(x64\)](#)

## Linux

Linux applications also have two parts: the main executable and its libraries (contained in a corresponding Libs folder). Both parts are required in order for the application to run. It is also possible that there is a Resources folder that may contain localization files (or other files that you chose to copy there using Build Automation).

**Note:** The Libs and Resources folder can be renamed to "AppName Libs" or "AppName

- Resources", where AppName is the name of your app. This can be useful should you want to package several apps into a single folder.

How you deploy a Linux application varies depending on the Linux distributions you support. Different distributions have different formats for packaging. Common installer formats are deb (used by Debian and Ubuntu) and RPM (RedHat Package Maker) used by RedHat.

For simple needs, you can also just zip the Linux application and Lib folder for distribution.

### Generic Installer

[InstallJammer](#) is an open-source product that has a simple user interface for creating an installer that works on a variety of Linux distributions. Unfortunately, this tool is no longer being actively developed, but it still works well and is a good choice if you do not use Linux often enough to master creating dedicated installers.

### Debian Installer

Debian installers are used by Debian-based Linux distributions, such as Ubuntu or Linux Mint. They can be installed by the Synaptic Package Maker or from the terminal. You create Debian installers using the dpkg-deb terminal application. Unfortunately, its usage is far more involved than can be covered here.

This tutorial describes how you can create a Debian package:

- [Debian Binary Package Building How To](#)

**Redhat Installer**

The Redhat installer format (RPM) is used by Redhat-based Linux distributions. You can create RPM installers using the rpmbuild terminal application. Unfortunately, its usage is far more involved than can be covered here.

The Fedora Project does have a good walkthrough, which is available here:

- [How to create an RPM package](#)

# Web Apps

Web applications have a graphical user interface that runs inside a web browser and a companion server application that runs on (or as) a web server.

## Table of Contents

- [App Object](#)
- [Design Considerations](#)
- [Build Settings](#)
- [Deployment](#)

## App Object

For a web project, the App object is a subclass of `WebApplication`. You use it to specify the default web page that opens when your application starts and a session connects, default messages, `HTMLHeader` and the application icon (in various sizes).

### Event Handlers

The `WebApplication` class for a web application has event handlers. They are:

- `Close`: The close event is called when the application quits.
- `HandleURL`: This event is called when the web app is accessed using a URL it cannot resolve.
- `HandleSpecialURL`: This event is called when the web app is accessed using the special URL, `/special/value`. In the `Request` parameter you can get information about the URL and return information back. This can be used to implement web services in your web applications. The URL `"/api/value"` is also supported and will result in this event handler being called.
- `Open`: This event handler is called when your application first starts.
- `UnhandledException`: Called when a runtime error occurs that is not caught by your code. This event gives you a "last chance" to catch runtime errors before they cause your application to quit.

### Properties

- `DefaultWebPage`: This is the web page that is displayed automatically when the application starts.
- `LaunchMessage`: On the application loading screen, a progress bar, the application icon and the launch message all appear while the application is being loaded.
- `DisconnectMessage`: This message appears when the client has lost its connection to the server.
- `HTMLHeader`: Can be used to add static HTML code to each page.
- `Icon`: This opens the Icon Editor and lets you specify the various size icons that are used by the application. These icons are used by the loading screen and as a the browser favicon.

## Design Considerations

### Client/Server

All web apps uses a client/server design. This means that a web app has two parts: the client user interface and the app that runs on the server.

The client is the part that the user interacts with. For a web app, this is the user interface that runs in a web browser. Web apps support recent versions of modern web browsers such as Safari, Chrome, Firefox and Internet Explorer.

The app is where your Xojo code runs and it is typically run on a server. The client and the web app components can reside on the same computer, but they are usually different computers: the user's computer for the web browser/user interface and a server that is running the app.

## Latency

When you are testing your web apps locally, both the client (web browser) and server (app) are running on your computer. This means that communication between them is quick. But when you deploy an app to a web server, there can be a longer delay as the web browser has to communicate across the Internet to the app on the server. This delay is called latency and has a lot of factors, including the speed of the user's Internet connection, general Internet congestion and the responsiveness of the server.

It is important when designing your app to keep latency in mind, by not designing your UI as if it expects fast (desktop-like) responsiveness.

## Multiple Users and Sessions

A web app typically has multiple users connected to it at one time. This is handled for you using a concept called Sessions. The Session object is a subclass of WebSession and is automatically added to all web projects. Each user that connects to your web application gets its own Session subclass that you can access using the Session global function.

Use the Session object to store information that is global to the connected session, but not to the overall app.

## Cookies

Cookies are a browser feature that allows you to save Session-specific data. Web apps have built-in support for Cookies as methods of the WebSession class.

Cookies are a standard way for web browsers to retain information pertaining to a session. For example, if your web app has users log in using a UserID, you could save the UserID as a cookie. The next time they open the web app, you can look for the cookie and if it is available, pre-fill the UserID name for them.

This is how you set a cookie value:

```
Session.Cookies.Set("UserName", "BobRoberts")
```

And this is how you retrieve a cookie value:

```
Dim userName As String  
userName = Session.Cookies.Value("UserName")
```

## Hash Tags

Hash tags are a way for you to identify different areas of the web application. Web applications always show the same browser URL, even as you navigate to different web pages. You can use the Hash Tag as a way to identify different pages (or data) so that the user can return directly to it later.

Hash tags are written using the “#” character like this:

```
http://www.mywebsite.com/#details
```

The `Session.HashTagChanged` event is called when the hash tag is changed by the user. In this event, you can get the new value and choose to navigate to a different page or display different data.

## Build Settings

The Build Settings section of the Navigator contains the build-specific settings for your application. You can check the box next to each target in order to build an application for that target.

### Shared

The Inspector for Shared settings contains these properties:

<b>Version</b>	
Major Version	<input type="text" value="1"/>
Minor Version	<input type="text" value="0"/>
Bug Version	<input type="text" value="0"/>
Stage Code	<input type="text" value="Development"/>
Non Release Version	<input type="text" value="0"/>
Auto Increment Version	<input type="checkbox"/> OFF
Short Version	<input type="text"/>
Long Version	<input type="text"/>
Package Info	<input type="text"/>
<b>Build</b>	
Use Builds Folder	<input checked="" type="checkbox"/> ON
Include Function Names	<input checked="" type="checkbox"/> ON
Language	<input type="text" value="Default"/>
Deployment Type	<input type="text" value="CGI"/>
Port	<input type="text" value="Choose Automatically"/>
Application Identifier	<input type="text" value="com.example.myapp"/>
<b>Debug</b>	
Command Line Arguments	<input type="text"/>
Destination	<input type="text" value="N/A"/> 
Debug Port	<input type="text" value="8080"/>

- **Major Version:** The Major version for your application. Version numbers are usually written as 1.2.3.4, where “1” is the major version.
- **Minor Version:** The Minor version for your application. Version numbers are usually written as 1.2.3.4, where “2” is the minor version.
- **Bug Version:** The Bug version for your application. Version numbers are usually written as 1.2.3.4, where “3” is the bug version.
- **Stage Code:** Used to indicate the type of application you are building (Development, Alpha, Beta, Final).
- **Non Release Version:** The Non Release (build) version for your application. Version numbers are usually written as 1.2.3.4, where “4” is the non release version.
- **Auto Increment Version Info:** When ON, the Non Release Version increases by one each time you do a Build.
- **Short Version:** A short text description for your application. Usually this contains just the version number (such as 1.2.3.4) and is displayed by some operating systems in Get Info or Property windows for the application.
- **Long Version:** A longer text description for your application. Usually this contains the application name, copyright, version and other information. This is displayed by some operating systems in Get Info or Property windows for the application.
- **Package Info:** A text description for your application that may be displayed by some operating systems in Get Info or Property windows for the application.
- **Use Builds Folder:** When ON, builds are placed in a separate folder alongside the project file. Each platform (OS X, Windows, Linux) also gets its own subfolder within this builds folder.
- **Include Function Names:** When ON, the actual names of your function calls are included in the built application. This is useful for debugging purposes and for getting stack traces.
- **Language:** Allows you to specify the language to use to resolve dynamic constants.
- **Deployment Type:** Web applications can be deployed as either CGI or stand-alone applications.
- **Port:** Select a specific port to use to connect to the web application or let the web application choose the port automatically.
- **Application Identifier:** Required for CGI applications, this is an unique identifier for the web application.
- **Command Line Arguments:** These are the command-line arguments that are passed to your application when running it in Debug mode.
- **Destination:** Specifies the path where the application is located when you run it in Debug mode. If not specified, then the application is placed alongside your project file.
- **Debug Port:** This is the port that is used when running your web applications in Debug mode. Note: If you ever have difficulty debugging your web applications, make sure the debug port is not in use by another application.

## Xojo Cloud

The Xojo Cloud section is used to deploy your web app to a Xojo Cloud server.

The screenshot shows a dialog box titled 'Build'. It has two input fields: 'Application Name' with the text 'MyApplication' and 'Server' with a dropdown menu showing 'XojoExamples' and a blue arrow icon.

- **Application Name:** The name of your web application. Your web app will be deployed into a folder with this name, which will be of the URL to the app.
- **Server:** The Xojo Cloud server to which to deploy. You will need to have purchased a Xojo Cloud server and be logged in to your account in Xojo in order to see your servers.

## OS X

The OS X section allows you to specify settings for the OS X web app.

- **Mac App Name:** The actual file name for the OS X app.
- **Architecture:** The CPU architecture for the app. Choices are x86 32-bit and x86 64-bit. Currently only 32-bit apps can be debugged.
- **Bundle Identifier:** This is the same as the Application Identifier.

## Windows

The Windows section contains build settings for your Windows web apps.

- **Windows App Name:** The actual file name for the Windows application.
- **Company Name:** On Windows 7, the “Company Name” appears in the Copyright section of the application properties in the Details tab.
- **Product Name:** The name of the product as installed in the Windows Start → All Programs menu. This also appears in “Product name” on the application properties Details tab.
- **Internal Name:** This is useful when your product has a different internal name than its external name. This is not shown on the app properties Details tab.
- **File Description:**
- **Architecture:** The CPU architecture for the app. Choices are x86 32-bit and x86 64-bit. Currently only 32-bit apps can be debugged.

## Linux

The Linux section contains build settings for your Linux web apps.

- **Linux App Name:** The actual file name for the Linux application.
- **Architecture:** The CPU architecture for the app. Choices are x86 32-bit and x86 64-bit. Currently only 32-bit apps can be debugged.

## Deployment

The easiest way to deploy your Xojo Web apps is to use Xojo Cloud. With Xojo Cloud, you can deploy your web app in a single step by clicking the Deploy button on the main toolbar. Xojo Cloud also adds many additional features which are covered in the Xojo Cloud section of the Web App Deployment chapter.

Should you prefer to deploy to your own servers, you can also do so as either a standalone app or a CGI app that works in conjunction with an existing web server (usually Apache).

For more information about web app deployment:

- [Web Deployment Overview](#)
- [Linux Deployment](#)
- [Web Deployment](#)
- [IIS Deployment](#)

# Console Apps

Unlike iOS, desktop and web apps, console apps are not event-based. Console apps have no graphical user interface and are designed to run from the terminal, command line or as a background process.

## Table of Contents

- [App Object](#)
  - [Event Handlers](#)
- [Background Applications](#)
  - [Daemon](#)
  - [Service Application](#)
- [Build Settings](#)
  - [Shared](#)
  - [OS X](#)
  - [Windows](#)
  - [Linux](#)
  - [This Computer](#)
- [Deployment](#)
  - [OS X](#)
  - [Windows](#)
  - [Linux](#)

## App Object

For a console project, the App object is a subclass of `ConsoleApplication`, but you can also choose to change the Super to `ServiceApplication` to create a Windows Service.

### Event Handlers

The Application class for a console application has these event handlers:

- **Run:** This event handler is called when your app first starts, either by running the app from Xojo or by running a built app. Your app quits when the Run event ends or you call the `Quit` method.
- **UnhandledException:** Executes when a runtime error occurs that is not caught by your code. This event gives you a “last chance” to catch runtime errors before they cause your app to terminate.

## Background Applications

Console applications are often used to create background applications. On OS X and Linux, background applications are called daemons. On Windows, background applications are called services.

### Daemon

To turn your console application into a daemon that can run in the background, simply call the `Daemonize` method of the `ConsoleApplication` class.

You can also use the `Daemonize` method on OS X, but Apple would rather you use `launchd` to start daemon

processes.

## Service Application

To create a console application that can run as a Windows Service you need to use the `ServiceApplication` class. When you create your console application, change the Super of the App from `ConsoleApplication` to `ServiceApplication`. Alternatively you can choose `ServiceApplication` from the Templates in the Project Chooser.

A service application gives you additional events that help you manage things when the services starts, stops or is paused. Refer to `ServiceApplication` in the Reference Guide for more information about these events.

## Build Settings

The Build Settings section of the Navigator contains the build-specific settings for your application. You can check the box next to each target in order to build an application for that target.

### Shared

The Inspector for Shared settings contains these properties:

- **Major Version:** The Major version for your application. Version numbers are usually written as 1.2.3.4, where “1” is the major version.
- **Minor Version:** The Minor version for your application. Version numbers are usually written as 1.2.3.4, where “2” is the minor version.
- **Bug Version:** The Bug version for your application. Version numbers are usually written as 1.2.3.4, where “3” is the bug version.
- **Stage Code:** Used to indicate the type of application you are building (Development, Alpha, Beta, Final).
- **Non Release Version:** The Non Release (build) version for your application. Version numbers are usually written as 1.2.3.4, where “4” is the non release version.
- **Auto Increment Version Info:** When ON, the Non Release Version increases by one each time you do a Build.
- **Short Version:** A short text description for your application. Usually this contains just the version number (such as 1.2.3.4) and is displayed by some operating systems in Get Info or Property windows for the application.
- **Long Version:** A longer text description for your application. Usually this contains the application name, copyright, version and other information. This is displayed by some operating systems in Get Info or Property windows for the application.
- **Package Info:** A text description for your application that may be displayed by some operating systems in Get Info or Property windows for the application.
- **Use Builds Folder:** When ON, a separate folder is created alongside the project file. Each platform (OS X, Windows, Linux) also gets its own subfolder within this builds folder.
- **Include Function Names:** When ON, the actual names of your function calls are included in the built application. This is useful for debugging purposes and for getting stack traces.
- **Command Line Arguments:** These are the command-line arguments that are passed to your application when running it from Xojo.
- **Destination:** Specifies the path where the running application is placed. Normally it is placed alongside your project, but you can choose a different location.

### OS X

The OS X section allows you to specify settings for the OS X applications.

- Mac App Name: The actual file name for your OS X application.
- File Types: n/a
- Architecture: The CPU architecture for the app. Choices are x86 32-bit and x86 64-bit.
- Bundle Identifier: The bundle identifier is used by OS X as a unique descriptor for your application. It is usually specified as a reverse domain name, such as com.xojo.myapplication. A bundle identifier is required for OS X applications.

## Windows

The Windows section contains build settings for your Windows applications.

- Windows App Name: The actual file name for the Windows application.
- Company Name: On Windows 7, the “Company Name” appears in the Copyright section of the application properties in the Details tab.
- Product Name: The name of the product as installed in the Windows Start → All Programs menu. This also appears in “Product name” on the application properties Details tab.
- Internal Name: This is useful when your product has a different internal name than its external name. This is not shown on the application properties Details tab.
- Architecture: The CPU architecture for the app. Choices are x86 32-bit and x86 64-bit.

## Linux

The Linux section contains build settings for your Linux applications.

- Linux App Name: The actual file name for the Linux application.
- Architecture: The CPU architecture for the app. Choices are ARM 32-bit (aka Raspberry Pi), x86 32-bit and x86 64-bit.

## This Computer

The **This Computer** section shows you the appropriate section (OS X, Windows or Linux) for the platform you are currently using. For example, if you are running Xojo on OS X, then This Computer shows the OS X settings.

# Deployment

Console applications are deployed by installing them on a computer and running them on the computer.

Console applications have two parts: the main executable and its libraries (contained in the Libs folder). Both parts are required in order for the application to run.

You can just zip the console application and its Lib folder for distribution.

## OS X

On OS X, you can start a console app from the Terminal. Navigate to the folder containing the app and type its name at the Terminal prompt. You will need to include the prefix `./` in order for the Terminal to actually find the app in the current folder:

```
./MyApp
```

---

## Windows

On Windows, you can start a console app from the Command tool. Just type its name at the command prompt:

```
MyApp
```

## Linux

On Linux, you can start a console app from the Terminal. Navigate to the folder containing the app and type its name at the Terminal prompt. You will need to include the prefix "./" in order for the Terminal to actually find the app in the current folder:

```
./MyApp
```

## Compilation Constants

Before you build your application, you choose the platform or platforms on which it will run. You build applications for Windows, Linux, and OS X as desktop, console or web. Additionally, you can build for iOS. While programming, you can selectively enable or disable segments of your code that are valid only for particular platforms by using compilation constants.

## Compilation Constants

The compilation constants tell you information about the platform on which the application is running. When used with conditional compilation you can specify blocks of source code to include or exclude for specific platforms.

These are the some of the compilation constants:

- `DebugBuild`: True when the app is running in Debug mode
- `TargetWin32`: True when the app is running on Windows. Because Xojo uses the Win32 API, this is True regardless of whether Windows itself is 32-bit or 64-bit.
- `TargetMacOS`, `TargetCocoa`: True when running on OS X.
- `TargetLinux`: True when the app is running on Linux. This is True regardless of whether Linux itself is 32-bit or 64-bit.
- `Target32Bit`: True when running on a 32-bit OS.
- `Target64Bit`: True when running on a 64-bit OS.
- `TargetDesktop`: True for desktop apps.
- `TargetWeb`: True for web apps.
- `TargetiOS`: True for iOS apps.
- `TargetConsole`: True for console apps.

For the complete list, refer to [Compiler Constants](#).

These constants are automatically set to either True or False depending on the platform being compiled. You can use conditional compilation commands to use specific code like this:

```
' Saving Preferences
#If TargetWin32 Then
  ' Use Registry
#ElseIf TargetMacOS Then
  ' Use plist
#ElseIf TargetLinux Then
  ' Use XML
#ElseIf TargetiOS Then
  ' Use JSON
#EndIf
```

## Versions

You can also check the version of Xojo being used with the `XojoVersion` and `XojoVersionString` constants. These two constants can be used to block out code that is not compatible with older (or newer versions) of Xojo.

# About Debugging

## Table of Contents

- [Types of Bugs](#)
- [Analyzing the Project](#)

What is debugging? Debugging means removing errors, both logical and syntactical, from your programming code. Errors in programming code are referred to as “bugs.” You are probably wondering why errors are called “bugs.” Well, back in the 1940’s, the United States Navy had a computer that occupied an entire warehouse. At that time, computers used vacuum tubes and the light from the tubes attracted moths. These moths would get inside the computer and short out the tubes. Technicians would have to go in and remove the bugs to make the computer work again. Since this was a government project, everything had to be logged, so they would put down “debugging computer” in the log. But enough of the history lesson.

Debugging is a normal part of programming. It’s the part of programming most programmers like the least. Fortunately, the Debugger makes it easy to track down those nasty bugs and squash them like a, well, bug.

## Types of Bugs

There are two types of bugs you can make in your code: logical and syntactical.

### Logical Bugs

These are bugs in your programming logic. You will know you have found one of these when your code compiles but does not produce the results you were expecting. The debugger can help you find these types of bugs by letting you watch your code execute one line at a time.

These are by far the most common source of bugs.

### Syntactical Bugs

These are bugs where you have mistyped the name of a keyword, class, property, variable, or method. These are often called Syntax Errors. You may have also tried to use two values together that don’t go together. For example, if you try to assign a String value to a variable or property of type Integer, you will get a Type Mismatch error because they are different data types.

These type of bugs are caught by the compiler, so you will never be able to even run your project if you have a syntax error.

## Analyzing the Project

Since a project cannot be compiled if it contains syntax errors, you have the option of analyzing the project as a preliminary step. Choose Project → Analyze Project or Project → Analyze Item, where Item is the current item in the IDE window. Analyze Project checks for issues but does not build the project. Some issues that it identifies are syntax errors, unused local variables and parameters, and type conversion issues.

Errors are indicated by a stop sign icon and will prevent you from being able to run or build your project. Warnings are indicated by a yellow warning triangle and do not prevent you from building or running your project.

If errors or warnings are found they are listed in the Errors panel at the bottom of the Workspace. Expand each row that is displayed to see the individual issues.

You can click on each issue and you will display the appropriate editor (usually the Code Editor) with the issue highlighted.

You should fix any errors that are reported. And you should evaluate the warnings to see if you think they need to be fixed. Not all warnings need to be fixed.

The Type and Location buttons in the Issues pane allow you to change how to view the Issues. The Type button groups the issues by the type of the issue. So all the “Unused local variables” issues would appear together.

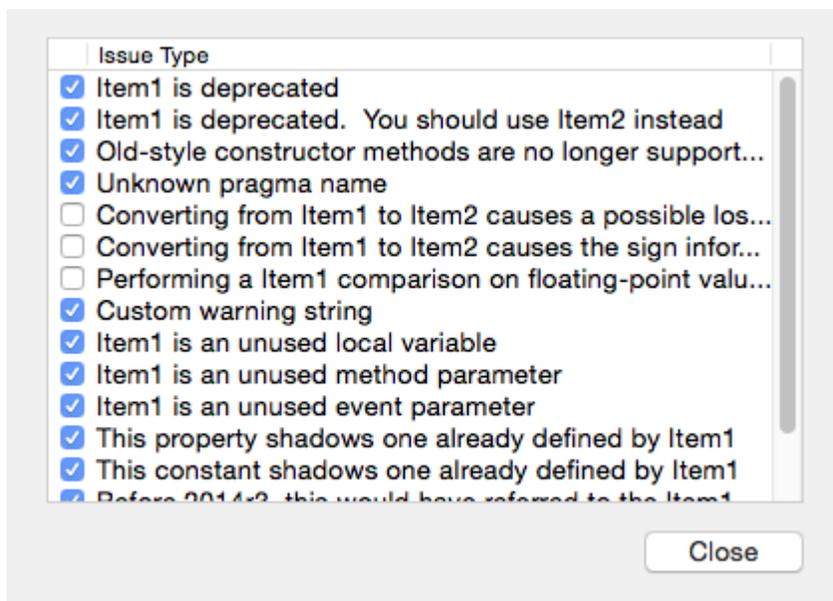
The Location button groups the issues by the object in which they occur. So all the issues for Window1 would appear together, for example.

## Analyze Item

The Analyze Item command, available from the Project menu or on the Code Editor toolbar, works the same as Analyze Project but instead of analyzing the entire project, it only analyzes the code in the Code Editor.

## Filtering Types of Warnings

You can control the types of warnings that the Errors panel displays. Choose Analysis Warnings from the Project menu to display a window that lists all the types of warnings that can be found. Only the selected warnings are reported when you analyze. Unselect any warnings that you do not wish to know about.



## Warnings

These are the available warnings:

- Item1 is deprecated
- Item1 is deprecated. You should use Item2 instead
- Old-style constructor methods are no longer supported. You should use "Constructor" instead
- Unknown pragma name
- Converting from Item1 to Item2 causes a possible loss of precision, which can lead to unexpected results
- Converting from Item1 to Item2 causes the sign information to be lost, which can lead to unexpected results
- Performing a Item1 comparison on floating-point values can yield unexpected results due to their inexact binary representation.
- Custom warning string
- Item1 is an unused local variable
- Item1 is an unused method parameter
- Item1 is an unused event parameter
- This property shadows one already defined by Item1
- This constant shadows one already defined by Item1
- Before 2014r3, this would have referred to the Item1, but now it refers to the Item2.
- Using method Item1 from the 'Item2' library on this target is suspicious.

# Using the Debugger

When you run your project from within Xojo, you are using the debugger. If there are no errors after compiling and building your application, it launches and the Workspace switches to the debugger tab. This is referred to as running the project in “debug mode”.

While you are using your application you will not normally see the debugger, but there are several ways it can be activated.

## Table of Contents

- Activating the Debugger
  - Manual Activation
  - App Error (Exception)
  - Breakpoint
  - Conditional Breakpoint
- The Debugger Screen
  - Watching Variables
  - Viewing Source Code
  - Viewing the Call Stack
  - Viewing Threads

## Activating the Debugger

There are several ways the debugger can become active: you can manually activate it, your app can raise an exception, you can set a breakpoint or call the Break method.

```

Private Sub LoadNotes()
    NotesList.DeleteAllRows
    Dim categoryText() As String = Array("Idea", "Task", "Reminder")
    For Each n As Note In App.Notes
        NotesList.AddRow(n.Title, categoryText(n.Category))
        NotesList.RowTag(NotesList.LastIndex) = n
    Next
End Sub

```

Stack		Variables	
Main Thread		Variables	
NotesWindow.LoadNotes		Name	Value
NotesWindow.Event.Open			<a href="#">Globals</a>
		categoryText	Nil
		self	<a href="#">NotesWindow.NotesWindow</a>

## Manual Activation

You can manually activate the debugger by clicking the Pause button on the Debugger toolbar while your app is running. If your code is currently running, then this displays the method that is running and highlights the next line of code that will execute.

If your app was idle, then this drops you into the Event Loop where you can view global objects and their variables (such as App, Runtime and modules).

## App Error (Exception)

Should your app raise an exception while you are running in debug mode, the debugger becomes active (called a Runtime Error). The source code where the exception was raised is displayed.

 This only occurs if you have Project → Break On Exceptions checked. When checked, the debugger is activated for all exceptions that are raised, even one that are caught later by a Catch statement.

From here you are able to view variable values and review the call stack (see below).

## Breakpoint

In your code you can set breakpoints. A breakpoint is an indicator that tells the debugger to activate itself when the line of code is reached.

For example, you might want to set a breakpoint at the start of a method if you want to carefully review the code as it executes.

```
LoadNotes
|
| NotesList.DeleteAllRows
|
| ● Dim categoryText() As String = Array("Idea", "Task", "Reminder")
|   For Each n As Note In App.Notes
|     NotesList.AddRow(n.Title, categoryText(n.Category))
|     NotesList.RowTag(NotesList.LastIndex) = n
|   Next
```

To set a breakpoint, you click on the “dashes” that appear in the gutter of the Code Editor. Each dash indicates a line of code that can have a breakpoint set. You can also set a breakpoint using the Project → Breakpoint → Turn On menu (⌘+) on OS X, Ctrl+\ on Windows and Linux). The same command turns off a previously set breakpoint on the line.

If you want to turn off all breakpoints throughout your project, use the Project → Breakpoint → Clear All menu. To see all breakpoints in your project, use Project → Breakpoint → Show All menu to display the breakpoints in the Find panel.

## Conditional Breakpoint

There will be times where you may want to stop at a breakpoint, but only if a specific condition occurs. For example, you could be in a long loop and you only want to stop at the 75th element.

You set up a condition breakpoint in your source code using a combination of a conditional If statement and the Break command. This example will stop at the breakpoint when the loop counter reaches 75:

```
For i As Integer = 1 To 100
  If i = 75 Then Break
Next
```

The Break command is called when *i* reaches 75 and activates the debugger.

 The Break command does not do anything in a built application. It is only useful when running your app in Debug Mode.

# The Debugger Screen

Once you are in the debugger, you can control it using the toolbar commands: Pause/Resume, Stop, Step, Step In, Step Out and Edit Code.

- **Pause/Resume:** Use Pause to pause a running application and activate the debugger. If you are in the debugger, the Resume button tells your application to continue running where it left off. You can also click the Run button on the main toolbar to Resume, select Project → Resume from the menu or you can use the ⌘+R (Ctrl-R on Windows and Linux) shortcut.
- **Stop:** The Stop button immediately stops the running application. The application is quit immediately and no further code is run. You can also use the shortcut Shift+⌘+R (Shift-Ctrl-R on Windows and Linux)
- **Step:** The Step button is used to run the code one line at a time. Each time you click Step, the highlighted code is executed and you remain in the debugger. If you click Step while on a method call, the method is called and you move to the next line of code. You will mostly use Step while debugging. In addition to clicking the button, you can use Project → Step → Step Over menu command or the shortcut Shift+⌘+O (Shift-Ctrl-O on Windows and Linux).
- **Step In:** The Step In button works like the Step button except when you reach a method call. Instead of calling the method and moving to the next line of code, Step In moves you to the first line of code in the method. In addition to clicking the button, you can use Project → Step → Step Into menu command or the shortcut Shift+⌘+I (Shift-Ctrl-I on Windows and Linux).
- **Step Out:** If you are in a method, clicking Step Out, runs the rest of the code in the method and then stops when the method returns. In addition to clicking the button, you can use Project → Step → Step Into menu command or the shortcut Shift+⌘+T (Shift-Ctrl-T on Windows and Linux).
- **Edit Code:** The Edit Code button allows you to jump to the Code Editor for the current method that is in the debugger. Here you can edit the code (which you can not do in the code display of the debugger). However, changes that you make to code in the Code Editor are not reflected in your application until the next time you run.

## Watching Variables

The watch pane of the debugger allows you to view the values of variables. By default it shows you the currently declared variables for the method that is executing.

You can change the values of booleans, integers, doubles and strings by clicking on the value (or selecting the pencil icon on the far right), entering a new value and pressing Return.

 Note: For booleans, anything entered besides True or true is treated as False.

With strings, the pencil icon on the right changes to show a magnifying glass. Clicking this displays the string editor which lets you view the string as text or as binary. It also allows you to make changes to the string using a larger editing area.

For integers, you can change the display format by right-clicking on the value and selecting View As Decimal, Hex, Binary or Octal.

For doubles, you can change the display format by right-clicking on the value and selecting View As Scientific, Decimal or Rounded.

If the variable type is a class, then you can click on the type to display the property values.

At the top of the variables pane is a Popup Menu that allows you to navigate the variable hierarchy. You can use it

to quickly jump back to the method variables after having displayed the string editor or class instance details.

On some objects (such as Windows and RecordSets), there is a special link at the top called “Contents”. Clicking this displays information about the object, such as the controls on a window or the fields in a RecordSet.

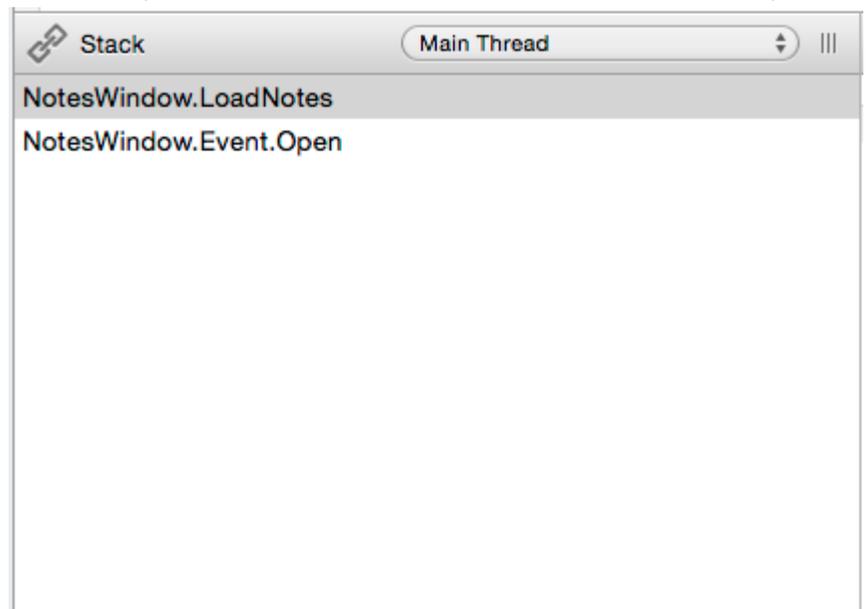
## Viewing Source Code

The source code for the current method (or event) displays in the debugger code viewer. The next line to execute is highlighted. In the code viewer, you can set additional breakpoints and watch the highlight move as you step through code.

If you want to edit the code, click the Edit Code button on the toolbar. Changes made to code do not take effect until the next time you run your project.

## Viewing the Call Stack

The call stack displays the calling hierarchy of your methods. If your main window Open event calls a method named Initialize then the call stack shows Initialize at the top (since it is the current method) with Window1.Open underneath it.



The call stack is very helpful for viewing your code path and for figuring out the methods that call other methods.

## Viewing Threads

Each thread in your application is tracked separately in the debugger.

Use the selector in the Stack section to see and show the threads that are currently running.

# Exception Handling

The debugger can help you verify that your code is working as you expect and it can help you find errors. Once you've found the source of errors, you want to make sure you handle them properly.

Exceptions are a type of error that occur when something unexpected happens. These errors will crash your application if you do not handle them in some way. The act of causing an exception to occur is called raising an exception.

All exceptions are subclasses of the `RuntimeException` class.

When an exception is encountered in your code, you can choose to have the Debugger displayed at the line causing the exception. To do this, select `Project → Break On Exceptions` in the menu so that it has a checkmark next to it.

## Table of Contents

- [NilObject Exceptions](#)
- [Try...Catch](#)
- [Exception](#)
- [App.UnhandledException](#)
  - [Web Apps](#)
  - [iOS Apps](#)
- [Creating Your Own Exceptions](#)

## NilObject Exceptions

The most common type of exception that occurs is a `NilObjectException`. These errors occur when you attempt to use an object but don't have an instance of it.

The simplest example of this is forgetting to use `New` to get an instance before you try to use a property or call a method:

```
Dim d As Xojo.Core.Dictionary
d.Value("ID") = "Test" ' NilObjectException
```

The correct way to write this code is with the `New` command:

```
Dim d As New Xojo.Core.Dictionary
d.Value("ID") = "Test" ' NilObjectException
```

Another place to check for this is in the return value of methods that return an instance. Some methods return `Nil` in certain situations, such as `GetOpenFolderItem` which returns `Nil` if the user clicks `Cancel`:

```
Dim f As FolderItem
f = GetOpenFolderItem("")
If f.Exists Then ' NilObjectException
    ' Do something
End If
```

To prevent these types of errors, you should always check if the value is Nil before you attempt to access it:

```
Dim f As FolderItem
f = GetOpenFolderItem("")
If f <> Nil Then
    If f.Exists Then
        ' Do something
    End If
End If
```

## Try Catch

Sometimes you may find that an exception is a normal part of processing. What you want to do in these situations is catch the exception so that you can deal with it appropriately. The `Try...Catch` command is used for this purpose.

For example, loading an XML file can raise an `XMLException` if the file is not valid XML. You can check for this by using a `Try..Catch` block:

```
Dim xmlFile As FolderItem
xmlFile = GetOpenFolderItem("")
If xmlFile <> Nil Then
    Try
        Dim xml As New XmlDocument
        xml.LoadXml(xmlFile)
    Catch e As XmlException
        MsgBox("Not an XML file!")
        Return
    End Try
End If
' Process the XML
```

If no `XMLException` is raised then the code in the `Try` section runs. If an `XMLException` is raised, then the code in the `Try` block stops running and the code in the `Catch` block is run.

## Exception

The `Exception` command is a simplified version of `Try..Catch`. Rather than focusing on a specific block of code, `Exception` catches errors for the entire method.

The `Exception` command goes at the end of the method and is called if an exception is generated anywhere in the method. The preceding example could be written like this using the `Exception` command:

```
Dim xmlFile As FolderItem
xmlFile = GetOpenFolderItem("")
If xmlFile <> Nil Then
    Dim xml As New XmlDocument
    xml.LoadXml(xmlFile)
End If
```

```
End If
' Process the XML

Exception e As XmlException
    MsgBox("Not an XML file!")
    Return
```

Although this is simpler, it is not as obvious what is occurring (especially if there is even more code in the method) and it is also far less flexible since it only works on the entire method..

## App.UnhandledException

There is a special event handler in the App object of your project that is called if an unhandled exception is not caught.

The event has one parameter, `error`, which is the exception itself. You can put code in this event handler to display a message to the user, capture log information or anything else you want. Return `True` to hide the default unhandled exception error dialog and attempt to allow you application to continue (which is usually not recommended).

This code displays a friendlier message to the user and then quits the application:

```
Dim msg As String = "An error occurred. Please notify the author."
MsgBox(msg)

Quit
Return True
```

You can also display the runtime stack so it can be used to help pinpoint the location of the problem so you can fix it:

```
Dim msg As String = "An error occurred. Please notify the author. Stack: "
MsgBox(msg + Join(error.Stack, EndOfLine))

Quit
Return True
```

## Web Apps

The default error dialog displays information about the error and provides a field for users to add additional information. This information is written to an `errors.log` file alongside the web application. You can also get the information in the `WebSession.JavaScriptError` event handler so that you can log it yourself.

Remember, if this event handler has been called and you return `False` (the default) it means your entire web application is going to stop running. Return `True` to hide the default error dialog and prevent the web app from quitting.

## iOS Apps

iOS apps do not display an error dialog by default; the app just terminates. You'll need to add your own code to display unhandled exceptions.

## Creating Your Own Exceptions

Since `RuntimeException` is a class like any other, you can subclass it to create your own exceptions. This allows you to raise your own exceptions that you have defined.

If you create a `RuntimeException` subclass called `InvalidXMLFormatException` then you can raise it using this syntax:

```
If incorrectXml文件格式 Then
  Raise New InvalidXMLFormatException
End If
```

# Remote Debugging

When you run your project it runs on the same platform you are using. So if you develop on OS X, then clicking Run will run your project on OS X.

Since Xojo is a cross-platform development tool, you are likely to want to also run your projects on other platforms for debugging purposes. You can do this using the Remote Debugger, which has two version: Desktop and Console.

The desktop version is used to remotely debug to an OS that has a desktop UI. It can be used with desktop and console apps. The console version is used to remotely debug to an OS that does not have a UI (such as servers). It can debug console and web applications.

## Table of Contents

- [Remote Debugger Configuration](#)
  - [Remote Debugger Desktop](#)
  - [Remote Debugger Console](#)
- [Development Machine Configuration](#)
- [Remote Debugging](#)

## Remote Debugger Configuration

The Remote Debugger is a small app that lets you run your project on a different target platform. Xojo communicates with it so that when you run your project, it sends it to the target platform rather than running it locally. Normally when you run your project from within Xojo (Project → Run), it runs the debug build locally. If you have the Remote Debugger configured, you can instead have Xojo send and run your debug build on a remote computer, which is incredibly useful for testing cross-platform applications. This remote build still communicates with Xojo, so you can access the debugger and test just as if you were running it locally.

### Remote Debugger Desktop

On the remote machine, you need to run Remote Debugger Desktop, which is included with your Xojo installation. There is a separate version for each platform (Windows, OS X, Linux). Copy the version you need to the platform you are using.

Remote Debugger Desktop has an Options window that lets you configure some settings. In the General tab, you want to give the machine a name (so you can identify it from Xojo) and choose a Download location for the transferred apps. You typically do not need to change the settings in the Networking tab.

After you have set the Options, click OK and leave Remote Debugger Desktop running.



If you are using a firewall on the remote machine, you need to make sure that **port 44553** is open for both UDP and TCP connections.

### Remote Debugger Console

The console version can debug console applications, standalone web applications and CGI web applications on

remote machines that do not have a desktop interface (such as servers). Currently it can only be used across a local network (or VPN).

If you are trying to debug a CGI app, set the Remote Debugger to not launch automatically (in the Options) and set the path to point to where your CGI apps must be installed in order for them to work with the web server. The web server will launch the app when you visit the appropriate URL in your browser.

Remote Debugger Console runs from Terminal or the command line. The first time you launch it, you are prompted for the settings:

- Machine Name
- Download Directory
- IP Address: Specifies the IP address to listen on (must match an IP address available on the computer).
- Maximum Connections: Sets the maximum number of connections.
- Auto Launch: Have the remote debugger automatically launch the app being debugged.
- Public: Indicates if the remote debugger is publicly visible.
- Password: Specifies a connection password.

These settings are saved in the RDS.config file in the same folder as the Console Remote Debugger.

You can also provide these and other options via the command line. Use the "--help" argument to get a list of available command line options.

## Development Machine Configuration

On the development machine, you need to configure Xojo so that it can see the Remote Debugger. In Preferences, select Debugger. There you'll see a list of configured remote machines.

Click Add to add your remote machine. Depending on your network configuration, the remote machine may appear as an "auto-discovered remote machine". If it does, you can just click its name and then OK to add it. If it does not appear, you can enter the IP address (specified on the Remote Debugger Stub on the remote machine) and give it a name.

If you are using a firewall on your development machine, you need to make sure that port 44553 is open for UDP and TCP connections and port 13897 is open for TCP connections.

Note: Virtual machine software such as VMware Fusion, Parallels Desktop, VMware Desktop and Microsoft Virtual PC all work great with remote debugging, but you typically want to have their networking set up to use "Bridged Networking".

## Remote Debugging

Now you can try running a project remotely. On the development machine, create a new desktop project.

To run the project remotely, instead of selecting Project → Run (or clicking the Run button on the toolbar), choose Project → Run Remotely and click your remote machine name. Your project is compiled and linked as usual, but you will now see an additional step where this debug build is sent over to the Remote Debugger Stub on the remote machine.

When the Remote Debugger Stub has received the debug build, it runs it. Interact with it on the remote machine. Any breakpoints you have set will jump to the debugger on the development machine.

# Profile Code for Performance Tuning

The Code Profiler is used to track the length of time each method in your application takes to run. This information allows you to focus on performance optimizing the parts of your application that might be considered slow.

When profiling for performance, remember the quote by famous computer scientist Donald Knuth:

*"The real problem is that programmers have spent far too much time worrying about efficiency in the wrong places and at the wrong times; premature optimization is the root of all evil (or at least most of it) in programming."*

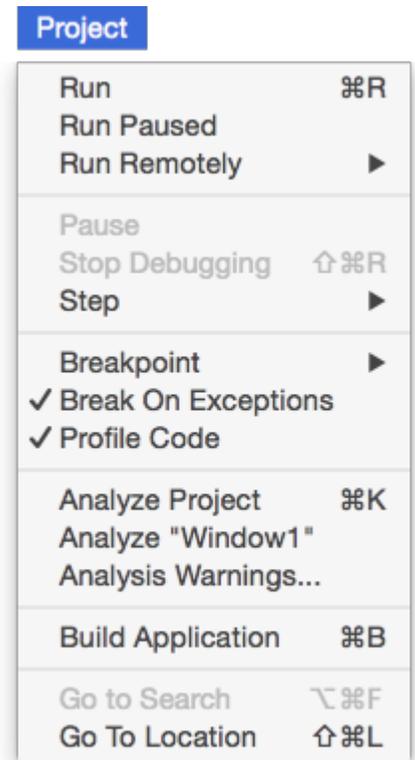
This simply means that you should not worry about performance until you have a reason to worry about performance. The good news is that when you are ready to worry about performance, the Xojo Code Profiler provides an easy way to identify the areas of your code that take longer to run.

## Table of Contents

- [Using the Code Profiler](#)
- [Using the Profiler with Built Apps](#)
- [Performance Analysis](#)
  - [High Number of Calls](#)
  - [High Total Time](#)
  - [High Average Time](#)
- [Optimization Techniques](#)
  - [Remove Unnecessary Loop Calculations](#)
  - [Invisible ListBoxes](#)
  - [Call Fewer Methods](#)
  - [Dim Outside Loops](#)
  - [Loop Analysis](#)
  - [Use a Different Algorithm or Technique](#)

## Using the Code Profiler

You can enable or disable the Profiler using the Project → Profile Code menu. A check mark appears next to the menu when profiling is enabled. After enabling Profile Code, you run and test your apps as you normally would. The Code Profiler silently gathers information about the methods that are called and how long each one takes to execute. Be aware that your application runs slower than usual when profiling is enabled due to the overhead of tracking and timing everything to get the profiler data.



When you quit your app, the Profiles section appears in the Navigator with an entry for the profile data.



Each time you run your project, the profile results appear in this Navigator section as a separate item. This allows you to compare current results to prior results. You can remove profile data by right-clicking on it in the Navigator and selecting Delete from the menu. Click on a Profile Data in the Navigator to display the Profile Results list.

Name	Called	Total (milliseconds)	Average (milliseconds)
Main Thread ChessBoardContainer.ChessBoard.Paint	2	1.0000	0.5000
Main Thread MainWindow.CancelClose	1	0.0000	0.0000
▶ Main Thread MainWindow.ChessBoardUpdateTimer.Action	172	47983.0000	278.9709
▶ Main Thread MainWindow.Open	1	2.0000	2.0000
▶ Main Thread MainWindow.SolveButton.Action	1	23.0000	23.0000
▶ Thread(24374128) EightQueensThread.Run	1	50657.0000	50657.0000

This is a list of all the methods that were called while you were using your app. It shows the number of times each method was called and how much time was spent in the method.

- **Name:** The class name and name of method. The thread is also displayed if your app uses threads.
- **Called:** The total number of times the method was called.
- **Total:** The total time (in milliseconds) that was spent in the method.
- **Average:** The average time (in milliseconds) that was spend in the method. This is Total / Called.

You can save the profiler data to a text file using the contextual menu. Simply right-click anywhere in the summary and choose “Save As...”.

You can expand methods to see the other methods that they called.

- When a method is collapsed, the time shown represents the time used by the method and any methods that

were called as a result of that method running.

- When a method is expanded, the time shown for the method represents the time used only by the method itself. Any called methods are shown below it and have their own time values.

As you are optimizing your application, you can compare the current Profiler Data with previous Profiler Data to see if your changes are improving performance.

 Profiler data is not saved along with your project. Use the Save As function described earlier if you wish to retain the profile data.

 Profiler data is only collected if your application quits normally. If it quits because of a crash (or you press the Stop button in the debugger), no profiler data is collected. For web apps you can simply close all the browser windows/tabs and wait for the IDE to detect that the app has terminated or you can add a button or action that calls the Quit method.

 Profiler data cannot be collected for OS X apps that have been sandboxed.

 The Code Profiler does not work with the **Remote Debugger**. If you need to use the Profiler to gather information about your app running on another platform, just install Xojo on the platform, copy your project over to it and run it from there with Profile Code enabled.

 The Code Profiler does not work with iOS projects.

## Using the Profiler with Built Apps

You can choose to create a standalone build of your application with embedded profiling code. You can then send this app to the users so that they can help generate profile data for you to analyze. To do this, just build your application with Profile Code selected in the Project menu. When you build with Profile Code enabled, a dialog appears to remind you that profiling code will be embedded in the application.

With this profiling code embedded, your application will log profiling data as it is being used. When it quits, the profile data is saved to a file called Profile.txt in the same folder as the application. Your user can then send you this file for you to analyze.

To view saved profiler data, take advantage of the open-source [Profile-Reader](#) project.

## Performance Analysis

When reviewing the results, each column tells you useful information, but not all of this information indicates you have a performance problem. Here are some suggestions of things to look out for.

### High Number of Calls

You should evaluate methods that have a high number of calls. Try to determine if the method needs to be called so often. Simply reducing the number of times a method gets called can improve performance.

For example, perhaps you have a Refresh method that is called often. In reviewing this, you may realize that the Refresh method is calling other methods unnecessarily and can perhaps be simplified. Or perhaps you'll notice that

Refresh is being called more often than necessary.

## High Total Time

It is worth reviewing methods that have a high Total Time. Now it could be that the Total Time is high because the method is called frequently. But this still warrants a review. It could be that the time is high because the method take a long time to complete, but this will mean it also has a high Average Time.

Regardless, these methods are worth reviewing to see if they can be improved.

## High Average Time

Methods with high Average Time can quickly degrade performance if they are called often. Average Time is simply Total Time / Calls, so you may already be reviewing these methods. Often, a simple improvement to a method that is called frequently can have a dramatic effect on the entire app.

# Optimization Techniques

To re-iterate, you should not spend your time optimizing code unless there is a clear case that it needs to be optimized. Sometimes it is just more pragmatic to move long-running code to a thread so that the user does not notice how long it takes. There may also be situations where the time spent optimizing the code does not yield a significant enough improvement to warrant the time spent on it.

But if you do run into situation where your code is running more slowly than needed, these these techniques may help.

## Remove Unnecessary Loop Calculations

If you have a calculation that occurs on a loop condition, then the calculation is done each time through the loop. Sometimes this may be necessary because the value could change. But often, this value does not change (it is what is called invariant -- it never varies), so you can instead save the value in a variable so that it does not get recalculated each time.

In this example, the tax rate is calculated each time through the loop. But the tax rate does not change while the loop is running, so the calculation is performed unnecessarily:

```
While (billAmount * TaxRate(state)) < 100
  AddLineItemToBill
Wend
```

Instead, you can save the tax rate:

```
Dim taxRate As Double = TaxRate(state)
While (billAmount * taxRate) < 100
  AddLineItemToBill
Wend
```

## Invisible ListBoxes

Populating ListBoxes with data is a common task. If you are filling a ListBox with lots of data (which itself may not be a great idea), you can first make the ListBox invisible before you add the rows and then make it visible after the rows have been added. This can prevent some unnecessary possible screen refreshes and resulting event calls, providing an overall speed improvement.

## Call Fewer Methods

Each method call can take a small amount of time overhead. In a loop, you may find that just putting the code directly into the loop, rather than its own method will result in a speed improvement. This should be done judiciously as it can quickly lead to unreadable code.

You might also consider using framework methods that can do multiple things in one method call, rather than multiple method calls. For example, with a ListBox using AddRow to add a blank row and then filling its columns using the Cell method is less efficient than just calling AddRow once with all the cell data:

```
// Inefficient
MyListBox.AddRow("")
MyListBox.Cell(MyListBox.LastIndex, 0) = "Col0"
MyListBox.Cell(MyListBox.LastIndex, 1) = "Col1"
MyListBox.Cell(MyListBox.LastIndex, 2) = "Col2"
MyListBox.Cell(MyListBox.LastIndex, 3) = "Col3"

// More efficient
MyListBox.AddRow("Col0", "Col1", "Col2", "Col3")
```

## Dim Outside Loops

Xojo allows you to Dim variables within code blocks, including loops. This can be handy for code organization and it is usually recommended. But you can get a speed improvement by declaring a variable outside the loop and re-using it within the loop instead.

```
While someConditionIsTrue
  Dim t As Text = CallMethod
Wend
```

By writing it this way instead, your loop can save some time:

```
Dim t As Text
While someConditionIsTrue
  t = CallMethod
Wend</code>
```

## Loop Analysis

As several of these tips have noted, improving loops is often the path to code optimization. Because loops repeat many times, something slow within a loop is magnified many times. So improving something within a loop can sometimes result in a significant speed improvement.

### **Use a Different Algorithm or Technique**

Sometimes your algorithm is just slow and performance tuning will not help enough to matter. For example, if you are searching data sequentially no amount of performance tuning is likely to help significantly. It will likely be faster to first sort the data and then do a binary search on the results. Or it may make sense to put data into an in-memory database and use its capabilities for fast searching.

# Build Automation

The Build Automation feature is used to automatically run specific tasks when you run or build your project. These tasks are called Build Steps. There are three build steps available: Copy Files, Script and External Script.

## Table of Contents

- [Overview](#)
- [Copy Files](#)
- [Script and External Script](#)

## Overview

To add a Build Step to your project, select the Insert menu using the toolbar of main menu and then select the Build Step submenu.

Build Steps added to the CONTENTS area of the Navigator are inactive by default.

To activate a Build step, you need to move it to a build target. You do this by dragging the Build Step from the CONTENTS area to the BUILD area of the Navigator. Drop the Build Step onto one of the targets (OS X, Windows, Linux or iOS) and the Build Step becomes active when running or building for that target.

Once you add a Build Step to the target, the target can be expanded to see the build step. When you expand it you will see the build step and an item simply called Build. The Build item allows you to specify whether the Build Step occurs before the project is built or after. Drag the Build Step before the Build item to have it be processed before the Build is done and drag it after the Build item to have it be built after the Build is done.

Each Build Step has an “Applies to” setting that allows you to specify: Both, Debug or Release.

Debug means the Build Step is processed only when doing a Debug Run. Release means the Build Step is processed only when doing a standalone release Build. And Both means the Build Step is always processed.

## Copy Files

The Copy Files step allows you to copy a file, files or a folder to a specified location during the build process. Simply drag the files you want copied to the empty area. For each of these locations you can also specify a subfolder for the files. The locations are: App Parent Folder, Resources Folder, Frameworks Folder, Bundle Folder Parent and Contents Folder.

- App Parent Folder: On Windows and Linux, the specified files are copied alongside the application executable. On OS X, the files are copied next to the executable in the Application Bundle (Contents->Mac OS).
- Resources Folder: The specified files are copied to a Resources folder. On Windows and Linux the Resources folder is created alongside the application itself. On OS X, the Resources folder is contained within the Application Bundle (Contents->Resources).
- Frameworks Folder: On Windows and Linux, the files are copied to the Libs folder for the application. On OS X, the files are copied to the Frameworks folder in the Application Bundle (Contents->Frameworks).
- Bundle Parent Folder: On Windows and Linux, the files are copied to the same folder containing the executable (same as App Parent Folder). On OS X, the files are copied to the folder containing the Application Bundle itself.
- Contents Folder: On Windows and Linux, the file are copied to the same folder containing the executable (same

as App Parent Folder). On OS X, the files are copied to the Application Bundle (Contents).

On iOS, there is an additional step added by default that is called "Sign". Any Copy File Build Steps that you add should be **before** the "Sign" step so that the files you have copied into the app bundle are properly signed for submission to the App Store.

## Script and External Script

Scripting is a powerful way to automated your application builds. Scripts can contain a wide variety of commands to control the build process. All the commands are listed in the [IDE Scripting](#) section, but some useful ones include: DoCommand, DoShellCommand and Speak.

DoCommand is used to run built-in IDE commands, perhaps the most useful is to save your project before it is run:

```
DoCommand("SaveFile")
```

To add such a script to your build process, add a Script Build Step using the Insert menu or Insert button. The script is added to the Navigator and the Script Editor appears. In the Script Editor, enter the code:

```
DoCommand("SaveFile")
```

Now you can drag the script to a target in the BUILD section of the Navigator so that it runs. If you want the script to run before the build is started, then drag it before the "Build" item. If you want it to run after, then drag it after the "Build" item.

The Speak command uses the built-in voice to speak the supplied text. You can use this to let you know when a task has finished.

```
Speak("Build Complete.")
```

DoShellCommand allows you to run a shell command. You can run any shell command that is available in the Terminal or Command Shell of the operating system. A shell command might be used for more robust file management, to manipulate permissions, run commands to digitally code sign your application or anything else that you want.

This example runs the codesign command on OS X:

```
DoShellCommand("codesign MyApplication.app")
```

An external script works the same except the script is stored in an external text file that you select. You would use an External Script if you have a script that you share across multiple projects.

# IDE Scripting

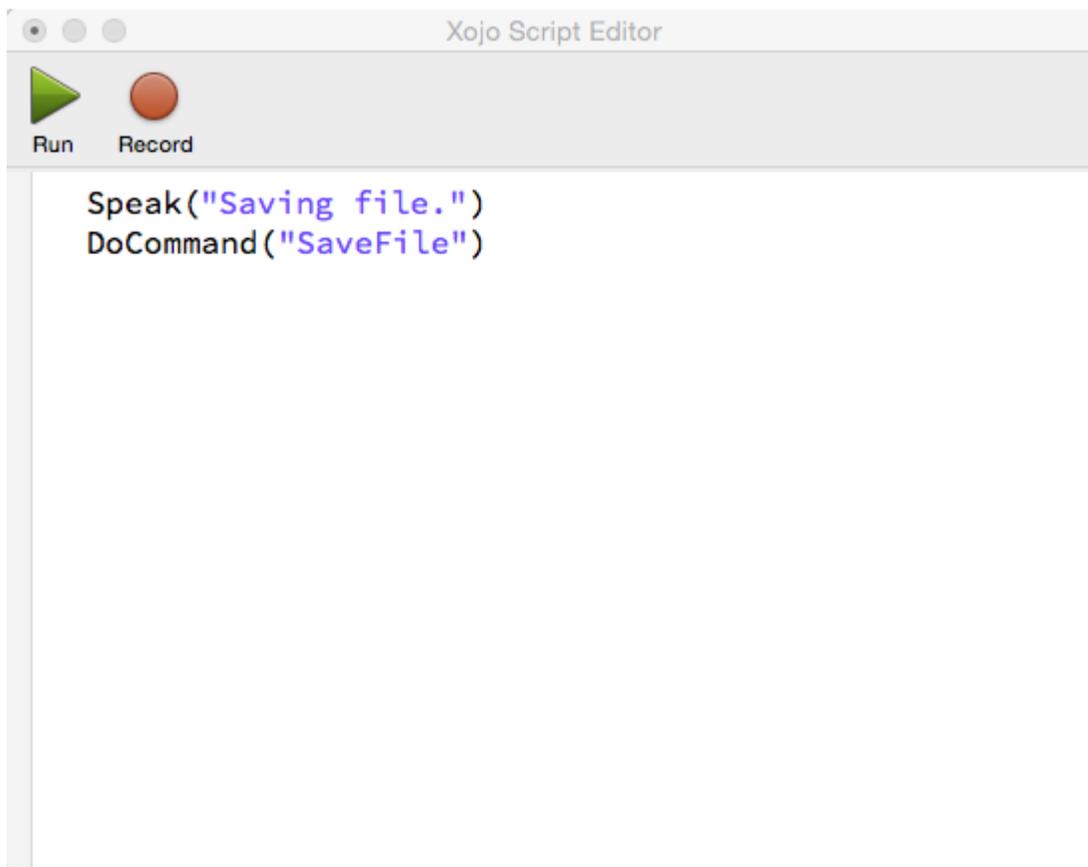
IDE Scripting is the ability to create scripts that allow you to manipulate the Xojo user interface (IDE).

## Table of Contents

- [Creating an IDE Script](#)
- [Saving and Loading IDE Scripts](#)
- [Recording a Script](#)
- [Controlling Xojo from the Command Line](#)

## Creating an IDE Script

To create an IDE script, select IDE Scripts → New IDE Script from the File menu. This opens the XojoScript Editor. You can open multiple IDE Script editors, each with their own scripts.



In this editor you write code using the XojoScript scripting language, which consists of Xojo programming language commands (such as If..Then or For..Next). You can also use special IDE Scripting commands such as DoCommand, DoShellCommand and more.

All the IDE Scripting commands are listed in the next section.

This simple script saves the current project:

```
Speak("Saving file.")
```

```
DoCommand("SaveFile")
```

To run the script, click the Run button on the Xojo Script Editor toolbar.

## Saving and Loading IDE Scripts

IDE Scripts are saved as text files and are not included within the project. To save a script, use File → Save or Save As. To load an existing script into the Script Editor, use File → Open.

You can also add scripts to the IDE Scripts submenu. Clicking scripts in this menu immediately runs them; they do not open in the Script Editor. To have a script appear in this submenu, add the saved script to the Scripts folder alongside the Xojo app or add them to a Scripts folder you create alongside your project file.

IDE Scripts can use Xojo language commands in addition to the commands described in the [IDE Scripting Commands](#) sections.

## Recording a Script

On the XojoScript Editor toolbar is a Record button. When pressed, any commands in the IDE that have a corresponding script command are recorded when they are used. To try this, click the Record button (the text changes to Stop Recording) and then select File → Open to open a project.

You'll see the this command appear in the XojoScript Editor:

```
DoCommand("OpenFile")
```

Now click Run to run your project (not the script) and you'll see command appear in the editor:

```
DoCommand("RunApp")
```

As you use the IDE, any actions that have corresponding script commands will appear in the XojoScript Editor as they are recorded. All these recordable commands are listed on the [DoCommand](#) page.

## Controlling Xojo from the Command Line

Although Xojo cannot be run from the command line, it can be controlled using a separate command-line app that communicates with Xojo using IPC sockets and the "XojoIDE" path.

The IDECommunicator example project (Examples/Advanced/IDE Scripting/IDE Communication) contains the project for an app that can be used control Xojo from the command line by sending IDE Scripts to Xojo for it to run.

Xojo must already be running in order to the communication to work. To use it, build the app for you platform and then run it from the command line with the "-h" option to see how it works. This command runs it on OS X and Linux (# indicates your shell/terminal prompt):

```
# ./IDECommunicator -h
```

# IDE Scripting Commands

IDE Scripts can use the Xojo language commands and data types in addition to the commands described in this section. IDE Scripting cannot use any framework classes or methods.

## Summary

Type	IDE Scripting
Groups	<a href="#">Building</a> <a href="#">DoCommand</a> <a href="#">Input/Output</a> <a href="#">Notification</a> <a href="#">Project</a> <a href="#">String</a>
Related	

# Building Commands

## Summary

Type	IDE Scripting
Commands	<a href="#">BuildApp</a> <a href="#">BuildLanguage</a> <a href="#">BuildLinux</a> <a href="#">BuildMacCocoa</a> <a href="#">BuildCurrentPlatform</a> <a href="#">BuildRegion</a> <a href="#">BuildWin32</a> <a href="#">BuildWebProtocol</a> <a href="#">BuildWebDebugPort</a> <a href="#">BuildWebPort</a> <a href="#">CurrentBuildAppName</a> <a href="#">CurrentBuildLocation</a> <a href="#">CurrentBuildTarget</a>
Related	

## Commands

### BuildApp(buildType As Integer[, reveal As Boolean]) As String

Starts building the current project.

#### Notes

The *buildType* can be one of these values:

Value	Build Target	32/64 Bit	Architecture
3	Windows	32-bit	Intel
4	Linux	32-bit	Intel
6	OS X console	32-bit	Intel
7	OS X GUI	32-bit	Intel
11	iOS device	32-bit	Intel
12	Xojo Cloud	32-bit	Intel
14	iOS device	64-bit	Intel
15	iOS device (Universal)	32 and 64-bit	ARM
16	OS X (all)	64-bit	Intel
17	Linux	64-bit	Intel

Value	Build Target	32/64 Bit	Architecture
18	Linux	32-bit	ARM
19	Windows	64-bit	Intel

If *reveal* is True, the built app is displayed using the OS file manager.

Returns a String containing the Shell path of the built app.

 BuildApp cannot be used in an IDE Script that is called by Build Automation.

### Sample Code

Display the location of an OS X build:

```
Dim appPath As String
appPath = BuildApp(7) ' OS X build
Print("Built: " + appPath)
```

## BuildLanguage As String

Used to get or set the build language. This is the “Language” property on the Shared Build Settings.

### Sample Code

Change the language to English:

```
If BuildLanguage = "Default" Then
    BuildLanguage = "English"
End If
```

## BuildLinux As Boolean

Gets or sets whether to build the project for Linux.

### Sample Code

Create a Linux build:

```
BuildLinux = True
DoCommand("BuildApp")
```

## BuildMacCocoa, BuildMacMachOx86

Gets or sets whether to build the project for OS X.

## BuildCurrentPlatform As Boolean

Gets or sets whether to build the project for the current platform.

---

## BuildRegion

???

---

## BuildWin32 As Boolean

Gets or sets whether to build the project for Windows.

---

## BuildWebProtocol As Integer

Gets or sets the type of web app to build.

### Notes

The values can be:

0 = CGI

1 = Standalone

### Sample Code

Set to build a web app as CGI:

```
BuildWebProtocol = 0 // CGI
```

---

## BuildWebDebugPort As Integer

Gets or sets the web port for running debug apps.

### Sample Code

Set the debug port:

```
BuildWebDebugPort = 8100
```

---

## BuildWebPort As Integer

Gets or sets the web port for built apps.

### Notes

Use a value of -1 to choose a port automatically.

---

## Sample Code

Set the port:

```
BuildWebPort = 8080
```

---

## CurrentBuildAppName As String (read-only)

Returns the name of the app that was built. This can only be used in a Build Automation IDE Script that runs after the Build Step.

## Sample Code

Display the built app's name:

```
Print(CurrentBuildAppName)
```

---

## CurrentBuildLocation As String (read-only)

Returns the shell path to the app that was built. This can only be used in a Build Automation IDE Script that runs after the Build Step.

## Sample Code

Display the built app's path:

```
Print(CurrentBuildLocation)
```

---

## CurrentBuildTarget As Integer (read-only)

Returns an integer that specifies the type of app that was built. This can only be used in a Build Automation IDE Script that runs after the Build Step.

## Notes

The return value is one of these values:

Value	Build Target	32/64 Bit	Architecture
3	Windows	32-bit	Intel
4	Linux	32-bit	Intel
6	OS X console	32-bit	Intel
7	OS X GUI	32-bit	Intel
11	iOS device	32-bit	Intel

---

<b>Value</b>	<b>Build Target</b>	<b>32/64 Bit</b>	<b>Architecture</b>
12	Xojo Cloud	32-bit	Intel
14	iOS device	64-bit	Intel
15	iOS device (Universal)	32 and 64-bit	ARM
16	OS X (all)	64-bit	Intel
17	Linux	64-bit	Intel
18	Linux	32-bit	ARM
19	Windows	64-bit	Intel

### Sample Code

Display the value for the target that was built:

```
Dim result As Integer = CurrentBuildTarget
```

# DoCommand

These commands are used to run other scripts or to execute commands that perform an IDE action.

## Summary

Type	IDE Scripting
Commands	<a href="#">DoCommand</a>
Related	

## Commands

### DoCommand(cmdName As String)

Executes the specified IDE command. Refer Commands for DoCommand for a list of available commands.

#### Sample Code

Save the project:

```
DoCommand("SaveFile")
```

## Commands used by DoCommand

The following commands may be used as parameters to the DoCommand method.

### Project Navigation and Management

- **OpenFile**: Displays the Open Project File dialog.
- **SaveFile**: Saves the current project with no prompt.
- **SaveFileAs**: Displays the File Save As dialog.
- **Import**: Displays the import file dialog.
- **CloseWindow**: Closes the current workspace window.
- **RunApp**: Runs the current project in debug mode.
- **RunPaused**: Runs the current project, but does not launch the debug app.
- **BuildApp**: Builds the current project.
- **Print**: Displays the print dialog.
- **PageSetup**: Displays the page setup dialog.
- **GoBack**: Go backward in tab history.
- **GoForward**: Go forward in tab history.
- **Help**: Shows the Language Reference window.
- **Library**: Toggles the display of the Library.
- **Inspector**: Toggles the display of the Inspector.
- **Find**: Toggles display of Find pane.

- ToggleErrors: Not implemented.
- ToggleMessages: Not implemented.
- NewTab: Adds a new tab to the current workspace window.

## Project Items

- NewClass: Add a new class to the current project.
- CopyFilesStep: Adds a new Copy Files Step to the current project.
- RunIDEScriptStep: Adds a new Script Step to the current project.
- NewRunExternalScriptStep: Displays file selector dialog to add a new External Script Step to the current project.
- NewInterface: Adds a new interface to the current project.
- NewContainerControl: Adds a new container control to the current project.
- NewFileTypes: Adds a new file type set to the current project.
- NewFolder: Adds a new folder to the current project.
- NewMenuBar: Adds a new menu bar to the current project.
- NewModule: Adds a new module to the current project.
- NewReport: Adds a new report to the current project.
- NewToolbar: Adds a new toolbar to the current project.
- NewWindow: Adds a new window to the current project.
- AddWebPage: Adds a new web page to the current project.
- AddWebDialog: Adds a new web dialog to the current project.
- AddWebStyle: Adds a new web style to the current project.

## Project Items Editing

- AddEventImplementation: Displays the Add Event Handler dialog.
- NewMethod: Adds a new method to the selected project item.
- NewProperty: Adds a new property to the selected project item.
- NewNote: Adds a new note to the selected project item.
- NewMenuHandler: Adds a new menu handler to the selected project item.
- NewComputedProperty: Adds a new computed property to the selected project item.
- NewConstant: Adds a new constant to the selected project item.
- NewDelegate: Adds a new delegate to the selected project item.
- NewEnum: Adds a new enumeration to the selected project item.
- NewEvent: Adds a new event definition to the selected project item.
- NewExternalMethod: Adds a new external method to the selected project item.
- NewSharedComputedProperty: Adds a new shared computed property to the selected project item.
- NewSharedMethod: Adds a new shared method to the selected project item.
- NewSharedProperty: Adds a new shared property to the selected project item.
- NewStructure: Adds a new structure to the selected project item.

## Editing

- Comment: Add the comment prefix to the selected text in the code editor (or the current line if no text is selected).
- CheckItemErrors: Equivalent to Project → Analyze Item.
- CheckProjectErrors: Equivalent to Project → Analyze Project.
- SelectAll: Not implemented. Use SelStart and SelLength instead.
- Copy: Copies the selected item in the Navigator to the clipboard.
- Paste: Pastes the text in the clipboard to the active code editor.

- Cut: Not implemented.
- Undo: Not implemented.
- DeleteSelection: Not implemented.

## Layout Editor

- AddFromLibrary: Not implemented.
- EditModeCode: Switch to Code Editor.
- GoToLastEvent: Go to last edited code.
- EditModeView: Switch to Layout Editor.
- StartInlineEditing: Open Default Property popout window.
- LockPositions: Toggle lock for selected control.
- ShowHideTabOrder: Displays Tab Order Editor dialog.
- ToggleMeasurements: Toggle measurements.
- OrderForward: Order selected controls forward.
- OrderToFront: Bring selected controls to front.
- OrderBackward: Order select controls backward.
- OrderToBack: Send selected controls to the back.
- FillWidth: Fill width on the selected control.
- FillHeight: Fill height on the selected control.
- AlignLeft: Aligns the selected controls to the left.
- AlignRight: Aligns the selected controls to the right.
- AlignTop: Aligns the selected controls to the top.
- AlignBottom: Aligns the selected controls to the bottom.
- AlignSpaceHorizontally: Align and space the selected controls horizontally.
- AlignSpaceVertically: Align and space the selected controls vertically.

## Menu Editor

- AddMenu: Adds a top-level menu to the menu bar.
- AddMenuItem: Adds a menu item to the selected menu.
- AddMenuSeparator: Adds a separator to the selected menu.
- AddSubmenu: Adds a submenu to the selected menu.
- ConvertToMenu: Converts a submenu item to a top-level menu.
- ViewAsWin32: Changes to the Windows menu view.
- ViewAsOSX: Changes to the OS X menu view.
- ViewAsLinux: Changes to the Linux menu view.

## File Types Set Editor

- NewFileType: Adds a new file type to the editor.
- Clear: Removes the selected file type from the editor.
- AddCommonFileType\$Pdf: Adds PDF file type.
- AddCommonFileType\$Rtf: Adds RTF file type.
- AddCommonFileType\$Mp3: Adds MP3 file type.
- AddCommonFileType\$Jpeg: Adds Jpeg file type.
- AddCommonFileType\$Png: Adds PNG file type.
- AddCommonFileType\$Any: Adds “any” file type.
- AddCommonFileType\$Text: Adds text file type.
- AddCommonFileType\$Mpeg: Adds Mpeg file type.

- `AddCommonFileType$Quicktime`: Adds QuickTime file type.
- `AddCommonFileType$More`: Adds “more” file type.

### **Report Layout Editor**

- `AddPageSection`: Add page header/footer section to report.
- `AddGroupSection`: Add group header/footer section to report.

### **Copy File Steps Editor**

- `AddFileToCopyFilesStep`: Displays file dialog to select a file to add to the editor.
- `RemoveFileFromCopyStep`: Removes the selected file from the editor.

# Input/Output Commands

These commands interact with the file system or OS.

## Summary

Type	IDE Scripting
Commands	<a href="#">DoShellCommand</a> <a href="#">EnvironmentVariable</a> <a href="#">OpenFile</a> <a href="#">RunScript</a>
Related	

## Commands

**DoShellCommand(command As String, timeOut As Integer = 3000, ByRef resultCode As Integer) As String**

Runs a shell command or shell script.

### Notes

The Shell environment is configured with these variables:

- IDE\_PATH: Path to the folder containing the IDE
- PROJECT\_PATH: Path to the folder containing the current project
- PROJECT\_FILE: Path to the actual project file

The command is run synchronously and returns the output as the result. The timeout is in milliseconds.

### Sample Code

Run the OS X code sign shell command:

```
Dim command As String
command = "codesign -f --deep -s ""Developer ID Application: YourName ""
""YourXojoApp.app "" ""
DoShellCommand(command)</code>
```

---

## EnvironmentVariable As String (read-only)

Gets the value of an environment variable.

### Sample Code

On OS X, displays the HOME environment variable:

```
Print EnvironmentVariable("HOME")
```

---

## OpenFile(filePath As String)

Attempts to open a project file using the specified path, which can be either a native or shell path.

### Sample Code

Open the file at the path:

```
OpenFile("c:\projects\IDEScriptTest.xoyo_binary_project")
```

---

## RunScript(scriptName As String)

Runs the passed script, found in the Scripts folder, either next to the IDE or next to the frontmost project file.

### Sample Code

Run "CommentScript":

```
RunScript("CommentScript")
```

# Notification Commands

The Notification commands are used to get the user's attention by displaying a message or making a sound.

## Summary

Type	IDE Scripting
Commands	<a href="#">Beep</a> <a href="#">Print</a> <a href="#">Speak</a> <a href="#">ShowDialog</a> <a href="#">ShowURL</a>
Related	

## Commands

### Beep

The Beep command plays the system beep sound.

#### Syntax

```
Beep
```

#### Sample Code

```
Beep
```

### Print(text As String)

Displays the specified text in a simple dialog box with an OK button.

#### Sample Code

```
Display "hello":
```

```
Print("Hello")
```

### Speak(text As String)

The Speak command speaks the supplied text.

## Sample Code

Say "hello":

```
Speak("Hello")
```

---

## ShowDialog(Message As String, Explanation As String, DefaultButtonCaption As String, CancelButtonCaption As String, AltButtonCaption As String, Icon As Integer) As String

The ShowDialog command displays a dialog box on the screen. You can specify the message and determine the buttons and their text.

### Notes

If you pass in an empty string to CancelButtonCaption or AltButtonCaption, then the button does not appear.

Returns a String containing the caption of the button that was pressed.

Values for Icon are:

-1: No icon

0: Note icon (App Icon on OS X)

1: Caution icon (On OS X, App Icon is superimposed)

2: Stop Icon (App Icon on OS X)

3: Question Icon (App Icon on OS X)

## Sample Code

Display a simple dialog:

```
Dim result As String
result = ShowDialog("Hello!", _
    "Is it me you're looking for?", _
    "Yes", "No", "", 1)
```

---

## ShowURL(url As String)

Displays the URL in the default web browser.

## Sample Code

```
ShowURL("http://www.xojo.com")
```

# Project Commands

These commands are related to working with the IDE workspace windows, creating new projects and to directly modify or get information about project items.

## Summary

Type	IDE Scripting
Commands	<a href="#">ChangeDeclaration</a> <a href="#">ConstantValue</a> <a href="#">DecryptItem</a> <a href="#">EncryptItem</a> <a href="#">Location</a> <a href="#">NewConsoleProject</a> <a href="#">NewGUIProject</a> <a href="#">NewiOSProject</a> <a href="#">NewWebProject</a> <a href="#">ProjectItem</a> <a href="#">ProjectShellPath</a> <a href="#">PropertyValue</a> <a href="#">QuitIDE</a> <a href="#">SelectWindow</a> <a href="#">WindowCount</a> <a href="#">WindowTitle</a> <a href="#">SelectProjectItem</a> <a href="#">SubLocations</a> <a href="#">Text</a> <a href="#">TypeOfCurrentLocation</a>
Related	

## Commands

**ChangeDeclaration(name As String, parameters As String, returnType As String, scope As Integer, implements As String)**

Changes the declaration of the current property or method.

### Notes

The value for scope may be one of the following:

0 = Public

1 = Protected

2 = Private

Generally this is used to rename a property or method after it has been added to a project item. In the case of a property, the parameters value is used for the default value of the property.

## Sample Code

Add a new property to Window1 and set its values:

```
If SelectProjectItem("Window1") Then
    DoCommand("NewProperty")
    ChangeDeclaration("UserName", "Bob Roberts", "String", 2, "")
End If
```

---

## ConstantValue(name As String) As String

Gets or sets the value of a project item constant.

### Sample Code

Change the value of an existing constant on App:

```
If SelectProjectItem("App") Then
    ' kBeta must already exist
    ConstantValue("kBeta") = "True"
End If
```

---

## DecryptItem(password As String) As Boolean

Decrypts the selected project item using the specified password. Returns True if it succeeded.

### Sample Code

Decrypt the project items in the array:

```
Dim items() As String
items = Array("Class1", "Class2")

Dim name As String
For Each name In items
    If SelectProjectItem(name) Then
        If Not DecryptItem("pa55w0rd") Then
            Print("Error encrypting " + name)
            Exit For
        End If
    End If
End For
Next
```

---

## EncryptItem(password As String) As Boolean

Encrypts the selected project item using the specified password. Returns True if it succeeded.

### Sample Code

Encrypt the project items in the array:

```
Dim items() As String
items = Array("Class1", "Class2")

Dim name As String
For Each name In items
    If SelectProjectItem(name) Then
        If Not EncryptItem("pa55w0rd") Then
            Print("Error encrypting " + name)
            Exit For
        End If
    End If
Next</code>
```

## Location As String

Used to get or set a location in the project. This can be a project item or a property, method, event (etc.) or a project item. Separate each item with a period. When you set a location, the appropriate project item is selected in the Navigator.

### Sample Code

Get the selected project item:

```
Dim loc As String = Location
```

Select "Window1" in the Navigator:

```
Location = "Window1"
```

To determine if a specific control is selected, add "Controls" to the object and follow it with the control name. For example, this checks if a button is selected on Window1:

```
If Location = "Window1.Controls.PushButton1" Then
    Print("PushButton1 is Selected!")
End If
```

---

## NewConsoleProject

Creates a new console project and opens a new workspace window.

## Sample Code

```
NewConsoleProject</code>
```

---

## NewGUIProject

Creates a new desktop project and opens a new workspace window.

## Sample Code

```
NewGUIProject</code>
```

---

## NewiOSProject

Creates a new iOS project and opens a new workspace window.

## Sample Code

```
NewiOSProject
```

---

## NewWebProject

Creates a new web project and opens a new workspace window.

## Sample Code

```
NewWebProject</code>
```

---

## ProjectItem As String

Returns the name of the project item that is selected in the Navigator (of the current tab, if more than one tab is open).

## Sample Code

Display the name of the selected project item:

```
Print(ProjectItem)
```

---

## ProjectShellPath As String

Returns the shell path for the project currently being edited.

## Notes

If the project has not yet been saved, this returns the empty string.

## Sample Code

```
Print(ProjectShellPath)
```

---

## PropertyValue(propName As String) As String

Gets or sets the value of a project item property. This only works for properties of project items that are part of the Xojo framework (such as Window or App). You can only modify properties that are part of the framework (such as Window1.Title), not properties that you have added.

## Notes

You can specify just the property name which will use the currently selected project item. Or you can specify a project item, followed by the property name using dot notation. When referring to the App object, always use the name "App" even if you have renamed it in your project.

The property value is always returned as a string, even if the property itself is not a string.

## Sample Code

Set the App.ShortVersion property to match the version numbers:

```
Dim version As String
version = PropertyValue("App.MajorVersion") + "." + _
    PropertyValue("App.MinorVersion") + "." + _
    PropertyValue("App.BugVersion") + " (" + _
    PropertyValue("App.NonReleaseVersion") + ")"

PropertyValue("App.ShortVersion") = version
```

---

## QuitIDE(saveChanges As Boolean)

Quits the IDE, either ignoring any unsaved changes or saving changes without prompting.

## Sample Code

Quit and save changes:

```
Quit(True)
```

---

## SelectWindow(windowTitle As String)

## SelectWindow(index As Integer)

Selects the IDE workspace window with the specified title or index (0-based), bringing it to the front.

### Notes

The title is the project filename without the extension.

### Sample Code

Bring the first workspace window to the front:

```
SelectWindow(0)
```

Bring the workspace named "Test" to the front:

```
SelectWindow("Test")
```

---

## WindowCount As Integer

Returns the number of open workspace windows in the IDE.

### Sample Code

```
Dim count As Integer = WindowCount
```

---

## WindowTitle(index As Integer) As String

Returns the name of a workspace window using its index (0-based).

### Sample Code

Loop through all open workspaces and saves their contents:

```
Dim saved As String
Dim comma As String
Dim i As Integer
For i = 0 To WindowCount-1
    SelectWindow(i)
    saved = saved + comma + WindowTitle(i)
    DoCommand("SaveFile")
    comma = ", "
Next
Print("Saved: " + saved)
```

## SelectProjectItem(itemPath As String) As Boolean

Selects the project item specified by the path.

### Notes

Returns True if the item was selected, False if not (usually because it does not exist). Specify the path using dot notation, with each level separated by a period.

To select a method, property or other item within a project item, use the Location method.

### Sample Code

Select Method1 on Window1:

```
Dim selected As Boolean = SelectProjectItem("Window1.Method1")
```

Select Class1 that is within Module1:

```
Dim selected As Boolean = SelectProjectItem("Module1.Class1")</code>
```

---

## SubLocations(baseLocation As String) As String

Returns all the locations within the given base location (a module or folder) as a tab-delimited String (ChrB(9)).

### Sample Code

Display the name of each item in a folder:

```
Dim itemList As String  
itemList = Sublocations("Folder1")
```

```
Dim items() As String  
items = itemList.Split(ChrB(9))
```

```
Dim name As String  
For Each name In items  
    Print("Item: " + name)  
Next</code>
```

---

## Text As String

Gets or sets the entire text of what is currently displayed in the current Code Editor.

### Sample Code

Add a comment header to the top of the currently displayed method:

```
Dim header As String
header = "// Copyright 2013 Acme, Inc."

Text = header + EndOfLine + EndOfLine + Text
```

---

## TypeOfCurrentLocation As String

Returns a string that is the type of the item at the current location.

### Sample Code

Show the type of the item at the current location:

```
If SelectProjectItem("Window1.Method1") Then
    Print(TypeOfCurrentLocation) ' displays Method
End If
```

# String Commands

These are general String-related commands that are useful when used with the other IDE Scripting commands.

## Summary

Type	IDE Scripting
Commands	<a href="#">Clipboard</a> <a href="#">EndOfLine</a> <a href="#">SelLength</a> <a href="#">SelStart</a> <a href="#">SelText</a>
Related	

## Commands

### Clipboard As String

Used to get or set the text contents of the OS clipboard.

#### Sample Code

Create a new Note for the selected project item and pastes in the contents of the Clipboard into the Note:

```
DoCommand("NewNote")
Text = Clipboard
```

### EndOfLine As String (read-only)

Returns the default line ending for the OS running the IDE. This is also the line separator used for code editor text that is returned by Text and SelText.

#### Sample Code

Add text to a new note:

```
DoCommand("NewNote")
Text = "Line1" + EndOfLine + "Line2"
```

### SelLength As Integer

Gets or sets the length of characters in the current selection of the code editor.

## SelStart As Integer

Gets or sets the offset of the selection (or insertion) point in the code editor.

---

## SelText As String

Gets or sets the selected text in the code editor. After assign a string to SelText, the selection will be empty and positioned immediately after the inserted text.

### Sample Code

Take the selected text and add an inline constant containing the text to the top of the method and replaces the selected text with the name of the constant:

```
Dim constantText As String  
constantText = SelText
```

```
Dim newConstant As String  
newConstant = "Const kValue = "" + constantText + """
```

```
SelStart = SelStart-1  
SelLength = SelLength + 2  
SelText = "kValue"
```

```
Text = newConstant + EndOfLine + EndOfLine + Text
```

# iOS User Interface Overview

This section covers the user interface components and controls available for iOS apps. You'll learn about Screens and the type of layouts they can contain. You'll learn about Views and Auto-Layout. Lastly, each of the iOS controls is covered, by related groups.

## Topics

- [Screens](#)
- [Views](#)
- [Auto-Layout](#)
- [Buttons](#)
- [Pickers](#)
- [Inputs](#)
- [Decorations](#)
- [Indicators](#)
- [Viewers](#)

# iOS Screens

A Screen defines how your initial layout appears on the iOS device.

## Table of Contents

- [Screen Layout](#)
- [Supported Orientations](#)
- [Device Rotation](#)

You can change the preview mode for the screen by using the toolbar buttons to change the orientation (portrait or landscape) and the device type. These settings affect the preview only and do not affect how the app runs.

The Inspector for a Screen has two sections of importance: Screen Layout and Supported Orientations.

The **Screen Layout** setting allows you to specify the Content property, which determines how the display looks and works. There are three choices: View, Split and Tabs.

## Screen Layout

### View

By default, Screen Layout simply points to View1. This is the initial view that will be displayed on the screen, filling it entirely. You can change this to point to any View that is in your project. This initial cannot be changed, but new Views can be displayed over it by calling the [PushTo](#) method on View.

### Split

A Split layout (SplitView) allows you to split the screen into two sections: a master on the left and a detail on the right. Each of these sections is its own View. This layout is only available on the iPadScreen used by iPad-sized devices. In landscape orientation, the master and detail appear on the screen at the same time.

 In portrait orientation, only the details view appears. Swipe from the left edge of the screen to the right to show the master view.

The Mail app included with iOS is an example of a SplitView in action (when displayed in landscape orientation). It displays the email messages on the left and the selected message in the detail area on the right.

You can also have each section of a Split be a Tab for further granularity.

**ID**

Name

Super  

Interfaces

---

**Screen Layout**

Content  

Left (Main)  

Right (Detail)  

---

**Supported Orientations**

Portrait (Home on Bottom)  OFF

Landscape (Home on Left)  ON

Landscape (Home on Right)  ON

Portrait (Home on Top)  OFF

## Tabs

Much like with a desktop app, the tabs layout allows you to have multiple views that the user can select by using the tab control. In the case of iOS apps, the tab control is displayed at the bottom of the screen. For an example, the iOS Clock app uses tabs at the bottom to select the type of clock you want to use. Use the Edit button for the Tabs property to add or remove tabs, specifying their name or icon. Use the Tab Content property to specify the initial View to display for each tab.



You need to click on each tab on the Screen to change the selected tab so that you can set the Tab Content property for it.

**ID**

Name

Super  

Interfaces

---

**Screen Layout**

Content  

Tabs:

Tab 1 Content  

---

**Supported Orientations**

Portrait (Home on Bottom)  ON

Landscape (Home on Left)  ON

Landscape (Home on Right)  ON

Portrait (Home on Top)  OFF

## Supported Orientations

You can control the types of orientations that are allowed using the Supported Orientation settings of the Inspector for the iPhone and iPad screens. There are four options:

- Portrait (Home on Bottom)
- Landscape (Home on Left)
- Landscape (Home on Right)
- Portrait (Home on Top)

By default, each of these settings is set to YES. If you want to disable an orientation for your app, set it to NO. For example, it is common for iPhone apps to have Portrait (Home on Top) disabled and many iPhone apps also have Landscape orientations disabled.

## Device Rotation

When the user rotates the device to change its orientation, the [Resized](#) event handler on the currently displayed [View](#) is called. In this event handler, you can check the size of the View (using its [Size](#) method) to see if it is now in portrait or landscape. With this information, you can then decide to reposition controls or to display a completely different view, allowing you to have one view for portrait and one for landscape.

To check if the view is in landscape, check if the height is greater than the width:

```
If Self.Size.Height > Self.Size.Width Then
    ' Landscape
End If
```

To check if the view is portrait, check if the height is less than the width:

```
If Self.Size.Height < Self.Size.Width Then
    ' Portrait
End If
```

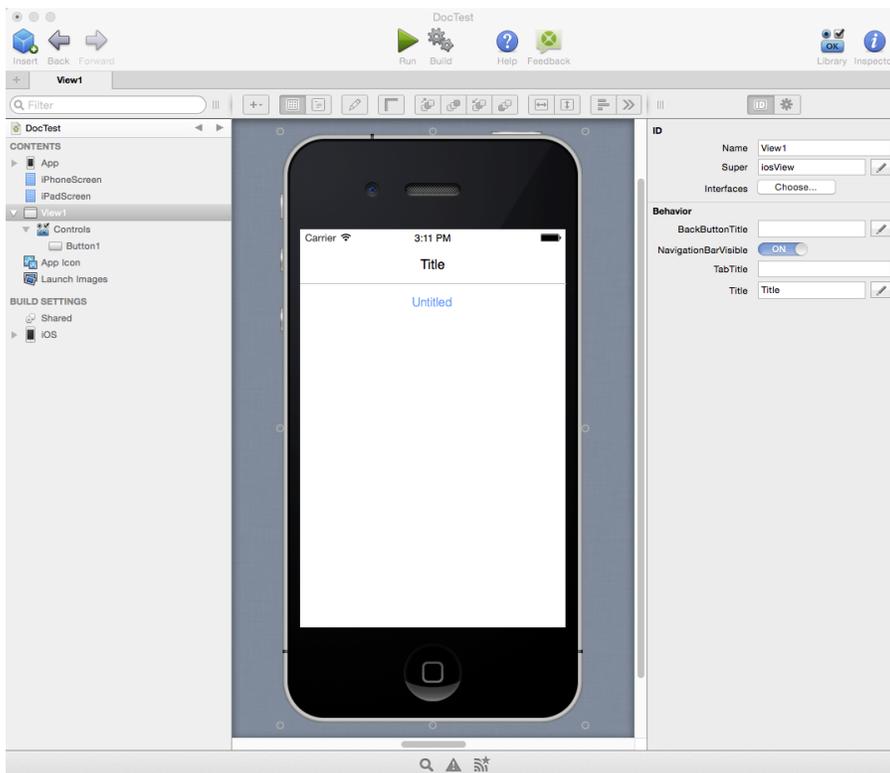
If you not use any special processing, the auto-layout rules you have specified for the controls on the view will adjust your controls. For example, a text field may expand to use more available width when a device is rotated from portrait to landscape.

# iOS Views

## Table of Contents

- [Events](#)
- [Properties](#)
- [Methods](#)
- [Displaying a New View](#)
  - [Displaying a Modal View](#)
- [Adding Toolbar Buttons](#)

Views are where you design the layout of the iOS screen by adding controls. This is similar to a Window in desktop projects or a WebPage in web projects. You simply drag controls from the Library onto the View, positioning them as appropriate. Note that iOS uses a feature called Auto-Layout to determine how the controls move on the screen as the device size or orientation changes.



Add event handlers to the controls the same way you do in desktop and web projects: choose the "+" button in the Layout Editor toolbar and select "Add Event Handler". A view can have several event handlers: Activate, Close, Deactivate, Open, Resized and ToolbarPressed.

## Events

 Refer to [iOSView](#) in the Reference Guide for details on all the events.

There are several event handlers that are used with views:

*Activate*

Called when the view is first opened or redisplayed after another view that was displayed over it is closed.

#### *Close*

Called when the view is closed.

#### *Deactivate*

Called when the view is deactivated by either closing it or displaying another view on top of it.

#### *Open*

Called when the view first opens. This is typically where you will do initialization such as setting up toolbars.

#### *Resized*

Called when the view size changes, which can occur when the device orientation changes. You can use this to reposition controls or perhaps show an entirely different view.

#### *ToolbarPressed*

Called when a button on the Navigation Bar or the Toolbar is pressed.

## Properties

 Refer to [iOSView](#) in the Reference Guide for details on all the properties.

The Behavior section of the View Inspector has these properties you can use: **BackButtonTitle**, **NavigationBarVisible**, **TabTitle** and **Title**.

BackButtonTitle is the title that appears on a subsequent view to get back to this view. For example, if this view contains Notes, then you might set BackButtonTitle to "Notes" so that if a later view displays details for a note (by pushing onto this view), its back button will show "Notes" instead of just "Back". When you then click the "Notes" button, the new view closes and this view reappears.

The Title property is the title of the view.

 The Title and BackButton are only displayed if the Navigation Bar is visible (NavigationBarVisible is ON or True).

## Methods

 Refer to [iOSView](#) in the Reference Guide for details on all the events.

These are the commonly used methods:

#### *Close*

Call this method to close the view. The initially displayed view cannot be closed.

#### *PushTo*

This method is used to display new views as shown below.

## Size

Returns the width and height of the view area as a [Size](#).

## Displaying a New View

You are often going to have more than just a single view for your app. Additional views are displayed by "pushing" them onto the current view. You do this using the `PushTo` method.

For example, say you have a view called `NotesView` and a view called `NoteDetailView` and you want `NoteDetailView` to appear when the user clicks on a note displayed in `NotesView`. This code on `NotesView` displays `NoteDetailView`:

```
Dim details As New NoteDetailView
Self.PushTo(details)
```

## Displaying a Modal View

A modal view is a view that is displayed without being "pushed" onto the current view. A modal view cannot have toolbars. You can show and close a modal view using the `Declare` command. To display a view modally:

```
Dim v As New ModalView

' Display ModalView modally
Declare Sub presentViewController Lib "UIKit" _
    Selector "presentViewController:animated:completion:" _
    (parentView As Ptr, viewControllerToPresent As Ptr, animated As Boolean, completion
As Ptr)

presentViewController(Self.Handle, v.Handle, True, Nil)
```

A button on the modal view is needed to close it:

```
' Close the Modal view to return to the calling view
Declare Sub dismissViewController Lib "UIKit" _
    Selector "dismissViewControllerAnimated:completion:" _
    (parentView As Ptr, animated As Boolean, completion As Ptr)

dismissViewController(Self.Handle, True, Nil)
```

## Adding Toolbar Buttons

A view can have a Navigation Bar, which displays at the top of the view, or a toolbar which displays at the bottom of the view.

Refer to [Toolbars](#) in this section for more information.

---

# Auto-Layout

## Table of Contents

- [Overview](#)
  - [Auto-Layout Rules](#)
  - [Example Projects](#)
- [Technical Information](#)
- [Setting iOSLayoutConstraint in Code](#)
  - [Adding a Constraint to a Control](#)
  - [Setting a Specific Attribute without a Relation](#)
  - [Modifying an Existing Constraint](#)

## Overview

Auto-Layout describes the ability for the controls on the layout to resize and reposition them as the layout size changes. It consists of a collection of rules that allow you to specify controls in relationship to other controls, making it possible to have layouts that work in all these situations:

- Different device screen sizes
- Changing device orientations
- Language translations
- User-specific settings such as font size
- OS changes such as control sizes, fonts and spacing

Without Auto-Layout, you typically would have to create multiple UI screens to account for different possibilities and then also add code to handle other situations. With AutoLayout, you can more easily design a single UI whose controls have constraints that allow them to adjust for the above situations.

For example, the Auto-Layout rules for a button could indicate the following:

- the Left margin of the button should match the Left margin of the titleLabel control (at the same scale with no offset)
- The Top of the button should be 10 points below the bottom of NoteTitleField (at the same scale).
- The Height is fixed at 40 points
- The Width is fixed at 100 points

Should you adjust the layout to change the position of titleLabel or NoteTitleField, the button will be repositioned on the screen as defined by the rules. This is handy when designing you layout, but is even more important when you want your layouts to appropriately use the available screen area of the many different size iOS devices.

Some Auto-Layout rules are set for you automatically as you move and drag the control on the layout. But you can also set any of the rules manually. To edit a rule, click on its name and then click the Edit button. To add a new rule, click the "+" button and choose the rule from the list.

The following Auto-Layout rules can be added, although not all controls have every rule available:

- Left: The left position of the control.
- Right: The right position of the control.
- Top: The top position of the control.
- Bottom: The bottom position of the control.
- Horizontal Center: The centered horizontal position for the control.
- Vertical Center: The centered vertical position for the control.
- Width: The width of the control.
- Height: The height of the control.
- Baseline
- Leading
- Trailing

You do not need to specify every rule for a control. In fact, sometimes that would be confusing. For example, if you specify a Left and Right rule for a control, then also specifying a Width is not necessary.

## Auto-Layout Rules

Each Auto-Layout rule has the following properties that can be set:

- Is (Equal To, Min or Max)
- Relative To (None, Containing View, TopLayoutGuide, BottomLayoutGuide or any other control on the layout)
- Edge (None, Left, Right, Top, Bottom, Horizontal Center, Vertical Center, Width, Height, Leading, Trailing, Baseline, Top Constraint, Bottom Constraint)
- Scale %: The percentage of the Edge property to use.
- Offset (points): The offset to use with the Edge property.
- Priority (Low, Auto, Manual, High, Required): Specifies the importance of the Auto-Layout rule.

This picture gives you an idea of how the various Edge settings are used:

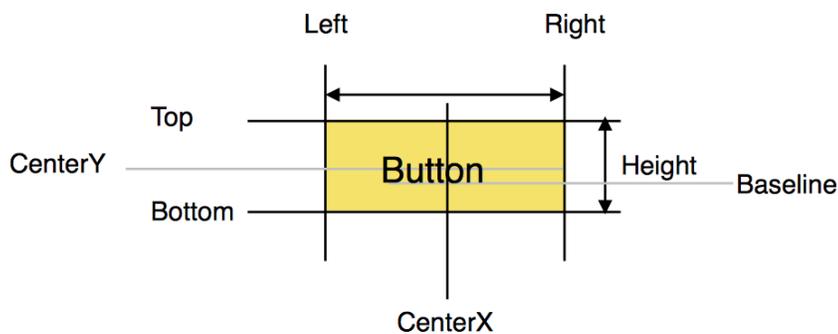
In left to right writing systems Left is the same as leading and Right is the same as trailing. In right to left writing systems, like Arabic, Right is Leading and Left is trailing as the normal flow of layout is from right to left or from the leading edge to the trailing edge.

### Example Projects for Auto-Layout

- [Examples/iOS/Auto-Layout/LayoutConstraintExample](#)
- [Examples/iOS/Auto-Layout/NoCodeProportionalSpaced](#)

## Technical Information

For any single control there are a number of attributes which can be used to refer to its position and size.



In left to right writing systems Left is the same as leading and Right is the same as trailing. In right to left writing systems, like Arabic, right is Leading and left is trailing as the normal flow of layout is from right to left or from the leading edge to the trailing edge.

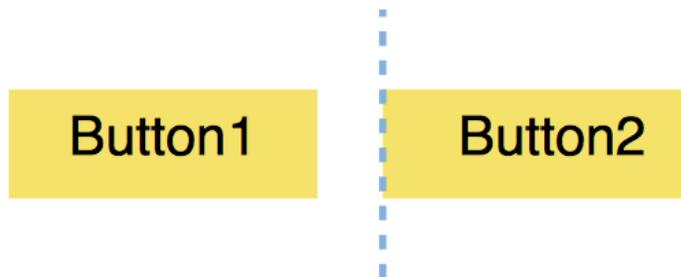
There are other positional and sizing attributes that are very useful as well.

Suppose you have an iOS View (seen below in grey) and a button placed on that view where the left guide line appeared. That gap between the View and the button is the `LeadingMargin`. There is also a trailing margin, top margin and bottom margin.



Suppose you have buttons positioned close to each other and you placed the right most after the left most. When you did this you snapped it to the guideline that appeared offset from the left most button. Something like below.

That gap between the two buttons is a “Standard gap” and is appropriately sized for the version of iOS that you are using.



`Button2.Leading = Button1.Trailing + Std Gap`

## Setting `iosLayoutConstraint` in Code

You can use the `iosLayoutConstraint` class to add new constraints to a control or modify existing constraints.

Constraints are mathematical expressions but are not written in mathematical form.

So say you want to express that ControlA's edge is equal to ControlB's edge. Here is the expression:

```
ControlA.EdgeA = ControlB.EdgeB * factor * offset
```

To configure that using an `iosLayoutConstraint`, these are the values that would be passed in the Constructor:

Parameter	Value
firstItem As Object	ControlA
firstAttribute As AttributeTypes	EdgeA
relation As RelationTypes	<code>iosLayoutConstraint.RelationTypes.Equal</code>
secondItem As Object	ControlB
secondAttribute As AttributeTypes	EdgeB
multiplier As Double	factor
addend	offset

## Adding a Constraint to a Control

This code adds a right constraint to a TextField with an offset of the StandardGap to the containing View (Self):

```
Dim rightConstraint As New iOSLayoutConstraint( _
    MyTextField, _                ' firstItem
    iOSLayoutConstraint.AttributeTypes.Right, _ ' firstAttribute
    iOSLayoutConstraint.RelationTypes.Equal, _ ' relation
    Self, _                       ' secondItem
    iOSLayoutConstraint.AttributeTypes.Right, _ ' secondAttribute
    1.0, _                         ' multiplier
    iOSLayoutConstraint.StandardGap) ' gap
```

## Setting a Specific Attribute without a Relation

Typically you do not specifically set the width or height for a control, you instead set them as relative to some other edge of the containing view or another control.

The reason to set a relation that says "keep this edge this far from some other edge" is so that if you rotate the screen, the control remains in its original relative positioning (say 20 pixels from the edge of the view it's in).

Now if you want to specifically say "keep this to 10 pixels high", remember the original expression:

```
ControlA.EdgeA = ControlB.EdgeB * factor + offset
```

This is how you would configure an iOSLayoutConstraint to set its height to 10:

```
Dim heightConstraint As New iOSLayoutConstraint( _
    MyLabel, _                ' firstItem
    iOSLayoutConstraint.AttributeTypes.Height, _ ' firstAttribute
    iOSLayoutConstraint.RelationTypes.Equal, _   ' relation
    Nil, _                    ' secondItem
    iOSLayoutConstraint.AttributeTypes.None, _  ' secondAttribute
    1.0, _                    ' multiplier
    10)                       ' addend
```

The key here is to specify Nil for the secondItem as you are not actually relating to another control, but instead setting a specific offset (addend) value. This essentially creates this expression:

```
Control.Height = (Nothing.NoEdge) * 1.0 + 10
```

Which evaluates to

```
Control.Height = 10
```

## Modifying an Existing Constraint

You can also modify existing constraints that you have configured using the Auto-Layout properties in the Layout Editor. Before you can do this, you first have to give the constraint you want to modify a name. To do so, select the

---

control, then select the specific constraint in the Auto-Layout section of the Inspector and click Edit. In the topmost field (that has the name of the constraint as its label), give it a name and click Done. This is the name you will use to refer to the constraint in code.

For example, a TextField with a Width constraint named TFWidth can be modified at run-time with this code:

```
Dim c As iOSLayoutConstraint = Self.Constraint("TFWidth")  
c.Offset = 100
```

# iOS Buttons

These are the controls that are part of the Buttons group in the Library. Buttons are tapped to make a selection or perform an action.

## Table of Contents

- [Buttons](#)
- [Segmented Control](#)

## Button

 Refer to [UIButton](#) in the Reference Guide for details on all its events, properties and methods.

Buttons are tapped by the user to evoke an action. A button is a native iOS control and as such, does not have a border unless you have turned on Button Shapes in iOS Accessibility settings.

The Set Default Value feature can be used to quickly change the button Caption.

These are the useful properties for a button:

### *Caption*

The text that displays in the button.

### *Enabled*

A boolean that indicates if the button is enabled and can be clicked or disabled and cannot be clicked.

### *TextColor*

The color of the text that is displayed in the button.

### *Visible*

A boolean that indicates whether the button is shown on the layout when the app is running.

You will primarily use this event handler with a button:

### *Action*

Called when the user taps on the button.

## Usage

Generally you will set the Caption for the button using the Inspector, but you can also set or change the Caption in your code by just assigning it a new value:

```
MyButton.Caption = "Add Task"
```

Enabling and disabling a button can be useful when the button does not because useful unless some data is provided. For example, you may require that a name be entered in a Text Field before a Continue button is enabled. This code in a Text Field TextChange event handler enables or disables a button depending on whether the field has

text:

```
If Me.Text = "" Then
    MyButton.Enabled = False
Else
    MyButton.Enabled = True
End If
```

You can change the color of the text by simply assigning a new color using either the Inspector or from your code:

```
MyButton.TextColor = &cff0000 ' Red
```

You can use the Visible property to show or hide the button. For example, perhaps you only want to show the button if a row in a table is selected. This code in the Action event handler for a table makes a button visible:

```
MyButton.Visible = True
```

The Action event handler is called when the button is clicked. You may have code here to display a new view or perform any type of action. This code in the Action event handler of a button displays a new view:

```
Dim v As New View2
Self.PushTo(v)
```

If your code can be triggered by multiple UI actions, put the code in a method and then call the method from the Action event handlers:

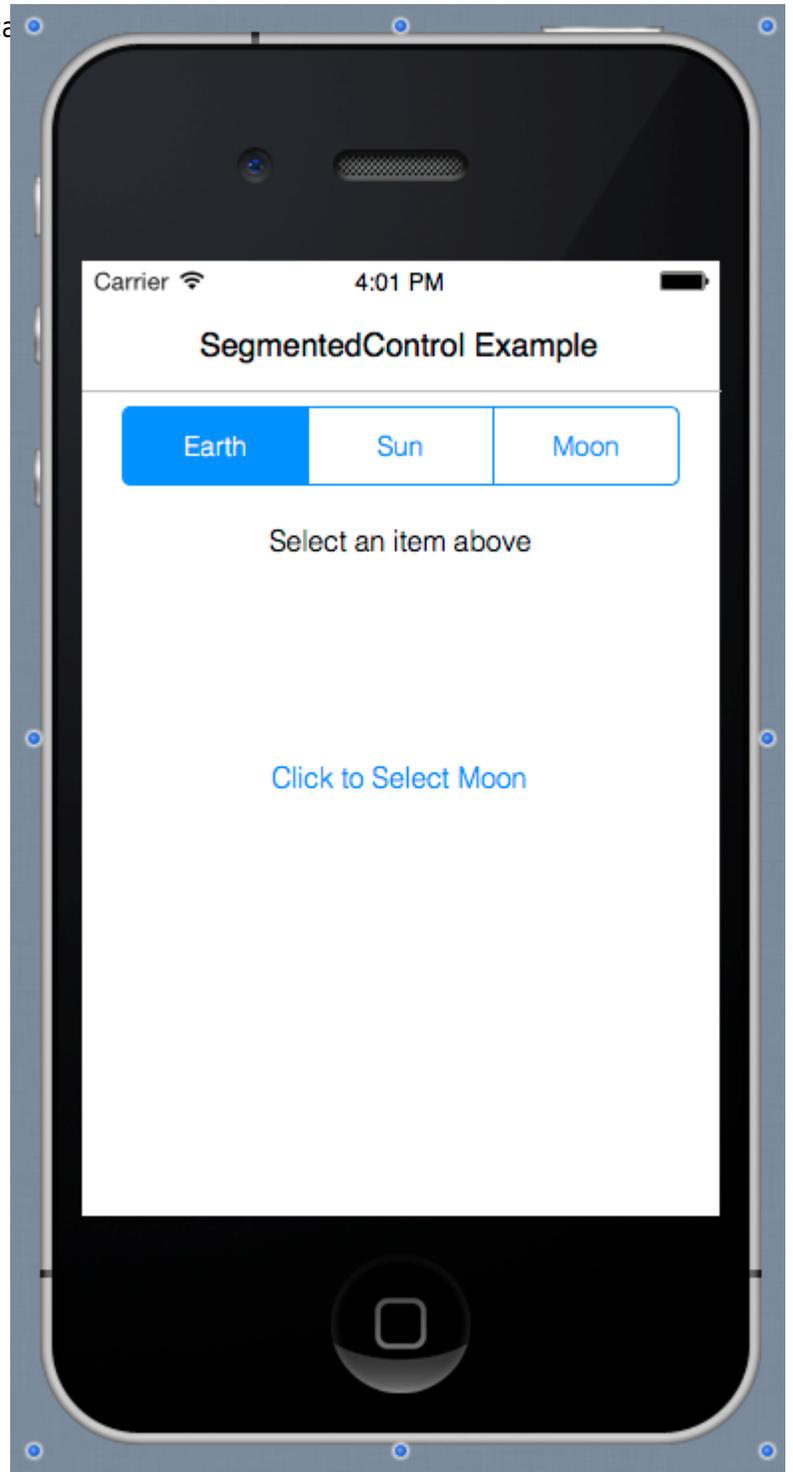
```
MyActionMethod
```

## Segmented Control

The SegmentedControl is a horizontal button made up of multiple segments. Each segment can be clicked independently of the others. Segments can contain text or icons, but do not typically contain both text and icons.

To edit the properties of the segments (Label, Icon and Selected) in the control, use the Segments button on the Inspector properties for the control. This displays the Segment Dialog where you can edit the Label, choose an Icon

and choose a segment to be selected by default. You can



These are the useful event handlers:

#### *Open*

Called when the control first appears on the view and is typically used to do initialization.

#### *ValueChanged*

Called when one of the segments is tapped, passing in the index of the newly selected segment.

These are the useful properties for a Segmented Control:

### *Enabled*

A boolean that indicates if the segmented control can be tapped or if it is disabled and cannot be changed.

### *Value*

Used to get or set the selected segment.

### *Visible*

A boolean that indicates whether the Segmented Control is visible when your app is running.

These are the useful methods:

### *Add*

Add additional segments to the control.

### *Count*

The number of segments in the control.

### *Insert*

Insert a segment before another segment in the control.

### *Item*

Gets the Segmented Control Item for the specified segment index.

### *RemoveAll/RemoveByIndex/RemoveByValue*

Removes all the segments or a specific segment.

## **Segmented Control Item**

Each segment in a Segmented Control is referred to as a Segmented Control Item which has these properties:

### *Icon*

An icon to display for the segment. If you have both an icon and a title, iOS uses only the icon.

### *Selected*

The selected segment. Only one segment should be set as a selected.

### *Title*

The text to display for the segment. If you have both an icon and a title, iOS uses only the icon.

### *Width*

The width of the segment. If you leave these blank, then all the segments will divide the available space equally.

## **Usage**

You can set up the Segmented Control using the Inspector and Segment Dialog, but you can also do everything in

---

code. This code (in the Open event handler) adds segments to a Segmented Control:

```
Me.Add(New iOSSegmentedControlItem("Earth"))  
Me.Add(New iOSSegmentedControlItem("Sun"))  
Me.Add(New iOSSegmentedControlItem("Moon"))
```

You will usually want to perform an action when one of the segments is clicked. This calls the ValueChanged event handler. In the event handler you can check the Value property to see what the index of the newly selected segment is. This code displays the title for the selected segment in a Label:

```
SegmentedControlLabel.Text = "You selected " + Me.Item(Me.Value).Title
```

The Value property (Me.Value) is used to lookup the index of the selected item (Me.Item) and then the Title property for the item is displayed.

# iOS Pickers

These controls are part of the Pickers group in the Library and are used to select a value.

## Table of Contents

- [Switch](#)
- [Slider](#)
- [Date Picker](#)

## Switch

 Refer to [iOSSwitch](#) in the Reference Guide for details on all its events, properties and methods.

A Switch has an on/off position and is used to choose a preference. When the Switch is on, it appears with a green background.



ON



OFF

This event is most used with a Switch:

### *ValueChanged*

Called when the value for the Switch is changed, either by the user clicking on it or by code changing its Value property.

These properties are often used with Switch:

### *Enabled*

A boolean that indicates if the switch is enabled and can be tapped or disabled and cannot be changed.

### *Value*

A boolean that indicates the state of the Switch. When True, the Switch is on and displays with a green background. When False, the Switch is off.

### *Visible*

A boolean that indicates whether the button is shown on the layout when the app is running.

## Usage

To turn a switch on, set its value property:

```
MySwitch.Value = True
```

This code in the ValueChanged event handler enables or disables a Text Field depending on whether a Switch is on or off:

```
If Me.Value Then
    MyTextField.Enabled = True
Else
    MyTextField.Enabled = False
End If
```

## Slider

 Refer to [iOSSlider](#) in the Reference Guide for details on all its events, properties and methods.

The Slider control provides an interface that is used for increasing or decreasing a numeric value. The Slider can only be displayed horizontally.

You can use the Set Default Value feature to set the default position of the slider.



The most commonly used event is:

### *ValueChanged*

Called when the Slider value changes, either by the user adjusting the Slider or by code changing its Value property.

These properties are often used with a Slider:

### *Enabled*

A boolean that indicates if the slider is enabled and can be adjusted or disabled and cannot be adjusted.

### *MaxValue*

The maximum Double value for the slider.

### *MinValue*

The minimum Double value for the slider.

### *Value*

An Double value that indicates the position of the Slider. If Value is assigned a value less than MinValue, then MinValue is used. If Value is assigned a value greater than MaxValue, then MaxValue is used.

### *Visible*

A boolean that indicates whether the button is shown on the layout when the app is running.

## Usage

To have a Slider update a Label with its Value as the Slider is adjusted, you can put this code in the Slider's

ValueChanged event handler:

```
SliderLabel.Text = Me.Value.ToText
```

## Date Picker

**i** Refer to [iOSDatePicker](#) in the Reference Guide for details on all its events, properties and methods.

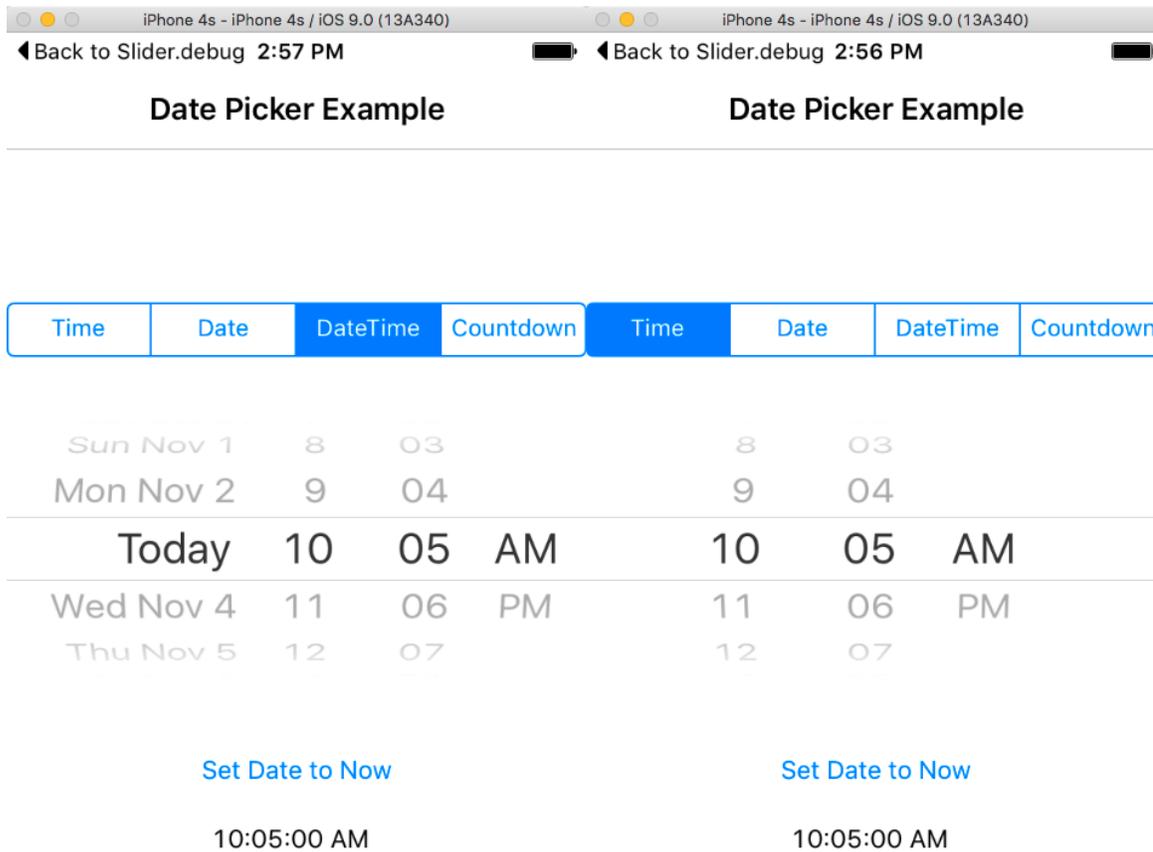
The Date Picker control is used to let the user choose a date or date-related value. The Date Picker can display date values in these formats:

- Time
- Date
- Date and Time
- Countdown Timer

By using a Date Picker, you can ensure that you get a valid date value from the user.



Time	Date	DateTime	Countdown	Time	Date	DateTime	Countdown
		43		September	1	2013	
0		44		October	2	2014	
<b>1 hour</b>		<b>45 min</b>		<b>November</b>	<b>3</b>	<b>2015</b>	
2		46		December	4	2016	
3		47		January	5	2017	
	<a href="#">Set Date to Now</a>			<a href="#">Set Date to Now</a>			
	6300.000000			10:05:00 AM			



The Date Picker has these event handlers:

#### *ValueChanged*

Called when the user selects a date (or time or countdown). Use the properties below to get the value that was selected.

These are the commonly used properties:

#### *CountdownDuration*

Specifies the initial duration (in seconds) to display when Mode is Countdown Timer. It is also the currently displayed countdown duration, so use this to get the value the user has picked.

#### *DefaultDate*

A Date value that specified the initial date to have centered in the Date Picker. It is also the currently displayed date, so use this to get the date value the user has picked. This is applicable when Mode is Date or DateAndTime.

#### *Enabled*

A boolean that indicates if the date picker is enabled and can be used or disabled and cannot be used.

#### *MaximumDate*

The maximum date that can be selected.

#### *MinimumDate*

The minimum date that can be selected.

### *MinuteInterval*

The number of minutes to show in the selector when Mode is Countdown Timer. The default is 1.

### *Mode*

Specifies the type of Date Picker that is displayed (using the [DatePickerMode](#) enumeration): Time, Date, DateAndTime and CountdownTimer.

There is a single method that can also be useful:

### *DefaultDate*

Allows you to set the default date, showing a scrolling animation to the specified date.

## Usage

If you want to get notified every time the Date Picker value is changed, then use the ValueChanged event handler. If you would prefer to get the value at a time or your own choosing, you can refer to the properties.

For example, if you want to have a button that says "Get Selected Date", then its Action event handler could have code like this:

```
Dim selectedDate As Date
selectedDate = MyDatePicker.DefaultDate
```

That same code can also go in the ValueChanged event handler, which means it will be called every time the date changes.

By default the Date Picker shows the current date (when in Date or DateAndTime modes). You can also have a different date as the default by setting the DefaultDate property. This code in the Open event handler for the Date Picker sets the default date:

```
Dim halloween As New Date(2015, 10, 31, TimeZone.Current)
MyDatePicker.DefaultDate = halloween
```

If you are using the Time mode and want to display a specific time, you would also set it using the CurrentDate property. In this case, be sure to set the time component of the date:

```
Dim halloween As New Date(2015, 10, 31, 11, 15, 00, TimeZone.Current) ' 11:15 am
MyDatePicker.DefaultDate = halloween
```

When using a Countdown Timer, you can set the initial countdown time by setting CountdownDuration in seconds. This sets the initial value to 15 minutes:

```
MyCountdownPicker.CountdownDuration = 60 * 15
```

# iOS Inputs

These are the text input controls that are part of the Inputs group in the Library.

## Table of Contents

- [Text Field](#)
- [Text Area](#)

## Text Field

**i** Refer to [UITextField](#) in the Reference Guide for details on all its events, properties and methods.

A Text Field control provides a single-line field for entering text. When the user taps in the field, the on-screen keyboard automatically appears. All standard iOS editing features are available, including copy/paste and auto-correct.



These are the useful events for a Text Field:

### *Open*

The Open event handler can be used for code that initializes the Text Field, perhaps to populate it with text.

### *TextChange*

Called each time the text changes. This means it may be called many times as the user is typing into the field.

These are the commonly-used properties:

### *Enabled*

A boolean that indicates if the text field is enabled and can have text entered or disabled and cannot have text entered or selected.

### *KeyboardType*

Lets you control the type of on-screen keyboard that is displayed when the user taps in the field. By default the full keyboard is displayed, but there are many other keyboards that are optimized for entering numbers, URLs, phone numbers, emails, etc. You change the type by using the `iosKeyboardTypes` enumeration.

### *Password*

A boolean that enables password mode for the field. This means that the text is replaced with dots after each character has been typed.

### *Placeholder*

Used to specify placeholder text that is displayed when the field is empty. This text is used to provide instructions or help and displays in a lighter color to distinguish it from text that is actually entered into the field. The text itself is not in the field and does not have to be cleared before the user begins typing.

## Text

This is the text that is typed in the field. You can use this property to set the default text that appears or check this property to get the text that has been typed.

### *TextAlignment, TextColor, TextFont*

These properties change the alignment, color and size of the text.

## Usage

To populate text in a `TextField`, use the `Text` property, which you can set using the Inspector or with code such as this:

```
NameField.Text = "Bob Roberts"
```

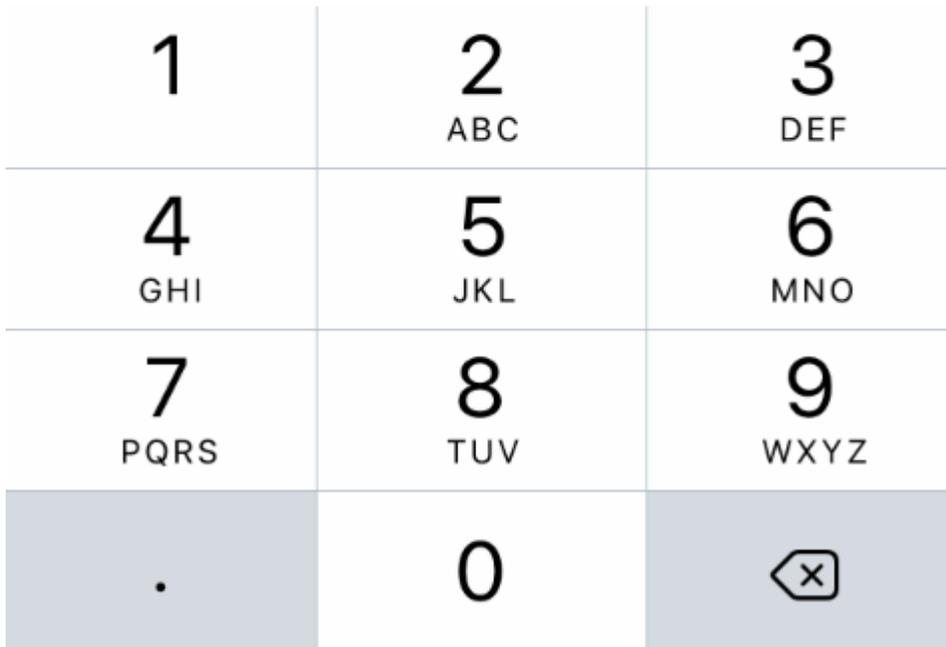
To get the text that was typed, you also use the `Text` property. This gets the text in the `NameField`:

```
Dim userName As Text  
userName = NameField.Text
```

To get the value as it is being changed, use the `TextChange` event handler and the `Text` property. This code updates a label with the text being entered in a text field:

```
NameLabel1.Text = Me.Text
```

Some types of text fields may contain specific information, so it makes sense to change the keyboard type. For example, if you want the user to enter an amount of money, you might want to use the `DecimalPad`, which looks like this:



You can set the `KeyboardType` in the Inspector or with code like this:

```
AmountField.KeyboardType = iOSKeyboardTypes.DecimalPad
```

A text field always has a border around it but there may be times when you do not want a border. This code in the Open event handler for the text field removes the border:

```
Declare Sub setBorderStyle Lib "UIKit" selector "setBorderStyle:" (id As Ptr, style As Integer)
setBorderStyle(Ptr(Me.Handle), 0)
```

## Text Area

**i** Refer to [iOSTextArea](#) in the Reference Guide for details on all its events, properties and methods.

A Text Area control provides a multi-line field for entering text. When the user taps in the field, the on-screen keyboard automatically appears. A text area does not have a border around it.

These are the events commonly used with a Text Area:

### *Open*

The Open event handler can be used for code that initializes the Text Area, perhaps to populate it with text.

### *TextChanged*

Called each time the text changes. This means it may be called many times as the user is typing into the field.

These are the most useful properties:

### *Editable*

A boolean that indicates if the user can edit the text in the text area. When false, the text cannot be modified, but the user can select and copy text it contains.

### *Text*

The text contained in the text area. Set this property to populate the text area, or access it to get the text entered into the text area.

### *TextAlignment, TextColor, TextFont*

These properties change the alignment, color and size of the text.

## **Usage**

To populate the Text Area, set its Text property either using the Inspector or with code:

```
DescArea.Text = "This is some default text."
```

To get the text within the Text Area, you also use the Text property:

```
Dim description As Text  
description = DescArea.Text
```

Like with a Text Field, the TextChange event is called as the text is changed by the user editing in the text area. You might use this event so save the text to a property for use elsewhere in your app:

```
Description = Me.Text ' Description is a property of the view
```

# iOS Decorations

These are the controls in the Decorations group of the Library. These controls display shapes, text or graphics.

## Table of Contents

- [Label](#)
- [Canvas](#)
- [Line](#)
- [Oval](#)
- [Rectangle](#)
- [Separator](#)

## Label

 Refer to [UILabel](#) in the Reference Guide for details on all its events, properties and methods.

The label control displays text on the view. This is often used for displaying information or providing labels to other controls.

You generally only use this event with a Label:

### *Open*

The Open event handler can be used for code that initializes the Label, perhaps to set its text.

These are the events you will use most often:

### *LineBreakMode*

Uses the `UILabelLineBreakMode` enumeration to indicate how text will wrap (or be truncated) when it is too long to fit in the label.

### *Text*

This is the text that appears in the label.

### *TextAlignment, TextColor, TextFont*

These properties change the alignment, color and size of the text.

### *Visible*

A boolean that indicates if the Label is visible when your app runs.

## Usage

Generally you will like set up the text for your labels using the Inspector. But you can also set the text using code:

```
NameLabel1.Text = "Enter your name:"
```

You may have situations where the label should not yet be visible. You can set its Visible property to OFF in the Inspector and then turn it on in your app when appropriate:

```
If showName Then
    nameLabel.Visible = True
End If
```

Labels will automatically show multiple lines for lengthy text if you increase the height. By default the text wraps using word wrapping. But you can change the `LineBreakMode` property to use any of these methods for wrapping and/or truncating:

- `BreakByWordWrapping` (default): Words are not split when text wraps to a new line.
- `BreakByCharWrapping`: Words may be split when text wraps to a new line.
- `BreakByClipping`: Text is clipped at the end, possibly mid-character.
- `BreakByTruncatingHead`: Ellipses appear at the beginning of the text if it cannot all fit.
- `BreakByTruncatingTail`: Ellipses appear at the end of the text if it cannot all fit.
- `BreakByTruncatingMiddle`: Ellipses appear in the middle of the text if it cannot all fit.

Refer to the [iOSLineBreakMode](#) enumeration for more details on these values.

Changing the formatting of a label is done using the `TextAlignment`, `TextColor` and `TextFont` properties. Alignment can be Left, Center, Right, Justified and Natural. To change the alignment you use the [iOSTextAlignment](#) enumeration. This centers the text in the label:

```
nameLabel.TextAlignment = iOSTextAlignment.Center
```

You can set the color to be any `Color` with the `TextColor` property:

To change the font, you use the `TextFont` property in conjunction with the [UIFont](#) class.

## Canvas

 Refer to [iOSCanvas](#) in the Reference Guide for details on all its events, properties and methods.

Canvas is a powerful control that can be used to display graphics, perform animation and even create custom UI controls. All drawing is done through the `Paint` event handler.

These are the useful events for a Canvas:

### *Open*

Called when the Canvas is first displayed and is used for any initialization.

### *Paint*

This method is called when the Canvas contents need to be drawn. This can happen by the OS automatically or by the OS after you call the `Invalidate` method. In this method you are given an `iOSGraphics` object to which you can draw using the methods of the `iOSGraphics` class.

### *PointerDown*

Called when the user taps (and holds) on the Canvas. You are given the coordinates of the tap and the type of tap (single, multi-touch).

### *PointerDrag*

Called when the user drags on the Canvas. You are given the coordinates of the tap and the type of tap (single, multi-touch).

### *PointerUp*

Called when the user stops touching the Canvas. You are given the coordinates of the tap and the type of tap (single, multi-touch).

These properties are used with a Canvas:

### *Height*

The height of the Canvas. This can change if the Canvas size changes due to the orientation changing.

### *Visible*

A boolean that indicates whether the Canvas is shown on the layout when the app is running.

### *Width*

The width of the Canvas. This can change if the Canvas size changes due to the orientation changing.

The methods are typically used with a Canvas:

### *Invalidate*

Call this method to tell the OS that the Canvas should be redrawn the next time the UI is updated. Call this method outside of the Canvas after you have made changes that would cause what is displayed in the Canvas to be updated.

## **iOSGraphics**

Contains all the drawing methods and properties that are used to draw to the graphics object supplied in the Paint event handler.

 Refer to [iOSGraphics](#) in the Reference Guide for details on all its properties and methods.

With `iOSGraphics`, you can draw shapes (outline and filled), text, images, bezier paths, rotate what is drawn and much more.

## **Usage**

With a Canvas you are primarily going to have code in the Paint event that draws something. What you want to draw can be literally anything, from custom UI controls to graphics for games. Starting simple, this code fills the entire Canvas with a blue rectangle:

```
g.FillColor = Color.Blue  
g.FillRect(0, 0, g.Width, g.Height)
```

Similarly, you can draw text to the Canvas like this:

```
g.FillColor = Color.Blue
```

```
Dim t As Text
t = "Hello, World!"
g.DrawTextLine(t, 0, 10, -1, iOSTextAlignment.Left, False) ' Left-aligned
g.DrawTextLine(t, 0, 10, -1, iOSTextAlignment.Center, False) ' Centered
g.DrawTextLine(t, 0, 10, 100, iOSTextAlignment.Right, False) ' Right-aligned
```

If you have an image in your project, you can easily draw it to the Canvas:

```
' MyImage is an image added to the project
g.DrawImage(MyImage, 0, 0)

' Draw image at half its original size
g.DrawImage(MyImage, 0, 0, MyImage.Width/2, MyImage.Height/2)
```

You can use a Timer in conjunction with a Canvas to perform animation. This code draws a square in a Canvas at the specified angle:

```
g.Rotate(RotateAngle, g.Width / 2, g.Height / 2)
g.FillColor = Color.Blue

Const kSize As Integer = 100
g.FillRect((g.Width-kSize)/2, (g.Height-kSize)/2, kSize, kSize)
```

RotateAngle is a property (type of Double) on the View. A Timer that is also on the View is set with a Period of 300ms. It has this code in its Action event handler to rotate the square 45 degrees ( $\text{Pi}/4$ ) each period:

```
Const Pi = 3.14159265358979323846
RotateAngle = RotateAngle + Pi / 4
Canvas1.Invalidate
```

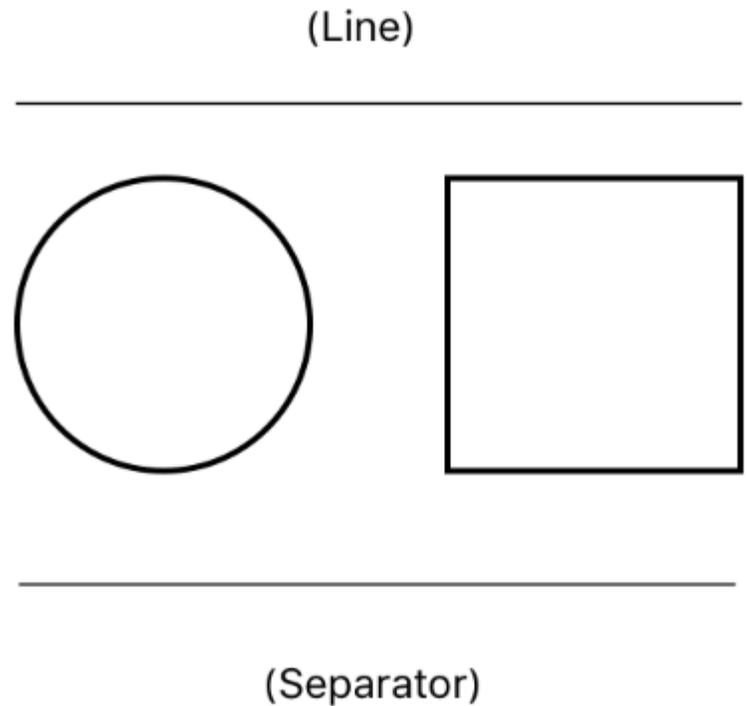
To make it rotate by smaller angles, divide Pi by larger values. The Invalidate method call is needed so that the Canvas gets told to update and redraw the square with the new angle.

## Line

Refer to [iOSLine](#) in the Reference Guide for details on all its events, properties and methods.

Line is a simple control that draws a single line on the layout.

Generally, users do not interact with a Line.



## Oval

Refer to [iOSOval](#) in the Reference Guide for details on all its events, properties and methods.

Oval is used to display ovals and circles on the layout.

Generally, users do not interact with an Oval.

## Rectangle

Refer to [iOSRectangle](#) in the Reference Guide for details on all its events, properties and methods.

Rectangle is used to display rectangles on the layout.

Generally, users do not interact with a Rectangle.

## Separator

Refer to [iOSSeparator](#) in the Reference Guide for details on all its events, properties and methods.

A separator is similar to a line, but displays slightly differently.

Generally, users do not interact with a Separator.

# iOS Indicators

These are the controls that appear in the Indicators group of the Library. They are used to indicate that an operation is in progress.

## Table of Contents

- [Progress Bar](#)
- [Progress Wheel](#)

## Progress Bar

 Refer to [iOSProgressBar](#) in the Reference Guide for details on all its events, properties and methods.

This control displays a horizontal progress bar to indicate how long an operation is taking and to show how close it is to completion.

This event can be used with the Progress Bar:

### *Open*

The Open event handler can be used for code that initializes the Progress Bar.

These are the properties that are commonly used:

### *CurrentValue*

A Double that indicates the current value of the Progress Bar. You use this to indicate how close the Progress Bar is to completion.

### *MaxValue*

A Double that is the maximum value of the Progress Bar. When Current Value reaches MaxValue, the Progress Bar indicates the task has completed.

### *MinValue*

A Double that is the minimum value of the Progress Bar. This is most commonly 0, but it can also be a other values, including negative values.

Loading...



### *Visible*

A boolean that indicates if the Progress Bar is visible when your app runs. You generally do not display the Progress Bar until the actual operation it is tracking is in progress.

This method can also be used:

## SetMinMaxValue

Sets all the value properties at one time.

## Usage

Generally, a Progress Bar is used in conjunction with a long-running task that is in a Thread. You will typically use a Timer to update the Progress Bar periodically based on information from the Thread.

There are two ways that Progress Bars are used. In some cases, you can set the MaxValue to the number of tasks the operation is performing and then increment CurrentValue each time a task is completed. In other cases, you may want to have the Progress Bar go from 0 to 100 and calculate a percentage complete based on the tasks being performed.

As a simple example so you can see how a Progress Bar value moves, you can have a Timer set with a Period of 200ms with this code in its Action event handler to update a Progress Bar:

```
TestProgressBar.Value = TestProgressBar.Value + 1
```

```
If TestProgressBar.Value > TestProgressBar.MaxValue Then
    Me.Mode = Timer.Modes.Off ' stop Timer
End If
```

When you know exactly what you are processing, you can set the Min and Max values. For example you could have code that is looping through an array to process its information. You could set the MaxValue to the UBound of the array and the update the CurrentValue each time through the loop. This code on a Timer could do that:

```
For i As Integer = 0 To MyArray.UBound
    value = MyArray(i)
    Process(value)
    MyProgress.CurrentValue = i
Next
```

If you are processing a large amount of data, it is usually a better idea to use a percentage rather than set MaxValue to some high number. In this case, you have MinValue as 0, MaxValue as 100 and calculate the percentage complete. For example, if you have 736 items and are on item 215, then you would calculate its percentage like this and use the percentage value to update the Progress Bar:

```
Const kMax = 736
For i As Integer = 0 to kMax
    Dim pct As Double
    pct = i / kMax
    MyProgress.CurrentValue = pct
Next</code>
```

## Progress Wheel

Refer to [iOSProgressWheel](#) in the Reference Guide for details on all its events, properties and methods.

A Progress Wheel is used to indicate that a task or operation of unknown length is being performed. When the Progress Wheel is visible it displays on the screen with a spinning animation.

This event is used:

### *Open*

The Open event handler can be used for code that initializes the Progress Wheel.

These properties are used:

### *Shade*

Specifies the shade (light or dark) for the Progress Wheel. Dark is the default.

### *Visible*

A boolean that indicates if the Progress Wheel is visible when your app runs. When it is visible, it displays a spinning animation. Set it to Invisible when it is not needed.

## Usage

Long tasks should be in a Thread so that the UI remains responsive and the spinning animation the Progress Wheel can work. Using a Progress Wheel is as simple as making it visible before you begin the task:

```
MyProgressWheel.Visible = True  
ThreadProcess.Run
```

# iOS Viewers

These are the controls that appear in the Viewers section of the Library. They are used to display information.

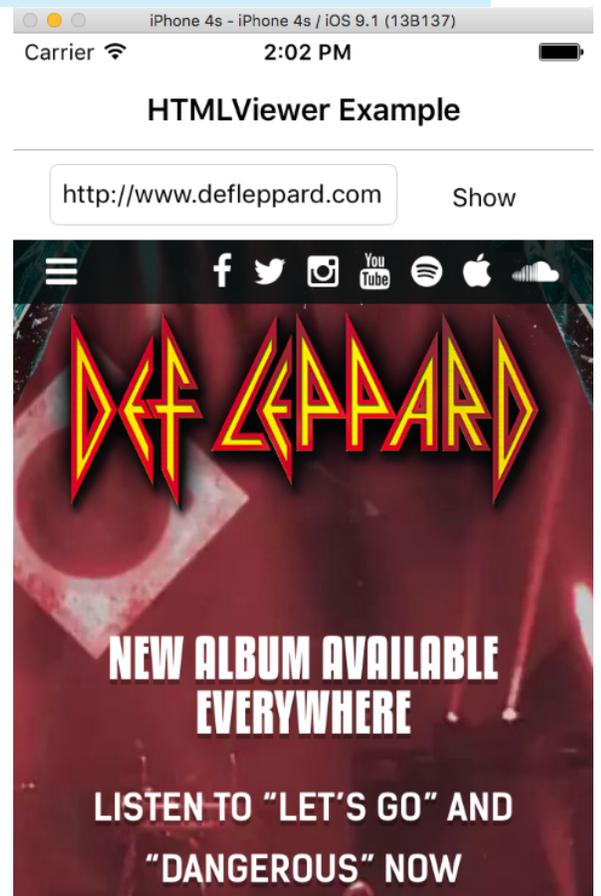
## Table of Contents

- [HTML Viewer](#)
- [Image View](#)
- [Table](#)

## HTML Viewer

 Refer to [iOSHTMLViewer](#) in the Reference Guide for details on all its events, properties and methods.

The HTML Viewer control displays rendered HTML.



This event is useful with HTML Viewer:

### *Visible*

A boolean that indicates if the HTML Viewer is visible when your app runs.

### *LoadURL*

Displays the web page at the specified URL.

## Usage

To display a web page, specify the URL:

```
HTMLViewer1.LoadURL("http://www.xojo.com")
```

If you want to display your own HTML, you first have to save it to a file and then pass the URLPath to the file:

```
Dim html As Text = "<html><body>Hello!</body></html>"
```

```
Dim htmlFile As FolderItem = SpecialFolder.Documents.Child("hello.html")
Dim output As TextOutputStream
output = TextOutputStream.Create(htmlFile, TextEncoding.UTF8)
output.WriteLine(html)
output.Close
```

```
HTMLViewer1.LoadURL(htmlFile.URLPath)
```

## Image View

 Refer to [iOSImageView](#) in the Reference Guide for details on all its events, properties and methods.

Use an Image View to display an Image that is added to the project, or any image that you create or load (from a file

or even image data that was downloaded from



This event can be used with Image View:

### *Open*

The Open event handler can be used for code that initializes the Image View.

These are the properties that are most commonly used:

### *ContentMode*

Indicates how the image is displayed. Uses the ContentModes enumeration to specify how the image should be scaled, stretched or aligned.

### *Image*

The actual image that is displayed.

### *Visible*

A boolean that indicates if the Image View is visible when your app runs.

## **Usage**

An Image View is a fast and easy way to display an image. You have control over how the image is scaled, stretched or aligned. An Image View can only show a single image, so if you require more advanced display of images, use a Canvas.

Generally you'll probably set up the ContentMode and image to display using the Inspector. In the Inspector, the

Image property has a PopupMenu that displays all the images that are in the project. You can select any one to display. You can also choose to browse for an image on a drive and the image will be added to your project and selected here for display.

To add an image in code, just set the Image property yourself. This code in the Open event handler for an Image View displays an image:

```
Me.Image = MyCompanyLogo
```

## Table

**i** Refer to [UITableView](#) in the Reference Guide for details on all its events, properties and methods.

A Table is used to display lists of data. Tables can display information in sections, can have "accessories" to indicate that more information is available. Each row of a Table can display images and up to 2 lines of text.

These events are available:

### *AccessoryAction*

Called when the Detail accessory for a row is tapped.

### *Action*

Called when a row is selected. You are provided the section and row that was tapped.

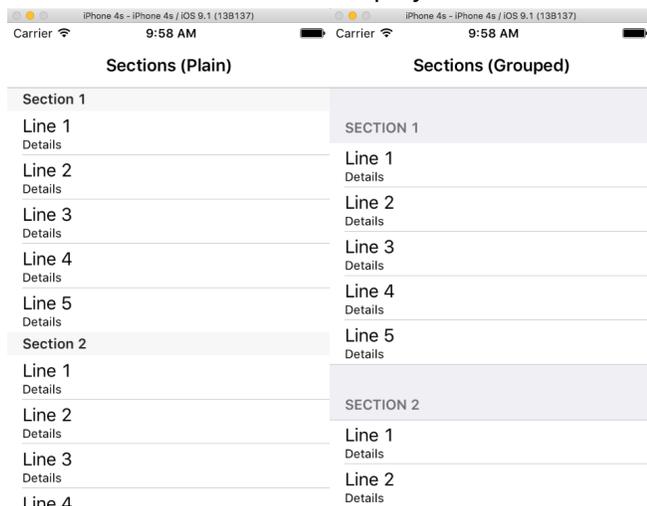
These are the primary properties you might use:

### *DataSource*

Use this to configure a Table to get its data from a data source class rather than populating it using the methods.

### *Format*

Indicates how the table is displayed. There are two choices: Plain and Grouped.



### *SectionCount*

The number of sections in the Table. Use this in conjunction with the various methods to get row counts and other information about the table.

### *Visible*

A boolean that indicates if the Table is visible when your app runs.

These are the methods used with a Table:

### *AddRow, InsertRow*

Adds new rows to the end or inserts rows at a specific position of the table.

### *AddSection, InsertSection*

Add new sections to the end or inserts sections at a specific position of the table.

### *ReloadData, reloadDataInSection, ReloadRow*

Reloads all the data in the table, just the data for a specific section or a single row.

### *RemoveAll, RemoveRow, RemoveSection*

Removes all rows in the table, a single row in a section or an entire section.

### *RowCount*

The count of rows in the specified section.

### *RowData*

Returns an instance of `UITableViewCellData` that contains all the information about a row in a section.

### *SectionTitle*

Gets or sets the title of a section.

When working with a Table, you will also often make use of two related classes: `UITableViewCellData` and `iOSTableDataSource`.

## Cell Data

 Refer to [UITableViewCellData](#) in the Reference Guide for details on all its properties.

`UITableViewCellData` contains information about a row in a table. To populate a table manually, you can create instances of `UITableViewCellData`, set its properties and then add it to the Table. Or you can add rows using just a subset of values manually.

Use the `RowData` method to get the `UITableViewCellData` for a specific row, which you can then use or modify to change what is displayed.

`UITableViewCellData` has these properties:

### *AccessoryType*

The type of (optional) accessory to display for the row. This uses the `AccessoryTypes` enumeration to choose the type of accessory: `None`, `Disclosure`, `Detail`, `Checkmark`.

### *DetailText*

The Detail Text is smaller text that displays below the main text for the row.

### *Image*

When an image is specified, it appears in front of the row Text.

### *Tag*

The Tag is an Auto that can be used to store any useful, related information about the cell for retrieval later (such as a primary key to a database or an instance of a class containing additional information).

### *Text*

The main text that is displayed for the cell row.

## Data Source

 Refer to [UITableViewDataSource](#) in the Reference Guide for details on its methods.

`UITableViewDataSource` is an [Interface](#) that you implement on a class that will provide the data to the Table. The table is then populated by the class so you do not have to use the Add/Insert methods.

These are the methods of the interface that have to be implemented in a class:

### *RowCount*

The number of the rows to the specified section.

### *RowData*

The `UITableViewCellData` for the specified section and row.

### *SectionCount*

The total number of sections.

### *SectionTitle*

The title for the specified section.

## Usage

Starting simply, you can manually add rows to a table. A table always has to have at least one section, even if you don't display it. So at a minimum, you would have code like this to add a row to the first section:

```
Table1.AddSection("")
Table1.AddRow(0, "First Row")
```

Using this version of the `AddRow` method only lets you provide the text for the row. If you need to set other row values, such as the Detail Text or an Image, you will need to first create an `UITableViewCellData`, set its properties and then add it to the Table:

```
Dim cell As New iOSTableCellData
cell.Text = "Product Name"
cell.DetailText = "Product Details"
cell.Image = ProductImage
Table1.AddRow(0, cell)
```

Insert works the same as Add but has an additional paramter to specify the row position within the section for the newly inserted row.

Using these methods, you can populate a table by looping through the data. For example, if you have an array of customer names, you can loop through them and populate the Table like this:

```
Table1.RemoveAll
```

```
Table1.AddSection("") ' add heading for first letter of name
For i As Integer = 0 To CustomerNames.UBound
    Table1.AddRow(0, CustomerNames(i))
Next
```

To add rows with accessories, you have to use iOSTableCellData because that is where the AccessoryType property is. This code adds rows to a table and sets the accessory to a checkmark:

---

Line 0	<input checked="" type="checkbox"/>
Line 1	<input checked="" type="checkbox"/>
Line 2	<input checked="" type="checkbox"/>

---

```
Dim cellData As iOSTableCellData

Table1.AddSection("")
For i As Integer = 0 To 50
    cellData = New iOSTableCellData
    cellData.Text = "Line " + i.ToText
    cellData.AccessoryType = iOSTableCellData.AccessoryTypes.Checkmark

    Table1.AddRow(0, cellData)
Next
```

To toggle the checkmark on and off when the cell is tapped, you can put this code in the Table's Action event handler:

```
Dim cell As iOSTableCellData
cell = Me.RowData(section, row)
```

```

If cell.AccessoryType = iOSTableCellData.AccessoryTypes.Checkmark Then
    cell.AccessoryType = iOSTableCellData.AccessoryTypes.None
    Me.ReloadRow(section, row)
Else
    cell.AccessoryType = iOSTableCellData.AccessoryTypes.Checkmark
    Me.ReloadRow(section, row)
End If

```

This code gets the selected cell and checks if the accessory is a checkmark. If it is, then it turns sets the accessory to None. If there is no accessory, then it sets it to a checkmark. It then reloads the changed row to show the change to the accessory.

The AccessoryTypes enumeration has other accessories you can use instead: Disclosure, Detail and None. The default for cells is None.

If you use an AccessoryType of Detail then tapping the detail indicator calls the Table's AccessoryAction event. In this event you can put the code that performs an accessory-specific action, which is typically to display a new View with additional details. One technique is to have the additional details stored in the Tag of the cell so that you can get it to send to the new View like this:

```

Dim details As New DetailView
details.ApplyData(Me.RowData(section, row).Tag)
PushTo(details)

```

The RowData method is used to get the details of a cell for a specific row. You can then use the data or you can modify it to change what is displayed. After modifying cell data, you'll need to reload the row (or rows) that were modified. This Action event code changes the text of a row that was tapped:

```

Dim cell As iOSTableCellData
cell = Me.RowData(section, row)
cell.DetailText = "This row was tapped."
Me.ReloadRow(section, row)

```

To display an image you also have to use iOSTableCellData. This adds a row using an image from the project:

```

Dim cell As New iOSTableCellData
cell.Image = Product1Image
cell.Text = "Product 1"
Table1.AddRow(0, cell)

```

Sections can be used to provide ways to group the displayed data. One example might be to group a list of names by the first character of their name. So all names that start with "D" would be in their own group. Assuming you have an array of names, you can sort the array and then populate the Table, creating a new group when the first letter of the name changes:

```

Dim names() As Text = Array("Anna", "Andy", "Bob", "Chris", "Chuck", "Dave", "Ed", _
    "Fred", "Greg", "Harry", "Henry")

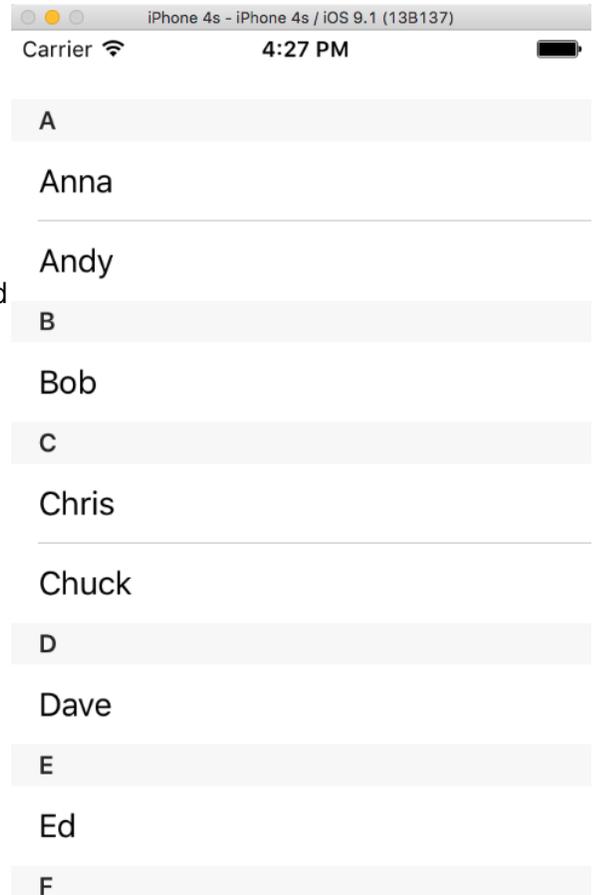
```

```

Dim lastSectionTitle As Text
Dim sectionNum As Integer
For i As Integer = 0 To names.Ubound
    If names(i).Left(1) <> lastSectionTitle Then
        lastSectionTitle = names(i).Left(1)
        sectionNum = Table1.AddSection(lastSectionTitle)
    End If
    Table1.AddRow(sectionNum, names(i))
Next

```

In each of these examples, the Table was populated manually using the various Add methods. Another way you can populate a Table is by using a class as the DataSource. The class is then responsible for getting the data and returning it using the methods provided by the `iOSTableDataSource` interface. This is a bit more work to set up, but does provide a nice separation of the Table UI and its data. The class can get the data from anywhere. It could have its data built-in, it could fetch data from a file, the Internet or a database.



To use a DataSource, you need to create a class that implements the `iOSTableDataSource` interface (refer to the [Interfaces](#) section of you need to refamiliarize yourself with this). In your class, you have to implement the four interface methods: `RowCount`, `RowData`, `SectionCount` and `SectionTitle`.

As an exercise, this is how you would create a Data Source class to populate the names in a section as shown previously.

First, create a new class called "NamesData" and set add the `iOSTableDataSource` interface to it.

In this new class, add two properties:

```
Names() As Text
```

```
SectionNames() As Text
```

Add a Constructor to the class and add the code to initialize these properties:

```
Names = Array("Anna", "Andy", "Bob", "Chris", "Chuck", "Dave", "Ed", _
    "Fred", "Greg", "Harry", "Henry")
```

```
Dim lastSectionTitle As Text
Dim sectionNum As Integer
For i As Integer = 0 To names.Ubound
    If Names(i).Left(1) <> lastSectionTitle Then
        lastSectionTitle = Names(i).Left(1)
        SectionNames.Append(Names(i).Left(1))
    End If
Next
```

Now you can add the code to the four methods to return the correct information.

This is the code for the SectionCount method:

```
Return SectionNames.Ubound + 1
```

This is the code for the SectionTitle method:

```
Return SectionNames(section)
```

This is the code for the RowCount method:

```
' Count the items in the array that start with the character
' of the specified section.
Dim count As Integer
For i As Integer = 0 To Names.Ubound
    If Names(i).Left(1) = SectionNames(section) Then
        count = count + 1
    End If
Next

Return count
```

This is the code for the RowData method:

```
' Find the appropriate item in the array
Dim lookupRow As Integer
For i As Integer = 0 To Names.Ubound
    If Names(i).Left(1) = SectionNames(section) Then
        ' Found the start of this section, which now
        ' serves as an offset into the array.
        lookupRow = i
        Exit For
    End If
Next
```

```
Dim cell As New iOSTableCellData
```

---

```
cell.Text = Names(lookupRow + row)
```

```
Return cell
```

This sets up the Data Source. Yes, it is a bit more complicated, but it eliminates the need for any code on the View to populate the Table. To set up the Table, go to View1 and add a Table (Table1). Also add a property (Names As NamesData).

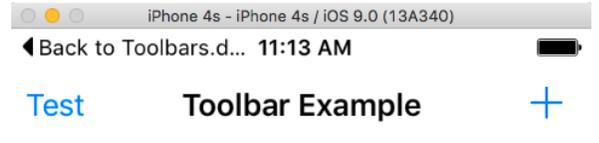
In the Open event handler for the View, create a new instance of NamesData and assign it to the DataSource for the Table:

```
Names = New NamesData  
Table1.DataSource = Names
```

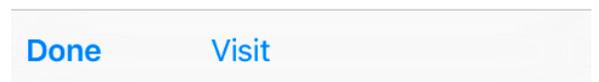
You may be wondering why it is necessary to have the Names property to store the NamesData instance? This is so that the instance does not go out of scope, preventing its data from being loaded into the table.

# iOS Toolbars

A view can have a Navigation Bar, which displays at the top of the view, or a toolbar which displays at the bottom of the view.



Click the buttons in the title bar and toolbar.



## Navigation Bar

To display the Navigation Bar, set the [NavigationBarVisible](#) property to True. You can do this in the Inspector for the view or in code, usually in the Open event handler for the view:

```
NavigationBarVisible = True
```

Buttons can be placed on the left side or right side of the Navigation Bar. To add buttons, use the [LeftNavigationToolbar](#) and [RightNavigationToolbar](#) properties. These properties return a toolbar ([iOSToolbar](#)) which has several methods you can use to add the buttons. Typically you will set up the toolbar in the Open event handler of the view. This code adds a save button to the right navigation toolbar:

```
RightNavigationToolbar.Add(iOSToolButton.NewPlain("Save"))
```

Breaking this command apart, you can see the [Add](#) method of [iOSToolbar](#) is called to add the button. The [Add](#) method can accept a button ([iOSToolButton](#)) to add. The button is created by calling the [NewPlain](#) shared method of [iOSToolButton](#) to create a plain button with the caption of "Save".

The `iOSToolButton` class has many shared methods you can use for creating different types of buttons (and spaces). You can create buttons with a text caption or with an icon, but not both.

If you are using an icon, note that only the mask of the icon is used. Multicolor icons are not used on iOS, so the mask is used with the standard OS tint color applied to it. Toolbar icons should be a maximum of about 66x66, but you can refer to the [Apple Docs on Icon Sizes](#) for more information.

This example adds a button using an image that was added to the project:

```
Dim b As iOSToolButton
b = iOSToolButton.NewPlain(ButtonImage) ' ButtonImage is an image in the project
LeftNavigationBar.Add(b)
```

You can also add buttons that use one of the default system images. Use the `NewSystemItem` shared method of `iOSToolButton` along with the `Types` enumeration to choose the system icon you want to use. This example adds a button using the `SystemRefresh` icon:

```
' Create the button
Dim button As iOSToolButton
button = iOSToolButton.NewSystemItem(iOSToolButton.Types.SystemRefresh)

' Add it to the view
RightNavigationBar.Add(button)</code>
```

## Toolbar

To display a toolbar (which appears at the bottom of the view), simply add buttons to it using the `Toolbar` property of the view the same way you did with the Navigation Bar properties above. This example adds the `SystemAdd` button to the toolbar:

```
Toolbar.Add(iOSToolButton.NewSystemItem(iOSToolButton.Types.SystemAdd))
```

## Handling Toolbar Button Presses

You use the `ToolbarPressed` event handler to check which toolbar button was pressed. This event handler is called for toolbar buttons that are pressed on either the Navigation Bar or the Toolbar. You are supplied with one parameter, which is the button that was pressed. You can then check this button to determine what was pressed so you can call the appropriate code.

This code checks to see if a system button was pressed and if not, then checks the caption of the button to see what was pressed:

```
Select Case button.Type
Case iOSToolButton.Types.SystemEdit
    ' Edit
Case Else
```

```
Select Case button.Caption  
Case "Report"  
    ' Show report  
End Select  
End Select</code>
```

## Toolbar Organization

Regardless of which toolbar you are using (Navigation Bar or Toolbar), you can control the buttons and their positions. For the Navigation Bar, keep in mind that you do not have a lot of room to work with (since the view title appears in the center) so only a few buttons can be displayed on each side. With the Toolbar, you have the entire width of the device, so you have more flexibility.

The examples above all used the `Add` method of `iOSToolbar` to add buttons. This adds buttons from left to right, with the new button appear to the right of the last button added.

You can also use the `Insert` method to add a button at a specific position. You can also remove all the buttons at once or remove individual buttons (`RemoveAll`, `RemoveAllByValue`, `RemoveByIndex`, `RemoveByValue`).

In addition to adding buttons as shown above, the `iOSToolButton` class lets you add two types of special "space" buttons: `FixedSpace` and `FlexibleSpace`. These are not actual buttons since they cannot be clicked but they do take up space on the toolbar. They are used to give your buttons separation or to force some buttons to always appear on the right side of the toolbar.

# iOS Guide

Here are topics about creating iOS apps with Xojo:

- [iOS Overview](#)
- [Auto-Layout](#)
- [Retina Images](#)
- [Using the iOS Simulator](#)
- [Launch Images and App Icons](#)
- [Copying Files to the Device](#)
- [Deploying iOS Apps](#)
- [Ad-Hoc Device Deployment](#)
- [Submitting to the App Store](#)

You may also want to refer to these documents:

- [The New Xojo Framework](#)
- [Migrating to the New Framework](#)

You can also watch these videos about Xojo iOS development

- [iOS QuickStart](#)
- [iOS Overview](#)
- [Xojo Framework](#)
- [Creating an iOS App](#)
- [Deploying iOS Apps](#)
- [Creating Simple iOS Declares](#)
- [Hour of Code with iOS](#)

# iOS Overview

Welcome! I'm sure you're eager to get started creating iOS apps, but before you jump right in, there are a few important topics to understand.

- Screens and Views
- Auto-Layout
- New framework
- Example Projects
- Running in the iOS Simulator
- Building to Deploy on iOS Devices

Additionally, you may want to review the Apple [iOS Human Interface Guidelines](#).

## Screens and Views

When you create your first iOS project, you get the following project items added automatically:

- **App:** The Application object works similarly to how it works for desktop and web projects. In the App Inspector, you can set the Screen to use for iPhone and iPad-sized devices. Leave a device screen blank if you do not want it to have a native UI for the device. For example, to prevent an app from working on iPhone, do not assign an iPhone Screen. If you do not want the app to run natively on an iPad, leave off an iPad Screen. The app will still run on the iPad, but will do so in the OS "scaled" mode.
- **iPhoneScreen:** Specifies the initial screen display when the app is started on iPhone-sized devices.
- **iPadScreen:** Specifies the initial screen display when the app is started on iPad-sized devices.
- **View1:** This is the main layout where you place controls. It is roughly equivalent to a Window or WebPage in desktop and web projects.
- **App Icon:** This item is where you place the various size app icons for the app. Use the ImageMaker utility to create and name the files.
- **Launch Images:** This item contains the various size launch images for the app. Launch images are used kind of like a "splash screen" while the app is actually launching. Typically these are screen shots of your empty View so that it makes it look like your app is starting quickly, but they can be almost anything. There are a variety of sizes for all the devices. In particular, the iPhone 5 size is required in order for an iOS app to fill the screen on iPhone 5 and larger devices.

### Screens

You can change the preview mode for the screen by using the toolbar buttons to change the orientation (portrait or landscape) and the device type. These settings affect the preview only and do not affect how the app runs.

The Inspector for a Screen has two sections of importance: Screen Layout and Supported Orientations.

The **Screen Layout** setting allows you to specify the Content property, which determines how the display looks and works. There are three choices: View, Split and Tabs.



## View

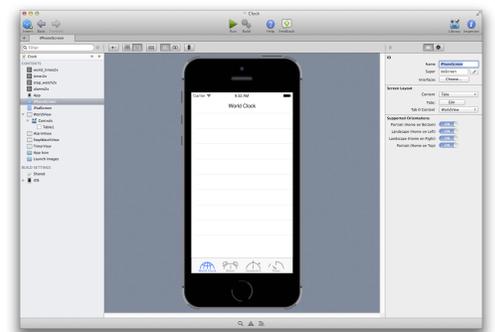
By default, Screen Layout simply points to View1. This is the view that will be displayed on the screen, filling it entirely.

## Split

A Split layout (SplitView) allows you to split the screen into two sections (a master on the left and a detail on the right). Each of these sections is its own View. This layout is only available on iPad-sized devices running in landscape orientation. The Mail app included with iOS is an example of a SplitView in action. It displays the email messages on the left and the selected message in the detail area on the right. You can also have each section of a Split be a Tab.

## Tabs

Much like with a desktop app, the tabs layout allows you to have multiple views that the user can select by using the tab control. In the case of iOS apps, the tab control is displayed at the bottom of the screen. For an example, the iOS Clock app uses tabs at the bottom to select the type of clock you want to use. Use the Edit button for the Tabs property to add or remove tabs. Use the Tab Content property to specify the initial View to display in each tab. You'll need to click on each tab on the Screen to change the tab so that you can set the Tab Content property for it.



## Views

Views are where you design the layout of the iOS screen by adding controls. This is similar to a Window in desktop projects or a WebPage in web projects. You simply drag controls from the Library onto the View, positioning them as appropriate. Note that iOS uses a feature called Auto-Layout to determine how the controls move on the screen as the device size changes. Auto-Layout is discussed in more detail below.

Add event handlers to the controls the same way you do in desktop and web projects: choose the "+" button in the Layout Editor toolbar and select "Add Event Handler".

The Behavior section of the View Inspector has three properties you can use: `BackButtonTitle`, `Title` and `TitleBarVisible`.

The `BackButtonTitle` is the title that appears on a subsequent view to get back to this view. For example, if this view contains `Notes`, then you might set `BackButtonTitle` to "Notes" so that if a later view displays details for a note, its back button will show "Notes" instead of just "Back".

The `Title` property is the title of the view. The `Title` and `BackButton` are only displayed if the `TitleBarVisibleProperty` is ON (True).

## Auto-Layout

Auto-Layout is a new way specify how controls appear on the layout. It consists of a collection of rules that allow you to specify controls in relationship to other controls, making it possible to have layouts that work in all these situations:

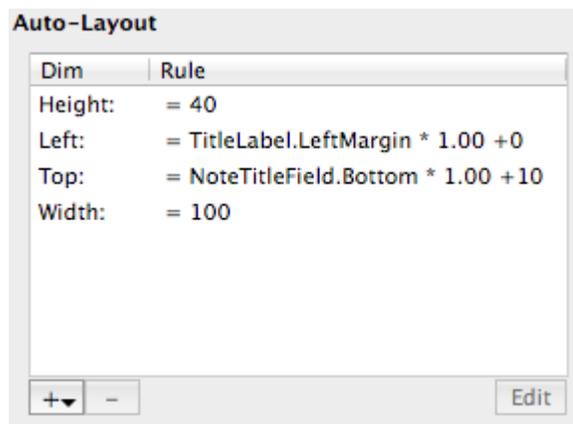
- Different device screen sizes
- Changing device orientations
- Language translations
- User-specific settings such as font size
- OS changes such as control sizes, fonts and spacing

Without Auto-Layout, you typically would have to create multiple UI screens to account for different possibilities and then also add code to handle other situations. With `AutoLayout`, you can more easily design a single UI whose controls have constraints that allow them to adjust for the above situations.

For example, the Auto-Layout rules for a button could indicate the following:

- the Left margin of the button should match the Left margin of the `TitleLabel` control (at the same scale with no offset)
- The Top of the button should be 10 points below the bottom of `NoteTitleField` (at the same scale).
- The Height is fixed at 40 points
- The Width is fixed at 100 points

Should you adjust the layout to change the position of `TitleLabel` or `NoteTitleField`, the button will be repositioned on the screen as defined by the rules. This is handy when designing you layout, but is even more important when you want your layouts to appropriately use the available screen area of the many different size iOS devices.



Some Auto-Layout rules are set for you automatically as you move and drag the control on the layout. But you can

also set any of the rules manually. To edit a rule, click on its name and then click the Edit button. To add a new rule, click the "+" button and choose the rule from the list.

The following Auto-Layout rules can be added, although not all controls have every rule available:

- Left: The left position of the control.
- Right: The right position of the control.
- Top: The top position of the control.
- Bottom: The bottom position of the control.
- Horizontal Center: The centered horizontal position for the control.
- Vertical Center: The centered vertical position for the control.
- Width: The width of the control.
- Height: The height of the control.
- Baseline
- Leading
- Trailing

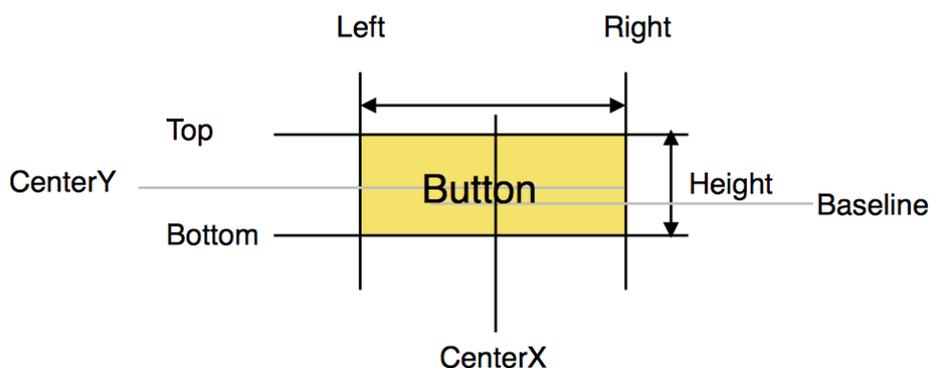
You do not need to specify every rule for a control. In fact, sometimes that would be confusing. For example, if you specify a Left and Right rule for a control, then also specifying a Width is not necessary.

## Auto-Layout Rules

Each Auto-Layout rule has the following properties that can be set:

- Is (Equal To, Min or Max)
- Relative To (None, Containing View, TopLayoutGuide, BottomLayoutGuide or any other control on the layout)
- Edge (None, Left, Right, Top, Bottom, Horizontal Center, Vertical Center, Width, Height, Leading, Trailing, Baseline, Top Constraint, Bottom Constraint)
- Scale %: The percentage of the Edge property to use.
- Offset (points): The offset to use with the Edge property.
- Priority (Low, Auto, Manual, High, Required): Specifies the importance of the Auto-Layout rule.

This picture gives you an idea of how the various Edge settings are used:



In left to right writing systems Left is the same as leading and Right is the same as trailing. In right to left writing systems, like Arabic, Right is Leading and Left is trailing as the normal flow of layout is from right to left or from the leading edge to the trailing edge.

## Example Projects for Auto-Layout

- [Examples/iOS/Auto-Layout/LayoutConstraintExample](#)
- [Examples/iOS/Auto-Layout/NoCodeProportionalSpaced](#)

## New Framework

The iOS framework is all-new, but it should feel familiar if you've created desktop or web apps. There are two main sections for the iOS framework. There is the iOS Framework itself, which contains all the iOS controls and other iOS-specific classes available to you. Some of the items available include: `iOSButton`, `iOSTable`, `iOSCanvas`, etc.

There is also the Xojo framework which contains other classes (which will eventually be available across all Xojo targets). The Xojo framework contains classes such as `Dictionary` and `TCPsocket`. Everything in the Xojo framework is grouped into a collection of namespaces, which helps make it easier to find things based on their functionality. However, you do not really have to worry about the namespaces in iOS projects. By default the Shared Build Setting "Simple Preferences" is set to ON which means you can just refer to things by their class name, such as `Dictionary` instead of `Xojo.Core.Dictionary`.

The Xojo Framework namespaces are: `Core`, `Crypto`, `Data`, `Introspection`, `IO`, `Math` and `Net`. Expect many additions to this framework in upcoming releases.

You can read all about the iOS and Xojo frameworks in the [Reference Guide](#).

In addition, there are a few other key points to mention:

- **Text:** There is a new data type, called `Text`, for dealing with text. It replaces the `String` data type you may have previously used.
- **Exceptions:** The Xojo framework raises exceptions for error situations rather than setting error codes.

## Example Projects

I encourage you to try out each of the example projects included in the iOS folder to get a feel for how all this works. And check the examples with each Beta, as new examples will be added often. There are examples that show how to use Auto-Layout, various iOS controls and other features.

# The New Xojo Framework

## Migrating to the New Framework

Refer to [Migrating to the New Framework](#) to learn how new framework classes and methods correspond with classic framework classes and methods.

The new **Xojo Framework** is an all-new framework designed to eventually replace the existing framework (now referred to as the **Classic Framework**). The Classic Framework has been around for over 15 years and is showing its age in many areas, which limits the ability to provide updates and fix bugs. With the new framework, you will find a more efficient and consistent framework that will allow Xojo to continue to evolve for the future.

The basic rundown is that the Classic Framework that has existed since version one (1998 or so) has accumulated more than a few warts. There are inconsistencies like not all functions being zero-based or the handling of errors. There are also places where the design made sense originally, but operating systems have gone in different directions (random access by index into the contents of a folder is a good example). Worst of all are things where the framework makes it really easy to do the wrong thing.

During planning for iOS support, the team came to the conclusion that it would be an opportunity for a fresh start to address these concerns. Sharing code between the desktop, web, or console frameworks is also very important, so the new framework exists there too. In order to do that without breaking millions of lines of code or forcing a massive rewrite of all existing applications, the new framework was put into the Xojo namespace so that it doesn't conflict with existing framework or user classes.

In most cases the two frameworks are similar, with the biggest differences being the use of the Text and Auto data types. It's also still in its early years, relative to the Classic Framework, so some functionality is missing and will improve over time.

What is the framework? First, you need to understand the main components of Xojo:

- [Xojo Language](#)
- Console Framework
  - [Classic Framework](#): The existing framework
  - [Xojo Framework](#): The new Xojo framework
- UI Frameworks
  - [Desktop](#)
  - [Web](#)
  - [iOS](#)

## Xojo Language

The Xojo Language has not significantly changed. In order to support the new Xojo Framework, a few small changes consisting of a several new keywords have been added:

- [Text](#) data type
- [Auto](#) data type
- [Global](#) command

- [Using](#) command

These language changes are available for all project types, including iOS, desktop, web and console.

The language is otherwise the same with the same object-oriented programming model you have always used.

## Console Framework

The console framework is the collection of classes and methods that supplement the language and that are not related to the user interface. These are things like `Date`, `Dictionary`, `FolderItem`, etc. The Console Framework that has always been a part of Xojo is now called the Classic Framework. It remains available for desktop, web and console projects. However, iOS projects can only use the new **Xojo Framework**.

The [Xojo Framework](#) has a wide variety of classes, such as `Dictionary`, `FolderItem`, `Date`, `TCPsocket`, etc., that are organized into namespaces. Namespaces are simply modules that are used as a way to collect related functionality. All the Xojo Framework features are contained in the Xojo namespace, which itself contains these namespaces:

- [Core](#)
- [Crypto](#)
- [Data](#)
- [Introspection](#)
- [IO](#)
- [Math](#)
- [Net](#)
- [System](#) (iOS-only)
- [Threading](#) (iOS-only)

So to use a class in the Core namespace, such as `Date`, you would typically refer to it as "Xojo.Core.Date". But iOS has a Shared Build Setting (it is ON by default) that enables "Simple References". When this is ON, you can simply refer to the classes by their name, without having to use the namespace prefix.

You can use both the Classic and the Xojo framework (Core namespace only) in your desktop, web and console projects. But because many classes in the Xojo Framework share the same name as classes in the Classic Framework, you will need to be specific about the one you want to use.

For example, in a desktop project if you want to use the new `Date` class in Xojo.Core, you could refer to it like this:

```
Dim d As New Xojo.Core.Date</code>
```

This can be a bit long to write often, so you can use the [Using](#) command to simplify this:

```
Using Xojo.Core  
Dim d As Date // actually uses Xojo.Core.Date</code>
```

You can also set up `Using` for an entire class or module by choosing "Insert->Using Clause" and entering the namespace name in the Inspector.

Keep in mind that you can mix and match Classic Framework classes and Xojo Framework classes in the same method. For cases when you need to do this, you will need to either use the full namespace for the Xojo Framework class or use the `Using` command in a local scope. Here are some examples:

```
Dim d1 As Date // Classic framework Date
Dim d2 As Xojo.Core.Date // Xojo framework Date

If True Then
    Using Xojo.Core
        d = Date // Xojo framework Date
    End If</code>
```

## UI Frameworks

The UI frameworks are the specific UI controls and related classes for creating the UI of your desktop, web and iOS apps. This framework is specific to each target, so the UI Framework for desktop projects is not the same as the UI Framework for web or iOS projects.

You'll notice that the class names differ between targets. For example, a button on the desktop is called **PushButton**, on web is called **WebButton** and on iOS is called **iOSButton**. These controls often operate similarly, but they may have different events, properties and methods.

The UI framework for desktop and web apps has not changed. For iOS apps, you need to use the new classes in the iOS Framework.

# Migrating to the New Xojo Framework

## Table of Contents

- [Namespaces](#)
- [Controls](#)
- [Framework Classes](#)
- [String to Text](#)
- [Converting Values to/from Numbers and Text](#)
- [Databases](#)
- [Global Functions](#)
- [Color](#)
- [Math Functions](#)
- [Date, Time and Intervals](#)
- [JSONItem](#)
- [Graphics](#)
- [MsgBox](#)
- [Variant to Auto](#)

The Xojo programming language (commands and core data types) is largely the same with only a few changes, but the new framework has more significant changes.

First, iOS projects use the New Framework exclusively. This means existing Xojo code you have in desktop or web apps will not be able to work as-is in iOS apps. Desktop, Web and Console projects continue to use the Classic Framework, but can also use a subset of the New Framework.

This section provides information that will help you understand the new framework and how it resembles the older framework.

## Namespaces in the new Framework

The iOS framework is all-new, but it should feel familiar if you've created desktop or web apps. There are two main sections for the iOS framework. There is the iOS Framework itself, which contains all the iOS controls and other iOS-specific classes available to you. Some of the items available include: `iOSButton`, `iOSTable`, `iOSCanvas`, etc.

There is also the Xojo framework which contains other classes (which will eventually be available across all Xojo targets). The Xojo framework contains classes such as `Dictionary` and `TCPsocket`. Everything in the Xojo framework is grouped into a collection of namespaces, which helps make it easier to find things based on their functionality. However, you **do not really have to worry about the namespaces in iOS projects**. By default the Shared Build Setting "Simple References" is set to ON which means you can just refer to things by their class name, such as `Dictionary` instead of `Xojo.Core.Dictionary`.

The Xojo Framework namespaces are: `Core`, `Crypto`, `Data`, `Introspection`, `IO`, `Math`, `Net`, `System` and `Threading`. Expect many additions to this framework in upcoming releases.

## Controls

This is a list of the equivalent Desktop, Web and iOS controls:

<b>Desktop Control</b>	<b>Web Control</b>	<b>iOS Control</b>
Application	WebApplication	<a href="#">iOSApplication</a>
PushButton	WebButton	<a href="#">iOSButton</a>
Picture	WebPicture	<a href="#">iOSBitmap</a> , <a href="#">iOSImage</a>
Canvas	WebCanvas	<a href="#">iOSCCanvas</a>
ContainerControl	WebContainer	<a href="#">iOSContainerControl</a>
n/a	n/a	<a href="#">iOSDatePicker</a>
Graphics	WebGraphics	<a href="#">iOSGraphics</a>
HTMLViewer	WebHTMLViewer	<a href="#">iOSHTMLViewer</a>
ImageWell	WebImageView	<a href="#">iOSImageView</a>
Label	WebLabel	<a href="#">iOSLabel</a>
Line		<a href="#">iOSLine</a>
Oval		<a href="#">iOSSOval</a>
ProgressBar	WebProgressBar	<a href="#">iOSProgressBar</a>
ProgressWheel	WebProgressWheel	<a href="#">iOSProgressWheel</a>
Rectangle	WebRectangle	<a href="#">iOSRectangle</a>
SegmentedControl	WebSegmentedControl	<a href="#">iOSSegmentedControl</a>
Separator	WebSeparator	<a href="#">iOSSeparator</a>
Sound		<a href="#">iOSSound</a>
Slider	WebSlider	<a href="#">iOSSlider</a>
CheckBox	WebCheckBox	<a href="#">iOSSwitch</a>
TabPanel		<a href="#">iOSTabBar</a>
ListBox	WebListBox	<a href="#">iOSTable</a>
TextArea	WebTextArea	<a href="#">iOSTextArea</a>
TextField	WebTextField	<a href="#">iOSTextField</a>
Window	WebPage	<a href="#">iOSView</a> <a href="#">iOSSplitView</a>
Timer	WebTimer	<a href="#">Xojo.Core.Timer</a>
Toolbar	WebToolbar	<a href="#">iOSToolbar</a>

## Framework Classes

Classic Framework Class	Xojo Framework Class
Date	<a href="#">Xojo.Core.Date</a>
Dictionary	<a href="#">Xojo.Core.Dictionary</a> , <a href="#">Xojo.Core.DictionaryEntry</a>
MemoryBlock	<a href="#">Xojo.Core.MemoryBlock</a> <a href="#">Xojo.Core.MutableMemoryBlock</a>
Realbasic.Point	<a href="#">Xojo.Core.Point</a>
Realbasic.Rect	<a href="#">Xojo.Core.Rect</a>
Realbasic.Size	<a href="#">Xojo.Core.Size</a>
Thread	<a href="#">Xojo.Threading.Thread</a>
Timer	<a href="#">Xojo.Core.Timer</a>
WeakRef	<a href="#">Xojo.Core.WeakRef</a>
Crypto	<a href="#">Xojo.Crypto</a>
Introspection	<a href="#">Xojo.Introspection</a>
BinaryStream	<a href="#">Xojo.IO.BinaryStream</a>
FolderItem	<a href="#">Xojo.IO.FolderItem</a>
SpecialFolder	<a href="#">Xojo.IO.SpecialFolder</a>
TextInputStream	<a href="#">Xojo.IO.TextInputStream</a>
TextOutputStream	<a href="#">Xojo.IO.TextOutputStream</a>
Abs, Ceil, Cos, Floor, Sign, etc.	<a href="#">Xojo.Math</a>
HTTPSocket	<a href="#">Xojo.Net.HTTPSocket</a>
TCPsocket	<a href="#">Xojo.Net.TCPsocket</a>

## String to Text

[Text](#) is a modern replacement for String, which combined both textual and binary data. By using Text instead of String, you can be assured that your text data always has a known encoding and is never treated as just a collection of raw data. All classes, methods and properties in the Xojo framework use the Text type.

In non-iOS projects, you can convert Text data to a String, if necessary, simply by assigning the Text value to a String like this:

```
Dim t As Text = "Hello"
Dim s As String = t
```

You can also convert a String to a Text:

```
Dim s As String = "Hello"
Dim t As Text
t = s.ToText</code>
```

String-related methods are now accessible through the [Text](#) type with some methods being renamed.

Asc	<a href="#">Text.Codepoints</a>
Chr	<a href="#">Text.FromUnicodeCodepoint</a>
EndOfLine	&uA <a href="#">Text.FromUnicodeCodepoint(10)</a>
InStr	<a href="#">Text.IndexOf</a>
Join	<a href="#">Text.Join</a>
Left	<a href="#">Text.Left</a>
Len	<a href="#">Text.Length</a>
Mid	<a href="#">Text.Mid</a>
NthField	<a href="#">Text.Split</a>
Replace/ReplaceAll	<a href="#">Text.Replace</a> , <a href="#">Text.ReplaceAll</a>
StrComp	<a href="#">Text.Compare</a>

## Converting Values to/from Numbers and Text

Methods for converting values from number to Text are on the number types.

Val	<a href="#">Integer.Parse</a> , <a href="#">Integer.FromText</a> , <a href="#">Double.Parse</a> , <a href="#">Double.FromText</a> , <a href="#">Currency.FromText</a>
Str	<a href="#">Integer.ToText</a> , <a href="#">Double.ToText</a> , <a href="#">Currency.ToText</a>
CStr	<a href="#">Integer.Parse</a>

You can also use these methods as a replacement for IsNumeric by testing for an exception when they convert:

```
Dim num As Integer
Try
    num = Integer.FromText(NumberField.Text)
Catch e As BadDataException
    num = 0 ' or some other preferred value
End Try
```

## Databases

Only SQLiteDatabase is available for iOS projects. Here are some of the differences.

Old SQLiteDatabase	New SQLiteDatabase
SQLiteDatabase	<a href="#">SQLiteDatabase</a>
SQLiteDatabase.Commit	<a href="#">SQLiteDatabase.SQLExecute("COMMIT")</a>
SQLiteDatabase.DBError	<a href="#">SQLiteExceptions</a> are now raised for errors so use Try/Catch.
RecordSet	<a href="#">SQLiteRecordSet</a>
DatabaseField	<a href="#">SQLiteDatabaseField</a>

## Global Functions

This table describes some of the methods and their equivalents in the Xojo Framework:

Old Global Function	Equivalent iOS Framework Method
MD5	<a href="#">Crypto.MD5</a>

## Color

Color methods are now on the type.

RGB	<a href="#">Color.RGB</a>
HSV	<a href="#">Color.HSV</a>

## Math Functions

All math functions are now in the [Math](#) namespace.

For random number generation, you can use [Xojo.Math.RandomInt](#) in place of Rnd or the Random class.

## Date, Time and Intervals

Now available in [Xojo.Core.Date](#).

The [Date](#) class is complemented by the [TimeZone](#) and [DateInterval](#) classes.

A Date object cannot be modified. To get a new date, you create a new Date object or create a DateInterval to add or subtract to an existing Date object.

A non-modifiable Date object is more reliable than one that can be modified. With the Classic Date class, you could modify some of the date properties, but you had to change the properties in the "right" order or the date value would not be what you expect. It was also a common issue for people to change a FolderItem.ModifiedDate by manipulating the date properties. But this never actually would change the modification date of the file. Instead you

had to create a new date object and assign it to the ModifiedDate property. This new design ensures that behavior.

Dim d As New Date	Dim d As Date = Date.Now Or just directly use <a href="#">Date.Now</a>
Ticks	<a href="#">System.Ticks</a>
Microseconds	<a href="#">System.Microseconds</a>

## JSONItem

JSON usages is done through two simple methods: [Data.GenerateJSON](#) and [Data.ParseJSON](#). All JSON data is processed through Dictionaries or arrays. To create JSON data, you populate a Dictionary and then GenerateJSON from it.

To load JSON data into a Dictionary, you call ParseJSON.

## Graphics

These classes are used when dealing with graphics on iOS: [iOSGraphics](#), [iOSBitmap](#), [iOSImage](#) and [iOSPath](#).

DrawString	<a href="#">DrawTextBlock</a> <a href="#">DrawTextLine</a>
DrawPicture for image scaling	<a href="#">DrawImage</a> <a href="#">Scale</a>
StringHeight StringWidth	<a href="#">TextBlockSize</a> <a href="#">TextLineSize</a>
DrawPolygon, FillPolygon	<a href="#">DrawPath</a> <a href="#">FillPath</a>

## MsgBox

On iOS, you cannot show a modal MsgBox. Instead you use the [iOSMessageBox](#) class to display a message and asynchronously get an event when a button is clicked. Your code is not stopped while an iOSMessageBox is displayed and you have to use its Action event handler to determine which button was pressed.

## Variant to Auto

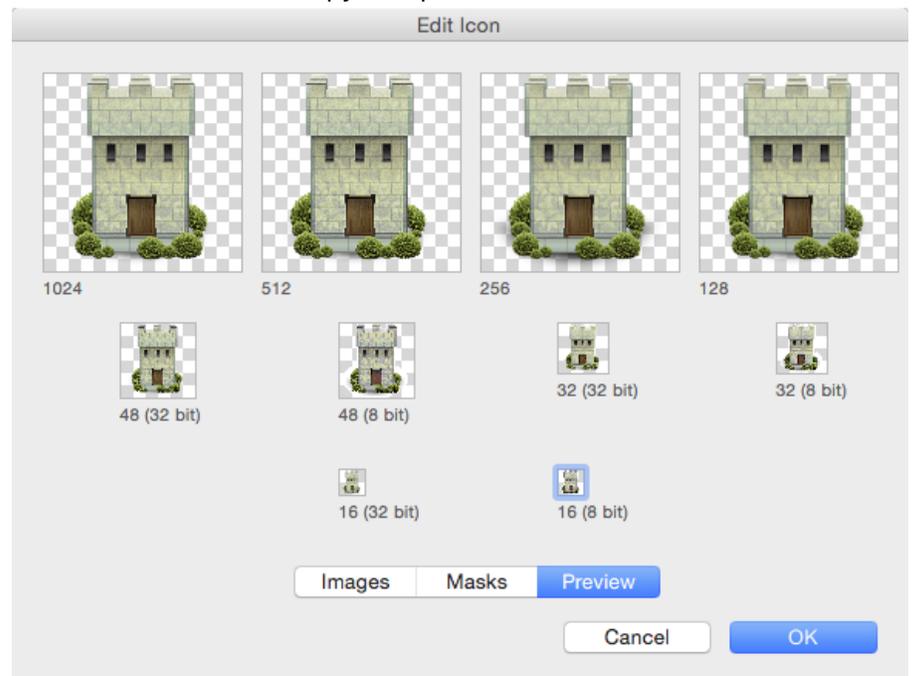
To avoid the many bugs that are inadvertently created by implicitly converting between types using Variant, the [Auto](#) data type has been added. Auto is a type that can contain a value of any type. This means it can hold intrinsic types such as Integer or Text, but also classes such as FolderItem or any of your own.

Auto does not do any implicit conversions. In order to use a value that is in Auto, you will want to assign it to a variable of the specific type first. You can use Introspection to get the type of any value that is in an Auto variable.

# App Icons

The Icon Editor is used to add application icons to desktop and web apps. Desktop app display the icon in the file viewer (Finder, Explorer, etc.) and in other places (such as the Dock on OS X or the Task Bar on Windows). Web applications display the application icon on the app's loading screen.

To use the Icon Editor, select the App object in the Navigator and then click the icon next to the Icon property in the Inspector. The Icon Editor displays as a window with blank areas for icons of various sizes. You can drag icons files to these areas to set the icon for that size. You can also copy and paste icons between the sizes; they are scaled automatically for you. If you copy and paste an icon, be sure to also copy and paste its mask.



Use the Images/Masks/Preview control to display the various components of the icon images. If you drag in an icon that has both the image and the mask (such as with most PNG files) then both the image and mask are added.

For best results, be sure to add icons (and masks) for every available size. Different platforms use different sizes. If the masks are missing, the icons may not appear on some platforms.

You can also drag an [ICNS](#) file onto the window background (instead of an image area) to load and populate all the image sizes at one time.

For iOS Apps, refer to [Launch Images and App Icons](#).

# Web App Deployment Overview

There are two ways to deploy web apps created with Xojo: Standalone and CGI. You can deploy your web apps to Virtual Private Servers (VPS) with proper configuration. Shared Hosting typically does not work due to general lack of configurability. For the simplest hosting and deployment experience, you should consider using [Xojo Cloud](#).

## Table of Contents

- [Deployment Options](#)
  - [Standalone](#)
  - [CGI](#)
  - [Xojo Cloud](#)
- [Platforms](#)
  - [Linux](#)
  - [Windows](#)
  - [OS X](#)
- [Test Apps for Deployment](#)
- [Troubleshooting](#)

## Deployment Options

### Standalone

A Standalone web app is an app that you manually run on your server. You have to start the app (usually from the command line) and leave it running in order to people to access the web app. In addition, a Standalone web app is accessed through a port, which you specify when building the app. Essentially, a standalone web app consists of both the web server and your web app.

A deployed standalone web app would be accessed with a URL such as this:

```
http://www.mywebsite.com:8080
```

### CGI

A web app built to use CGI uses Apache as its web server. The web server then communicates to your web app using CGI. To facilitate this, a companion Perl script (supplied when you build your app) handles communication between the web server and your web app.

Some web browsers (notably Safari) may continue to display a loading indicator even after the web page has finished loading. This is a result of the method used by the web server to communicate with your web app.

Because a CGI deployment uses your existing web server software, you do not have to specify a port when accessing your web app. A typical URL looks like this:

```
http://www.mywebsite.com/cgi-bin/mywebapp.cgi
```

---

## Xojo Cloud

Xojo Cloud is for those that want the fastest and easiest way to deploy web applications. Xojo Cloud is a fully managed, highly secure hosting environment for your Xojo web apps and has these features:

- Excellent security
- One-click deployment directly from Xojo
- Support for SFTP
- MySQL
- PostgreSQL
- Unlimited app deployments (memory permitting)

## Platforms

If you decide not to use Xojo Cloud, your web app can be compiled for any of the platforms supported by Xojo. When it comes to web servers, Linux is the most commonly used operating system, followed by Windows and then OS X.

### Linux

In complete opposition of the situation on the desktop, the majority of web servers use some form of Linux. The two most common types of hosting are shared and VPS (Virtual Private Server). Shared hosting usually costs less, but is also often rather restricted. Most shared hosting providers work best with static web sites or pre-configured tools (such as WordPress) and do not allow general purpose applications to run on them. They are rarely a good choice for a Xojo web app.

Your best bet is to use a VPS (Virtual Private Server) to host your Xojo web apps. A VPS gives you your own server, usually running the Apache web server, with its own specs running inside of a Virtual Machine (VM). With a VPS you have complete control over the server and can configure it to run Xojo web apps.

### Windows

Windows web servers primarily use IIS (Internet Information Server), but they can also run Apache. Windows servers are far less common than Linux servers. There are also fewer hosting companies offering Windows servers and they usually cost more. It is much more difficult to configure IIS for Xojo web apps than Apache.

### OS X

OS X web servers typically use Apache and setup is mostly the same as it is for Linux. There are even fewer OS X servers than Windows, but OS X servers can be simpler to configure. There are no known hosting companies that offer OS X hosting, but there are several that offer colocation services for your own Mac hardware.

## Test Apps for Deployment

To help you test deployment of Xojo apps on your web server, here are several versions of a simple compiled app for you to try.

---

OS Type	Deployment Type	Operating Systems	Download
32-bit	Standalone	Linux, Windows, OS X, Raspberry Pi	<a href="#">Download</a>
32-bit	CGI	Linux, Windows, OS X	<a href="#">Download</a>
64-bit	Standalone	Linux, Windows, OS X	<a href="#">Download</a>
64-bit	CGI	Linux, Windows, OS X	<a href="#">Download</a>

## Troubleshooting

Here are some troubleshooting tips:

- Verify that you compiled your web app as Standalone or CGI depending on how you are trying to access it.
- Verify that you compiled your web app for your Server platform and not your desktop platform.
- Be sure to set your Application Identifier to a unique value.
- Check the permissions for the files and folders containing your web app.
- Ensure that the web app and the libraries in the Libs folder are all set to be executable.
- For CGI web apps, ensure that you add "AddHandler cgi-script .cgi" somewhere in your Apache configuration or .htaccess file.
- For standalone apps or CGI apps that are not set to "Choose Port Automatically", make sure that the port you are using is open and available and that the web app is actually running. Port numbers should be  $\leq 65535$ . If you are not a privileged user, the port should be  $\geq 1024$ .
- Always check your server logs to see if they have additional information.

---

# Linux Web App Deployment

## Table of Contents

- [Introduction](#)
- [Standalone Web App](#)
- [CGI Web App](#)
- [Troubleshooting](#)

## Introduction

Web applications are often much simpler for your users and customers to use than desktop apps. They don't have to worry about installing any software or dealing with updates because now you do. Instead of your customers deploying your software on their computers, you have to deploy the software on your web servers.

In general, these are the steps to deploy a web app to a Linux server:

1. Build your web app for Linux.
2. Connect to your web server using an FTP client of your choice (using **Binary mode**).
3. Upload your web app (including the Libs and Resources folders)
4. Verify that the execute flag is still set for the files that you just uploaded. Some FTP clients have been known to change this flag during upload.

But the specifics can be trickier.

You can deploy your web apps to Virtual Private Servers (VPS) with proper configuration. Shared Hosting typically does not work due to general lack of configurability. For the simplest hosting and deployment experience, you should consider using [Xojo Cloud](#).

## Standalone Web App

If you are uploading a standalone web app then you should follow these steps:

1. Build your web application for Linux with the Standalone build setting selected.
2. Connect to your web server using the FTP client of your choice.
3. Navigate to the folder where you want your web application.
4. Upload your web app (including the Libs and Resources folders) in **Binary mode**.
5. Verify execute flag.

Now you need to run the standalone web app on the server. You'll need to connect to the server in order to do this, using ssh (Secure Shell). Secure Shell (and the ssh command) are available from the terminal or command line or Windows, OS X and Linux.

1. Connect to the server using ssh (secure shell) with this command: `ssh login@mywebsite.co`
2. Enter the password when prompted.

3. Navigate to the folder where you uploaded your web app.
4. Start the app using this command: `./TestApp`

You can now access your web app in your web browser using the domain name and port. If "TestApp" was compiled to use port 8080, you can access it using a URL like this:

```
http://www.mywebsite.com:8080
```

Note that if you exit from ssh your web app will terminate unless you daemonize it. Use this code in the App.Open event to daemonize a standalone web application:

```
Call Daemonize
```

On OS X, the use of Daemonize is [discouraged](#) by Apple. You should use [launchd](#) instead.

## CGI Web Application

If you are uploading a CGI web app then you should follow these steps:

1. Compile your web app for Linux with the CGI build setting selected.
2. Connect to your web server using FTP.
3. Navigate to the folder where your web server expects to find CGI app. This is often called cgi-bin.
4. Upload your web app (including the Libs folder, Resources folder, config.cfg and cgi file). Use **Binary mode**.
5. Verify execute flag.

If your web server is already configured properly to run CGI apps using Apache, then you will only need to type the URL in your browser. For example, for a web app called "TestApp", you would use a URL like this:

```
http://www.mywebsite.com/cgi-bin/testapp.cgi
```

## Troubleshooting

Things don't always go this smoothly, however. This section has some tips for you as you troubleshoot.

### 32-bit Libraries

By default, 64-bit version of Linux do not include the 32-bit libraries that are needed by Xojo web app. Please refer to the [System Requirements](#) for details on how to apply the 32-bit libraries to your Linux distribution. Also make sure that the libicu library is available.

### Dependencies

Not all Linux installations have the necessary dependencies for Xojo apps pre-installed. Refer to the [System Requirements](#) for details. To determine which libraries are used by Xojo, you can use the **ldd** command:

```
ldd Xojo
```

## Web Logs

If you get an "Internal Server Error" or any error, you need to check the web server log for specifics. A common source of problems is incorrect permissions.

### Permissions

Your web app needs to have the correct permissions enabled so that the web server can run the application. This is done using the `chown` and `chmod` commands. The necessary permissions vary by the Linux distribution and Apache installation.

### Unable to Launch Application

This error indicates that the CGI (Perl script) has executed properly, but was unable to launch your web application. More information will follow this error, which is usually a "Permission denied" message. Some shared hosting users receive a "Connection refused" message. This message usually means the hosting provider has a security policy that prevents execution of your app. Check with your hosting provider if you receive this message.

### Executable Settings

When uploading your application to the server, ensure the app and its `libs` folder are uploaded in **binary mode**, everything else in ASCII mode. Many FTP clients get this correct automatically, but some do not.

If you get an error that says "Application launched, but unable to connect on port #", the CGI was able to launch your app, but was unable to connect to it using the port specified. Your host's firewall may be blocking this port. You will need to deploy using a static port, and ask your host to open a port for you to use.

### Unable to Locate a Library in Libs Folder

First, ensure you have uploaded all the contents of the `Libs` folder in **Binary mode**.

Check the `Options FollowSymLinks` in your Apache configuration.

### Internal Server Error

This can many any number of things. Check your web server logs. Some things to verify:

- Ensure Apache is configured to allow CGI apps
- Ensure Perl is installed and properly configured

# Drag & Drop

 Available with **2015r3**

You can now create pages with controls that can be dragged onto other controls. You do this using the new methods and events on `WebControl` in conjunction with the `WebDragItem` class.

## Supported Controls

These are the controls that can be dragged:

- [WebLabel](#)
- [WebLink](#)
- [WebContainer](#)
- [WebImageView](#)
- [WebRectangle](#)
- [WebCanvas](#)
- [WebControlWrapper](#)

These are the controls that can have another control dropped on them:

- [WebTextField](#)
- [WebTextArea](#)
- [WebSearchField](#)
- [WebLabel](#)
- [WebLink](#)
- [WebContainer](#)
- [WebImageView](#)
- [WebRectangle](#)
- [WebCanvas](#)
- [WebControlWrapper](#)

If you try to use drag & drop with a control that does not support it, a message is written to the console (and displayed in the Messages pane).

## Dragging a Control

To allow a control to be dragged, you call one of the `Allow` methods on the control, typically in the `Shown` event handler. A control can support dragging of pictures, text and any raw data. This code enables text dragging for a label:

```
Me.AllowTextDrag(WebDragItem.DragActionCopy)</code>
```

The parameter specifies the type of drag. There are several types on the `WebDragItemClass`: `DragActionCopy`, `DragActionMove`, `DragActionLink`. These types do not cause the drag to work differently but you can check the type when the drop occurs to do a specific action. The type you choose does change the type of cursor the browser displays for the drag. As example, if you used `DragActionMove`, you may want the control accepting the drop to hide the original control so it cannot be dragged again.

You should only specify a single drag type on the source control. This specifies the control can be dragged as a copy:

```
Me.AllowTextDrag(WebDragItem.DragActionCopy)</code>
```

Dragging controls around the page by is not useful by itself. You'll also want to allow the controls to be dropped onto other controls. You do this by calling the Accept methods for the control, again typically in the Shown event handler. A drop can support 1 or more drag types. This code allows a copy or move drop to occur on a text area:

```
Me.AcceptTextDrop(WebDragItem.DragActionCopy + WebDragItem.DragActionMove)</code>
```

Now when the control is dragged, you will be able to drop it onto this text area. When you do so, it calls the DropObject event handler for the text area (or whatever control you are using). This event provides you with the WebDragItem which contains information about the control that was dropped.

Xojo includes two examples to help you see how Web Drag & Drop works:

- Web/Drag and Drop/SimpleDragAndDrop
- Web/DragTest

## Known Limitations

- Internet Explorer 9 does not show drag previews or specific drag actions.
- WebListBox does not support dragging or dropping.
- You cannot drag or drop items to or from the desktop or other apps. Only dragging and dropping within the page is supported.

# Web App Security

We take the security of your web application very seriously in the Xojo web app framework. Because web apps are accessible to any number of online users, their security is paramount.

Most traditional web development languages are interpreted, meaning your web app is a set of files on a server. If someone gains access to that server, they gain access to your source code. Xojo web apps are compiled to binary code so your source code is not stored on the server.

In order for someone to alter your application they would have to be very familiar with x86 assembly code and be willing to expend a serious amount of effort into tracing through that code. This is, at the least, an order of magnitude far more difficult than hacking HTML, JavaScript, CSS, AJAX, and PHP or Java source code.

The Open Web Application Security Project (OWASP) provides information on web application security and recently posted a list of the top 10 web application security issues. While a few of these issues require the developer to be more diligent, most cannot be used to hack into a web application created with Xojo.

Hack	Xojo Protection
SQL Injection Attacks	The Xojo framework has PreparedStatement classes for all supported databases which use database binding to prevent SQL injection.
Cross-Site Scripting	Xojo web apps can't be used for this purpose because all data sent to the browser is automatically escaped. As a result, the user cannot inject HTML into a page. Also, because the developer doesn't work in HTML or JavaScript, there's no way for the developer to accidentally create this security breach.
Application Authentication	Xojo does not have authentication routines to compromise and session tokens are automatically protected from theft.
Insecure Direct Object References	Xojo does not allow direct object references in this manner so it would be impossible for such a security hole to be created.
Cross-Site Request Forgery	When the user logs into a web site (such as a banking site) and then leaves by navigating to a page of another site without first logging out, the original site will still see the user is logged in until their session times out. The developer can mitigate this by reducing the timeout from the 60 second default.
Security Misconfiguration	This involves the developer making sure they have good passwords for their server, not exposing data that does not need to be exposed, etc. This particular security concern is completely within the control of the developer and is outside the scope of what any development tool can guard against. With that said, <a href="#">Xojo Cloud</a> is a fully managed secure way to host your Xojo apps.
Insecure Cryptographic Storage	Be sure to use appropriate function in the Crypto module to properly secure your data.
Failure to Restrict URL Access	Because Xojo web apps create the HTML page on the fly, there's no way for a hacker to access any page except the one that is currently in their browser. However, if the developer chooses to support bookmarking, they would need to make sure they authenticate the user before taking the user to the requested page.

---

<b>Hack</b>	<b>Xojo Protection</b>
Insufficient Transport Layer Protection	Web Servers provide SSL support which is the appropriate place to handle this issue.
Unvalidated Redirects and Forwards	There is nothing any development tool can do to prevent this. It's up to the developer to make sure their app doesn't depend on untrusted data when redirecting or forwarding the user to another site. For example, the developer should always use the EncodeURIComponent function to encode any values used in a URL which come from a user or database.



---

# Add Web Apps to iOS Home Screen

To make it easier to launch Xojo web apps, you can add them to your iOS device's home screen.

To add a web app to the home screen, you open it using Mobile Safari as you normally do and then click the "Sharing" button and choose "Add to Home Screen". This creates an app on your home screen that uses the icon you specified for the web app. When you click on the icon on the home screen your app launches without any of the usual Mobile Safari controls.

However, if you have an iPhone 5 (or later) you will notice that the web app does not use the full screen height if you launch the app from the home screen. There are black bars at the top and bottom of the screen.

It seems that this might be a bug in iOS, but there is a workaround. You need to override the viewpoint so that it does not use the default setting.

In the case of a Xojo web app, this means using the `WebSession.PrepareSession` event handler. In it you can add code like this to specify a new viewport:

```
Select Case Me.Platform
Case WebSession.PlatformType.iPhone
  // Allow iPhone 5 (and later) to scale web app to use entire screen
  // height when it is added to the home screen
  HTMLHeader = "<meta name=""viewport"" content=""initial-scale=1.0"">"
End Case
```

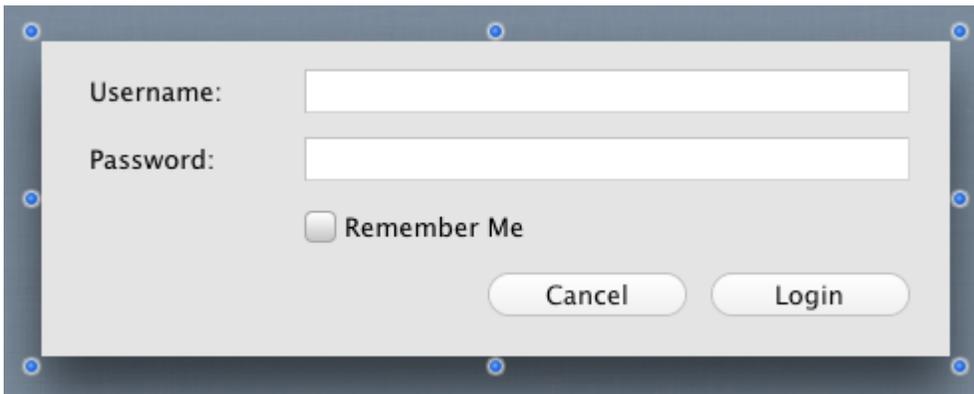
## Secure Login Screens

More and more customers are using Xojo to create applications that may contain private end-user information. This means that it's becoming increasingly important that your web applications be protected with an SSL certificate and some kind of authentication to make sure the user is who they claim to be.

Note: To follow along, you will need to build a CGI application and run it on a server that has an SSL certificate installed. Otherwise, you'll get an error.

Traditionally, a login screen requests a UserID of some kind (whether an email address or other unique identifying identifier of some kind, like an account number) and a password (automatically generated or entered by the end-user).

A good method for creating login screens in Xojo is to use a Modal Dialog on an otherwise empty page, because there is no question about what you need the user to do next. It also allows you to deal with the complexities of the login process without any other code possibly getting in the way.



Something to remember: SSL connections are notably slower than non-SSL connections because no caching is allowed, so you might not want your whole web application to be secure (like if you were creating a publicly accessible site with special features for members-only). What you need to be able to do is to switch from non-SSL to SSL when a user gets to the login screen.

## Making the Switch

In this example, the login screen is simply a blank web page that shows a web dialog in the Shown event. Before you get to that point, you need to check to see if the connection is secure. In the LoginPage.Open event:

```
' Make sure the user is connecting securely,
' If not have the browser load the application securely
If Not Session.Secure Then
    Dim host As String = Session.Header("Host")
    Dim url As String = "https://" + host
    ShowURL(url)
End If
```

Okay great, but this will take the user back to the default web page, not the login page, so you need to tell the

session how to handle that. In the `WebSession.Open()` event:

```
If Self.Secure Then
    LoginPage.Show
End If
```

In this case the code makes an assumption that an end-user that is just connecting to the app (i.e. creating a new Session) and is connecting securely will want to be directed to the login page.

## Remember Me

If you look at the screen shot, there is a "Remember Me" checkbox. Typically these are used to remember the UserID to help jog the end-user's memory about what their password might be. This function uses a browser feature called Cookies which stores a snippet of information on the computer where the user is (assuming cookies are enabled, of course).

**NOTE:** It is best to not remember the user's password because if this box gets checked on a public computer (like a Library or Internet Cafe) the user's account could be compromised.

To make the checkbox work, you need two pieces of code - in `LoginDialog.Shown`:

```
If Session.Cookies.Value("username") <> "" Then
    UserNameField.Text = Session.Cookies.Value("username")
    RememberMeCheck.Value = True
    PasswordField.SetFocus
End If
```

First, check to see if the "username" cookie has been set, and if it has, place the value into the Username text field, check the RememberMe checkbox and set the focus to the Password field (mostly a convenience to the users don't need to do that manually).

Next, in the `Action()` event of the Login button:

`DoLogin`

Then create the `DoLogin` method on `LoginDialog`:

```
If RememberMeCheck.Value Then
    Session.Cookies.Set("username", UserNameField.Text)
Else
    Session.Cookies.Remove("username")
End If
```

' Now validate the credentials and if they are OK, you can close the dialog  
' and show the main page.

```
If ValidateCredentials Then
    Self.Close
```

```
MainPage.Show  
Else  
    MsgBox("The username/password combination does not match.")  
End If
```

You may be asking yourself why should you create a separate method for what would normally be in the Action event? The reason is because you may want to close this dialog from another location (see below), but first, when the user clicks the login button, you check the value of the RememberMe checkbox - If True, set the Username Cookie to the what was typed as the user name. If False, remove the cookie (otherwise users wouldn't be able to remove the cookie by unchecking the box).

## Capturing Returns

One last thing. Users expect things to work just like all the other web applications out there, and that means that when they press the Return key on their keyboard, they usually expect the login action to occur. You can do this by adding code to the KeyPressed event of the window itself:

```
If Details.KeyCode = 13 Then  
    LoginDialog1.DoLogin  
End If
```

Which is why you put the code from the Action event into a separate method! Now, when the user types their username and/or password and hits the Return key, the dialog will be dismissed just as if they had clicked the Login button themselves!

# Windows Service

To run an app in the background on Windows, you use a Service. Only console (non-GUI) apps can be used as Service apps. Since web applications are actually console apps that are a subclass of ServiceApplication, they can easily be made to run as a Windows Service. The only tricky part is actually installing it as a Windows Service.

Windows has a built-in command for this, `sc` which stands for "service control".

Here are the steps to install your standalone web app as a Windows Service:

1. Build your app as a Standalone Web App and place the app in an accessible location. Be sure to note the port number that you used when building the app. You'll want to be sure to use a port number that is not already in use on the system. An example might be 8104.
2. Start the Windows Command line app with administrator privileges
3. Use the `sc` command to install the service as follows:
  - `sc create RSWebSvc type= own start= auto binpath= c:\Path\To\Exe\WebApp.exe`
4. After you press Return you should see: `[SC] CreateService SUCCESS`
5. Now open up Control Panel and go to the Services Manager. It is located in the Administrative Tools section and is called Services.
6. Find the service you just created. It will be listed by its app name. Click on it and select Start.

That's it. Your web application is now running as a service. To test it, navigate to the URL in your browser:

```
http://localhost:8104
```

To stop the service, click on it in Service Manager and click Stop.

# Linux Daemon

A daemon is essentially a background process. You can use a daemon easily to deploy your Standalone web applications to remote web servers.

Here are steps to create a Standalone web app that can run as a daemon:

1. Add a single line of code to the App.Open event handler (or App.Run for a Console app):
  - Call `Daemonize`
2. Now build the app as Standalone and select a port number that is not in use on the machine. An example might be 8104. Copy the app to an accessible location.
3. Open the Terminal and navigate to the location of your app.
4. Start the app:
  - `./MyWebApp`
5. You will immediately return to the command line because the app is running as a daemon.

6. Verify the daemon is running using the `ps` command:

- `ps -C MyWebApp`

That's it. Your application is now running as a daemon. To test it, navigate to the URL in your browser:

```
http://localhost:8104
```

If you are remotely connected to a Linux server (using SSH), then disconnecting will also quit an app started in this manner. Instead you'll want to either start the app using the `"&"` suffix or using the `nohup` command.

# Xojo Web App Security

Because web apps are accessible to any number of online users, the security of web apps is paramount. Xojo web apps are serious about security.

Most traditional web development languages are interpreted, meaning your web app is a set of files on a server. If someone gains access to that server, they gain access to your source code. Xojo web apps are compiled to binary code so your source code is not stored on the server.

In order for someone to alter your app they would have to be very familiar with x86 assembly code and be willing to spend a lot of time tracing through that code. This is, at the least, an order of magnitude far more difficult than hacking HTML, JavaScript, CSS, AJAX, and PHP or Java source code.

The Open Web Application Security Project (OWASP) provides information on web app security and [posted a list of the top 10 web app security issues](#). While a few of these issues require the developer to be more diligent, most cannot be used to hack into a web app created with Xojo.

Hack	Xojo Protection
SQL Injection Attacks	Xojo provides developers with <a href="#">prepared statement</a> support for database access. This takes the values to be used in a query and sends them separately to the database server so that it can determine if the values are valid or contain SQL.
Cross-Site Scripting	Applications created with Web Edition can't be used for this purpose because all data sent to the browser is automatically escaped. As a result, the user cannot inject HTML into a page. Also, because the developer doesn't work in HTML or JavaScript, there's no way for the developer to accidentally create this security breach.
Application Authentication	Xojo does not have authentication routines to compromise and session tokens are automatically protected from theft.
Insecure Direct Object References	Xojo does not allow direct object references in this manner so it would be impossible for such a security hole to be created.
Cross-Site Request Forgery	When the user logs into a web site (such as a banking site) and then leaves by navigating to a page of another site without first logging out, the original site will still see the user is logged in until their session times out. The developer can mitigate this by reducing the timeout from the 60 second default.
Security Misconfiguration	This involves the developer making sure they have good passwords for their server, not exposing data that does not need to be exposed, etc. This particular security concern is completely within the control of the developer and is outside the scope of what any development tool can guard against.
Insecure Cryptographic Storage	Be sure to use appropriate function in the Crypto module to properly secure your data.

<b>Hack</b>	<b>Xojo Protection</b>
Failure to Restrict URL Access	Because Xojo web apps create the HTML page on the fly, there's no way for a hacker to access any page except the one that is currently in their browser. However, if the developer chooses to support bookmarking, they would need to make sure they authenticate the user before taking the user to the requested page.
Insufficient Transport Layer Protection	Web Servers provide SSL support which is the appropriate place to handle this issue.
Unvalidated Redirects and Forwards	There is nothing any development tool can do to prevent this. It's up to the developer to make sure their app doesn't depend on untrusted data when redirecting or forwarding the user to another site. For example, the developer should always use the <a href="#">EncodeURLComponent</a> function to encode any values used in a URL which come from a user or database.

# Getting Started with Raspberry Pi

Raspberry Pi is essentially a tiny, inexpensive computer (about US\$35). Because it is tiny, it can be used in all kind of things that a typical computer does not work well with, such as robotics and embedded systems. Since it is also inexpensive, it functions as a great learning tool, making it possible for anyone to have their own computer. Additionally, because a Raspberry Pi is a fully functional computer, with input/output, storage and wifi capabilities, it can also be used to interface and control other things. This makes the Raspberry Pi a favorite amongst tinkerers, Makers, electronics hobbyists and anyone else with a cool idea for a project.

What makes the Raspberry Pi somewhat unique is that it uses an ARM processor (similar to what you see on cell phones and tablets), rather than an Intel CPU that is in most things typically categorized as "computers".

Still, a Raspberry Pi is a full computer. You can connect a keyboard, mouse and display to it. You can plug in storage and install an operating system on it (typically Linux). Now it is not a powerful computer, to be sure, but it is still powerful enough for many tasks.

Xojo can create desktop, web and console apps that run on Raspberry Pi 2 (32-bit Linux ARMv7).

## Useful Links

- [RaspberryPi.org](http://RaspberryPi.org)
- [CanaKit on Amazon](#)
- [CanaKit](#)
- [AdaFruit](#)
- [RobotStore](#)
- [SparkFun](#)
- [MagPi Magazine](#)

## Creating a Raspberry Pi App

Creating Raspberry Pi app is no different than creating an app for other targets (Windows, OS X or Linux). Select the type of project you want from the Project Chooser and then when you are ready to build for Raspberry Pi, click on **Linux** in **Build Settings** and change the **Architecture** property in the Inspector to "ARM 32-bit". Build your app and then transfer the built folder to the Raspberry Pi.

## Desktop Apps

To use HTMLViewer in a desktop app, you first have to install libwebkitgtk:

```
sudo apt-get install libwebkitgtk-1.0.0
```

## Web Apps

The Raspberry Pi web browser identifies itself as Safari 6.0 on Macintosh with an ARM processor. In Xojo web apps you can check for this using [WebSession](#) properties:

```
If Session.Browser = WebSession.BrowserType.Safari And _  
  Session.Platform = WebSession.PlatformType.Macintosh And _  
  Session.Header("User-Agent").Instr("ARM") > 0 Then  
  ' This is the Raspberry Pi browser  
End If
```

## Some Raspberry Pi Tips

If you have the default installation of Raspbian (with the desktop), you'll be able to connect your Pi to any display (including a TV) using HDMI. Plug in a USB keyboard and mouse and you can use it as a real computer.

### Updating the OS

To update the OS, you can run these two commands from Terminal:

```
sudo apt-get update  
sudo apt-get upgrade
```

### Transferring Files to Raspberry Pi

By default, there is an SFTP server installed. You can connect to this using any SFTP client. You'll need to know the IP address of the Raspberry Pi on your network and the username and password that you created when you initially installed Raspbian ('pi'/'raspberrypi' by default).

### Connecting to Raspberry Pi via Shell

You can also use ssh without any Pi configuration. Start ssh with a command such as this:

```
ssh pi@10.0.1.194
```

and then enter your password ('raspberrypi' by default). You'll now be connected to the Pi in a terminal window. Here you can easily run Xojo apps that you've transferred using SFTP.

## Connecting to Raspberry Pi via VNC

VNC (aka Remote Desktop) allows you to view the desktop of the Pi and interact with it using your mouse. This is not necessary for testing Xojo apps because they are console only, but it can be useful to more easily access configuration screens or use the Pi in some other fashion.

In order to use VNC, you first need to install a VNC server on the Raspberry Pi. TightVNC seems to be the preferred one and instructions are here:

- [VNC \(Virtual Network Computing\)](#)

This command starts the VNC server with a 1080p screen size and 24-bit color:

```
vncserver :1 -geometry 1920x1080 -depth 24
```

Once you have TightVNC server installed and running you can connect to it using any VNC client (Windows/Mac/Linux). Some suggestions:

- OS X Screen Sharing
  - Finder, Go->Connect to Server, vnc://10.0.1.194:5901 (substitute your own IP address and port)
- Vine VNC
- Screens
- RealVNC

## Using GPIO

To work with [GPIO](#) (the General Purpose Input/Output port), you will need to [install the wiringPi library](#). After doing so you can use the WiringPiXojo module to communicate with the GPIO port on the Raspberry Pi. This simple code in the Run event handler of a Console app flashes an LED:

```
GPIO.SetupGPIO
```

```
Const kLEDPin = 4 // "#4" on the pinout
```

```
// Set the pin to accept output  
GPIO.PinMode(kLEDPin, GPIO.OUTPUT)
```

```
// Blink LED every 1/2 second  
While True
```

```
    // Turn the pin on (give it power)  
    GPIO.DigitalWrite(kLEDPin, GPIO.ON)  
    App.DoEvents(500)
```

```
    // Turn the pin off (no power)  
    GPIO.DigitalWrite(kLEDPin, GPIO.OFF)  
    App.DoEvents(500)
```

## Wend

GPIO apps usually have to be started with `sudo`, so if you built a console app called [LEDBlink](#) you would start it like this from terminal:

```
sudo ./LEDBlink
```

You can also create a desktop app using GPIO. To start a desktop app as `sudo`, you have to use the `gksudo` command from terminal:

```
gksudo ./LEDBlinkUI
```

Starting with Raspbian Jessie, you no longer have to use `sudo` to access GPIO. In order to not require `sudo`, you have to first set an environment variable before you run the app:

```
export WIRINGPI_GPIOMEM=1
```

Refer to these tutorials for step-by-step instructions on how to create a circuit and a Xojo app:

- [Blinking LED Tutorial](#)
- [Button LED Tutorial](#)

## Limitations

Currently these limitations exist for Raspberry Pi apps:

- The Remote Debugger is not yet available.
- XojoScript does not yet work.
- To use HTMLViewer you will need to install `libwebkitgtk 1.0` from the terminal:
  - `sudo apt-get install libwebkitgtk-1.0.0`

# GPIO Module

GPIO is the General Purpose Input/Output port on the Raspberry Pi. You can use this port to connect external hardware to the Pi.

The Xojo GPIO module maps the wiringPi library to a set of methods that you can call in your Raspberry Pi Xojo apps. In order to use the GPIO module, you'll need to [install the wiringPi library](#) on your Raspberry Pi.

The GPIO module is included with the Xojo examples and is located here:

- [Examples/Platform-Specific/Linux/RaspberryPi](#)

The official docs will not be replicated here, but links are provided for your reference:

- [Setup methods](#)
- [Core Functions](#)
- [Raspberry Pi Specifics](#)
- [Timing](#)
- [Priority, Interrupts and Threads](#)
- Serial (not yet implemented in Xojo GPIO module)
- [SPI Library](#)
- [I2C Library](#)
- [Shift Library](#)
- Software PWM Library (not yet implemented in Xojo GPIO module)
- Software Tone Library (not yet implemented in Xojo GPIO module)

# Blinking LED Tutorial

In this tutorial, you will create a simple circuit with an LED and will then make the LED blink using a simple Xojo app.

In order to build the circuit, you'll need some parts:

- 1 ribbon cable with Raspberry Pi GPIO connectors ([AdaFruit link](#))
- 1 cobbler ([AdaFruit link](#))
- 1 breadboard ([AdaFruit link](#))
- 3 wires ([AdaFruit link](#))
- 1 LED ([AdaFruit link](#))
- 1 10k resistor ([SparkFun link](#))

In order to make the LED blink with Xojo, you'll need to [install the wiringPi library](#) and grab the WiringPiXojo module, located here:

- [Examples/Platform-Specific/Linux/Raspberry Pi](#)

## Connect Ribon Cable to Pi

First, connect your ribbon cable to the Raspberry Pi. The white/black side is typically facing the side of the Pi that does not have connectors. There is not a lot of room to work so take your time and make sure the pins are all aligned before you push it down.



## Setup the Breadboard

A breadboard is the place where you will wire your circuit. The ribbon cable and cobbler are essentially used to extend the GPIO port pins to the breadboard. On the breadboard you can wire everything without soldering. You just plug things into the holes on the breadboard.

First, you'll want to plug your cobbler into the breadboard. The cobbler has to be in the center so that the left and

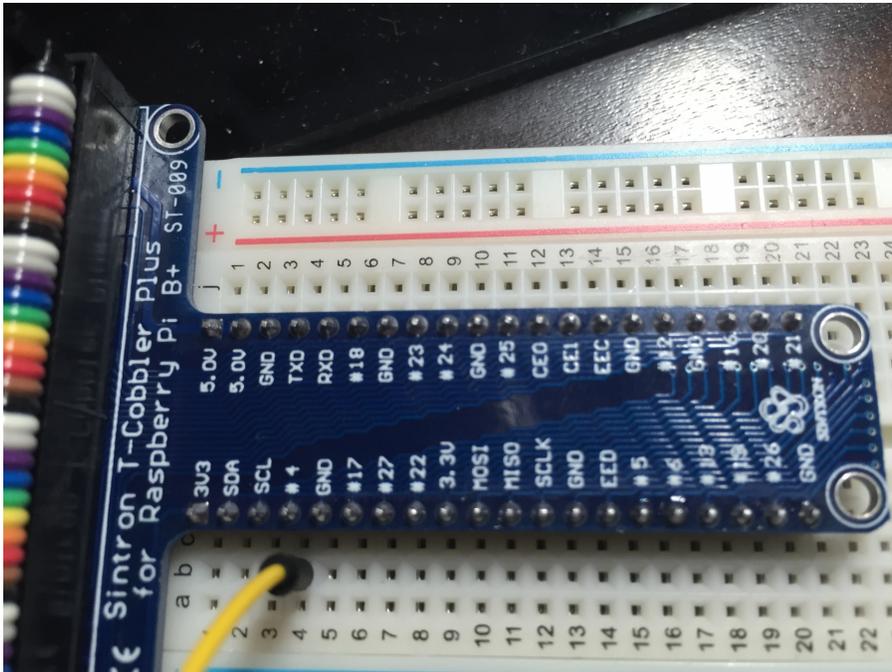


Each row on the breadboard that matches up to a pin on the cobbler is "connected" to the pin on the cobbler. Any wires you connect on that row will act as if they are connected to pin on the GPIO port. For example, looking at the cobbler, you can see that the pin marked "#17" is aligned with row 6 on the cobbler.

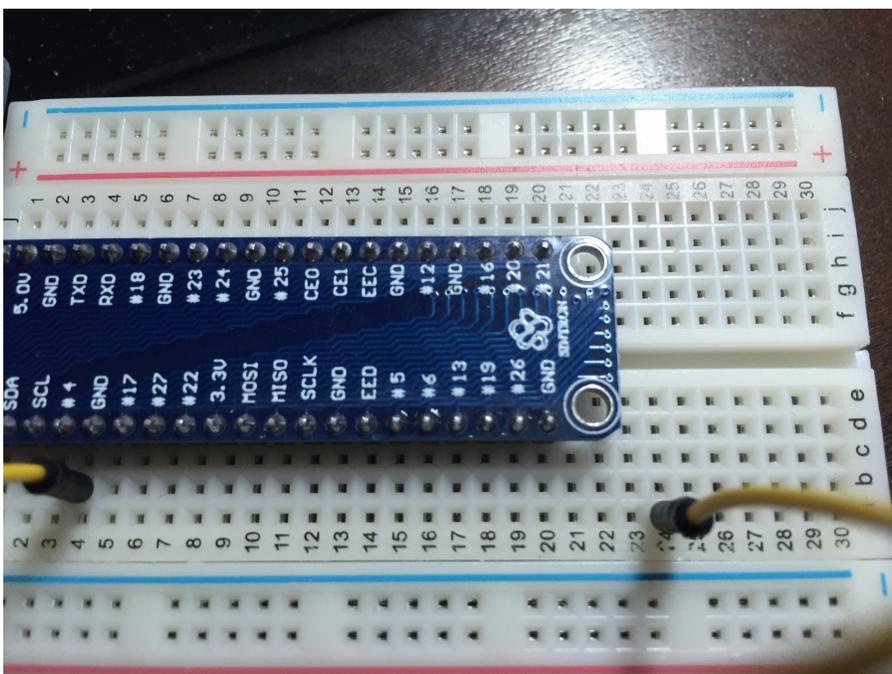
## Wire the Circuit

Now that you have the breadboard hooked up to the Raspberry Pi, you can start on building the blinking LED circuit. In this step you'll use 3 wires, 1 LED and a resistor.

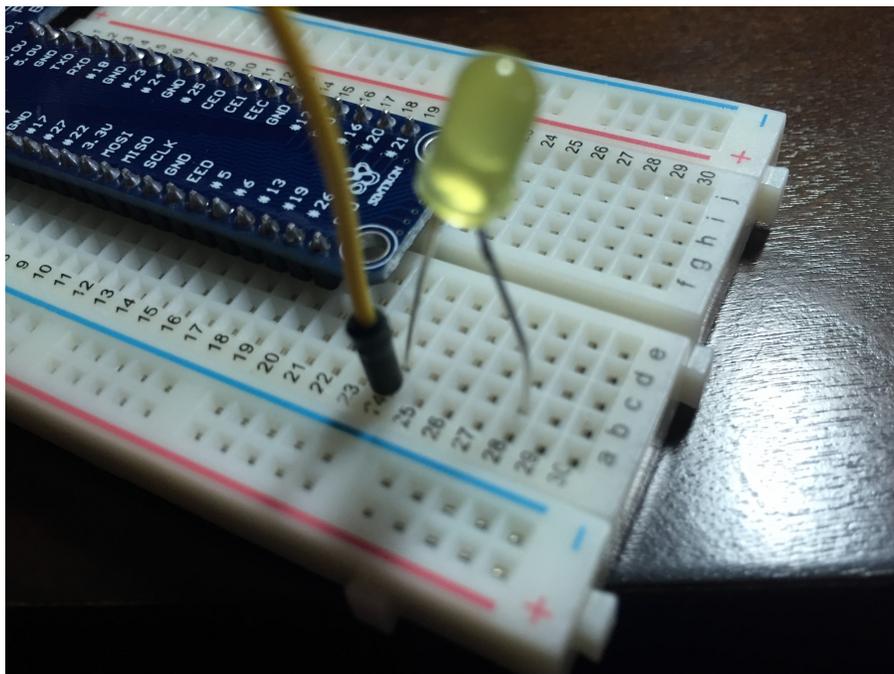
1. Connect a wire to the pin marked "#4". Yellow is used in this example.



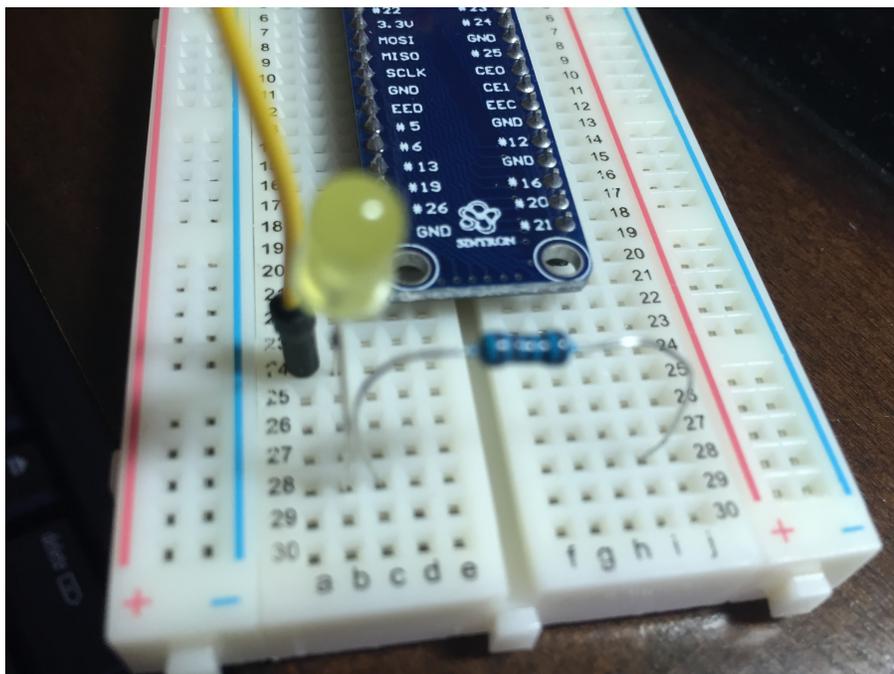
2. Connect the other end of the wire to an open spot on the board away from the cobbler. Here row 24 is used.



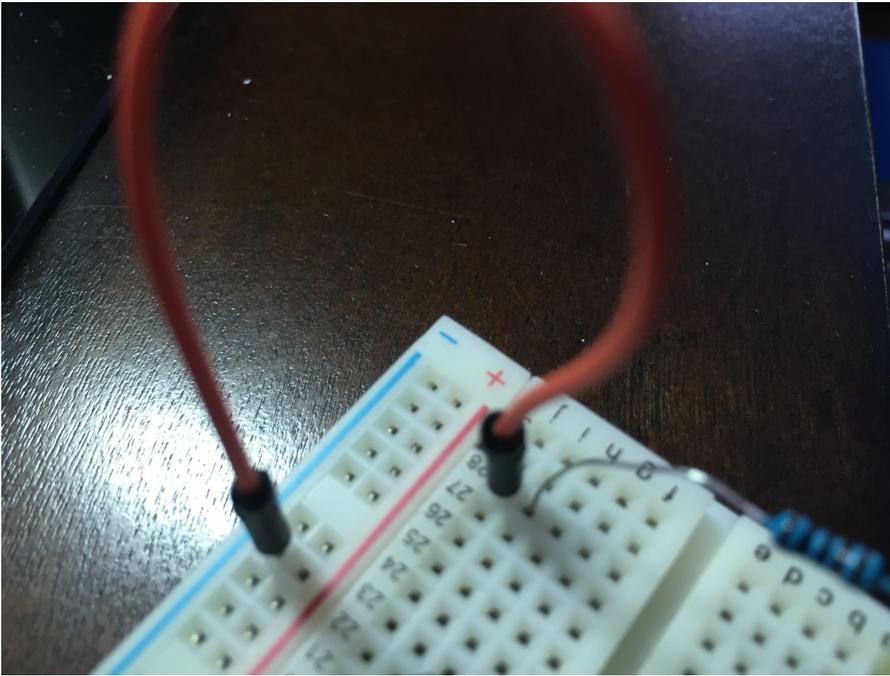
- Grab the LED. Note that one of the wires coming from the LED is longer than the other. The longer end is the positive (+) connector. Plug the long end into a hole on the breadboard adjacent to the wire (yellow) and the short end to an open row on the breadboard.



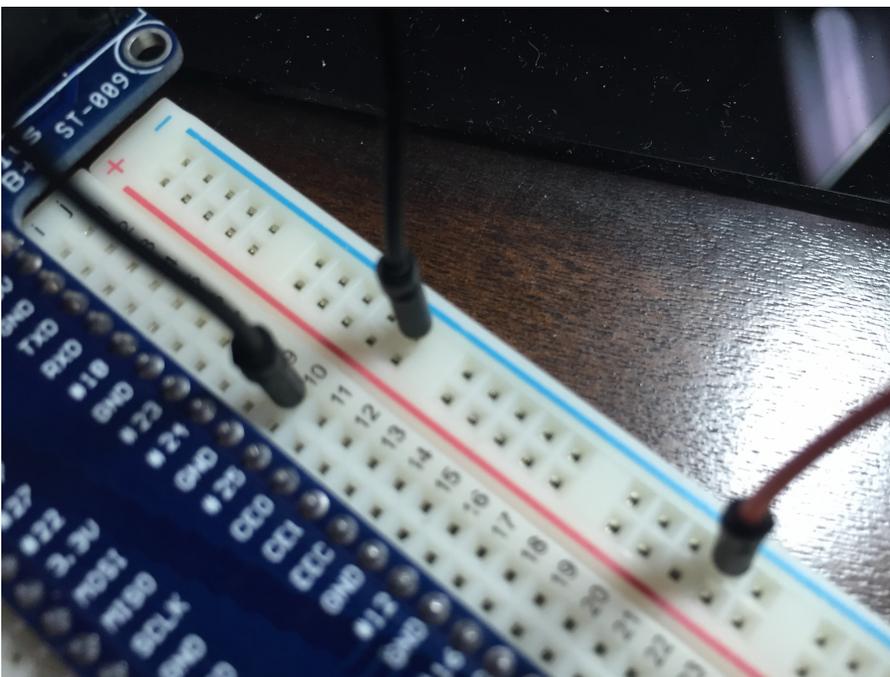
- A resistor is needed to prevent the power provided by the Pi from blowing out the LED. Take the 10k resistor and connect one end so that it is adjacent to the negative wire of the LED. Connect the other end of the resistor to an open spot on the board. If you cross over the center of the board to the other side, you can take advantage of the same row, which is what is done here.



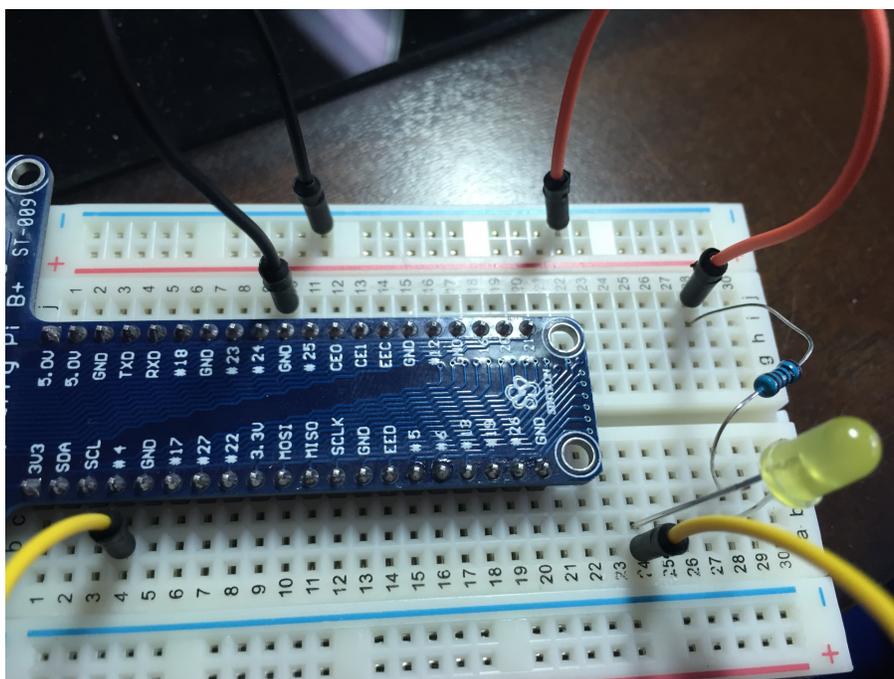
- Now connect another wire adjacent to the unconnected resistor wire. Connect this wire to the negative (-) column on the breadboard. This is usually the column on the outer edge.



6. Lastly, connect a wire from the negative column of the breadboard to a GND pin on GPIO. There are several GND pins; you can use any one that is easily reachable.



7. This completes the circuit.



You can quickly test the circuit by moving the yellow wire from pin #4 to pin 3v3. This directly connects the circuit to 3.3 volt power. The LED should immediately light up. Switch it back to the #4 pin before moving on to creating the Xojo app.

## Create the Xojo app

To test your circuit, you'll use a simple Xojo app that will alternate between ON (HIGH) and OFF (LOW) on pin #4. When the pin is ON (HIGH), the LED will illuminate.

1. Create a new Console project and call it LEDBlinker.
2. Add the GPIO module to the project.
3. Add this code to the App Run event handler:

```
GPIO.SetupGPIO
```

```
Const kLEDPin = 4 ' "#4" on the pinout
```

```
' Set the pin to accept output
GPIO.PinMode(kLEDPin, GPIO.OUTPUT)
```

```
' Blink LED every 1/2 second
While True
```

```
' Turn the pin on (give it power)
GPIO.DigitalWrite(kLEDPin, GPIO.ON)
App.DoEvents(500)
```

```
' Turn the pin off (no power)
GPIO.DigitalWrite(kLEDPin, GPIO.OFF)
```

```
App.DoEvents(500)
```

```
Wend
```

Build the app.



Before building, make sure you've selected **Linux** in Build Settings and its architecture to **ARM 32-bit** in the Inspector.

## Transfer and Run the Xojo app

Start your SFTP app and connect to the Raspberry Pi. Transfer the LEDBlinker folder from the Linux build folder to the Pi.

On the Pi, open the Terminal (or ssh to it from your development machine). In the terminal, navigate to the location of the LEDBlinker folder and then CD to the folder:

```
cd LEDBlinker
```

Run the app:

```
sudo ./LEDBlinker
```

 Normally sudo is needed to access GPIO. The GPIO.SetupGPiOSys method does provide access to some GPIO functionality without requiring sudo.

The LED should start blinking. To stop the app, press Control-C.

Note that if you press Control-C while the LED is on, it will remain on.

Starting with Raspbian Jessie, you no longer have to use sudo to access GPIO. In order to not require sudo, you have to first set an environment variable before you run the app:

```
export WIRINGPI_GPIOMEM=1
```

The Xojo project is included with the Xojo examples and is located here:

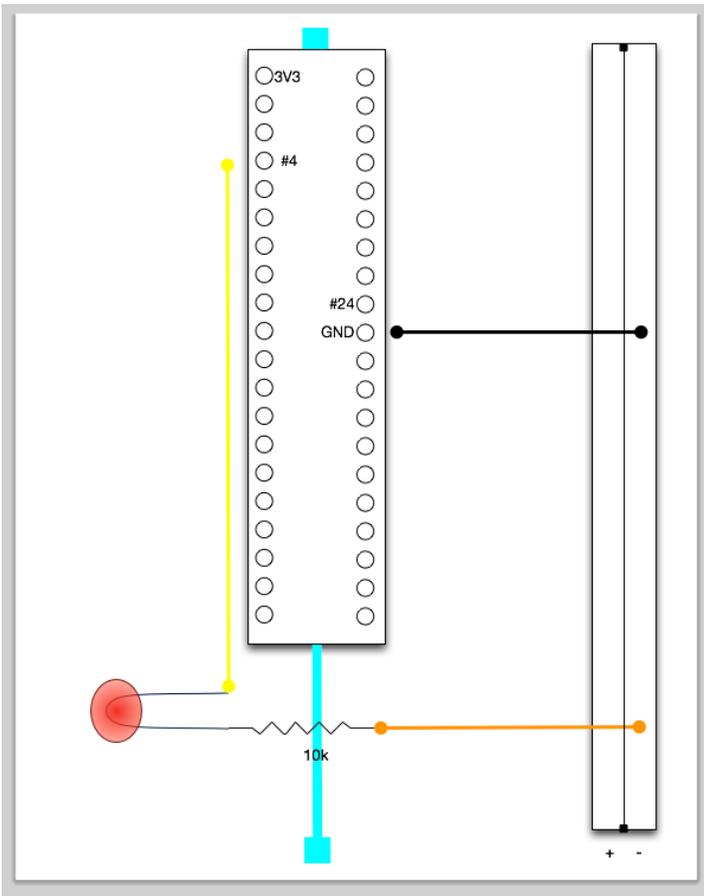
- Examples/Platform-Specific/Linux/RaspberryPi

## Video

Watch the step-by-step video:

## Circuit Diagram

Here is a circuit diagram for reference:



# Button LED Tutorial

In this tutorial you will extend the [Blinking LED Tutorial](#) to add a button. With Xojo code you will turn on the LED when the button is pressed and turn off the LED when the button is not pressed.

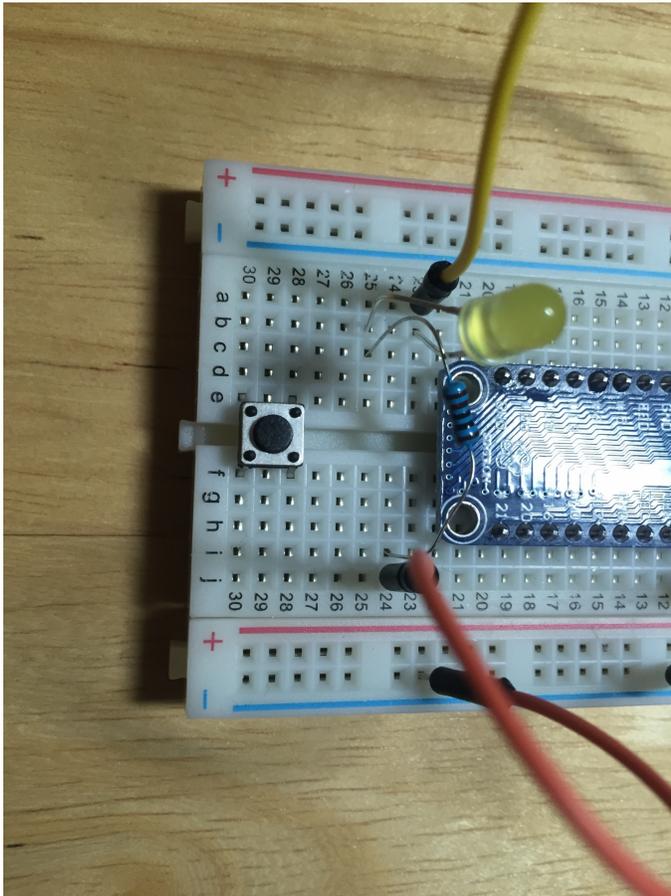
## Parts

In order to build this circuit, you'll need some additional parts:

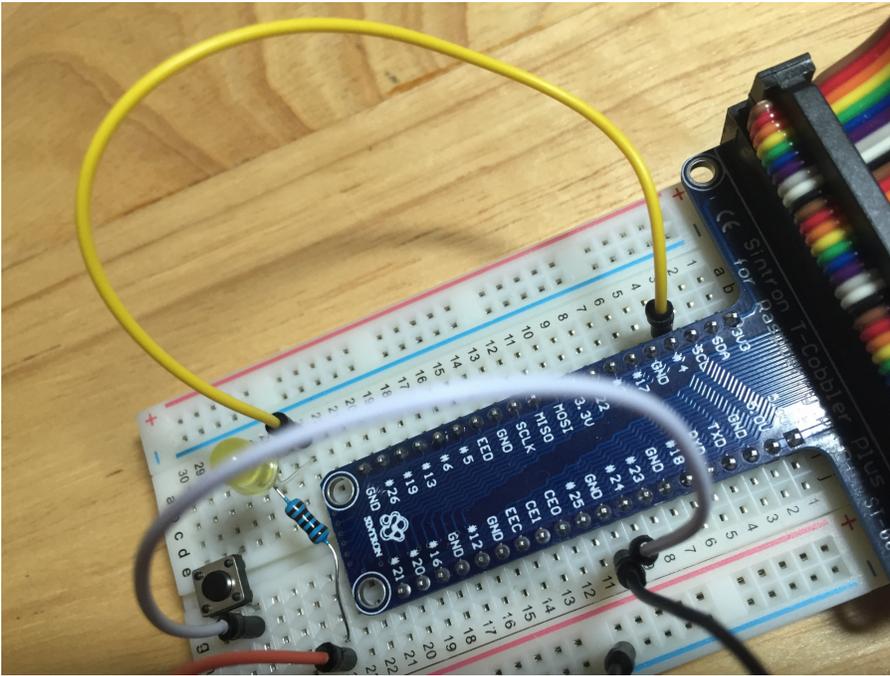
- A digital button
- 2 male-to-male jumper wires
- 1 10k resistor

## Wire the Button Circuit

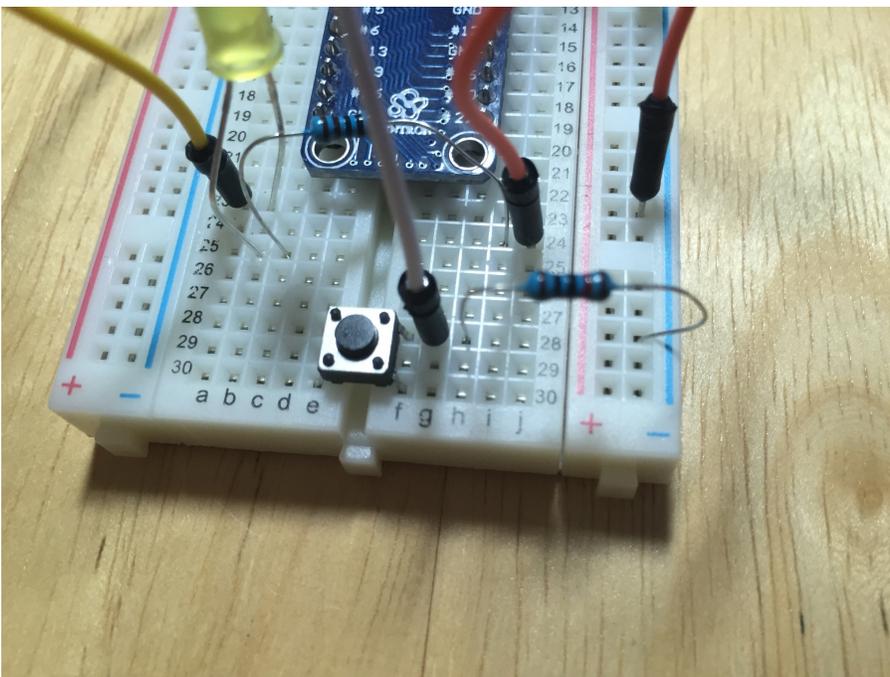
1. Plug the button into the breadboard so that it straddles the center channel.



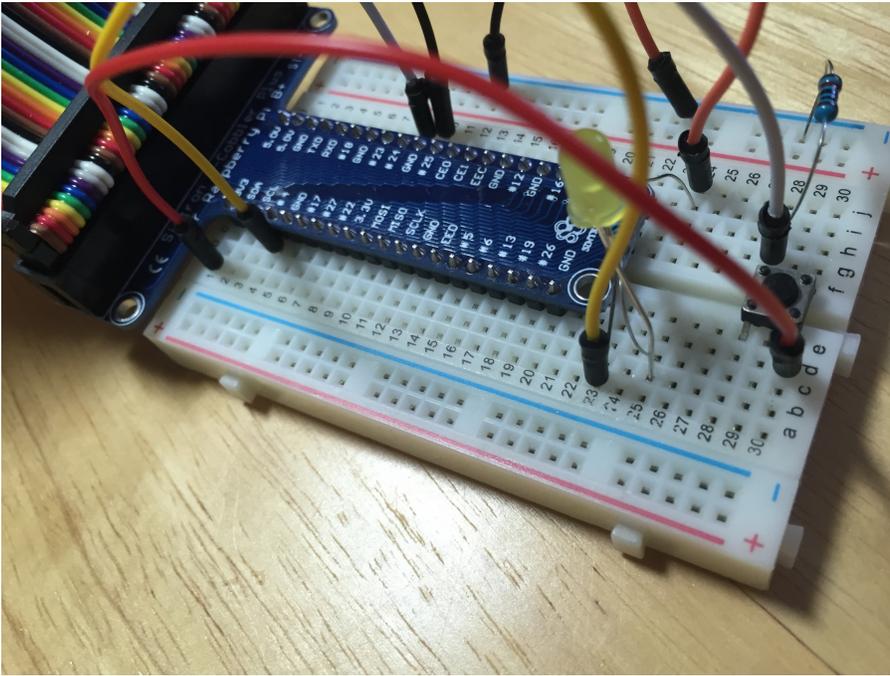
2. Connect a wire from the top of the button (either side) to pin #24.



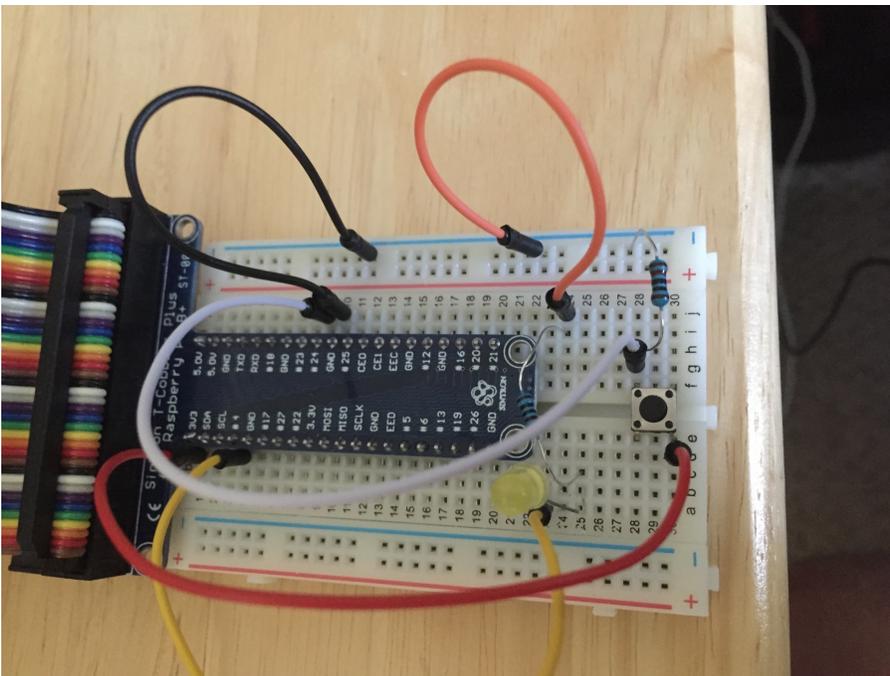
3. Connect a 10k resistor from the top of the button to the same negative (-) column on the breadboard that you used in the Blinking LED Tutorial.



4. Connect a wire from the 3v3 pin to a bottom pin on the button.



5. This completes the circuit.



## Create the Xojo App

To test your circuit, you'll use a simple Xojo app that will check the state of the button. If it is pressed (ON), then the LED will illuminate. When the button is not pressed (OFF), the the LED will be off.

1. Create a new Console project and call it LEDButton.
2. Add the GPIO module to the project.
3. Add this code to the App Run event handler:

```

GPIO.SetupGPIO

Const kLEDPin = 4 // "#4" on the pinout
Const kButtonPin = 24 // "#24" on the pinout

// Set the LED pin to accept output
GPIO.PinMode(kLEDPin, GPIO.OUTPUT)

// Set the button pin to accept input
GPIO.PinMode(kButtonPin, GPIO.INPUT)

// Blink LED when button is pressed
While True
  // If the Button Pin is ON then turn on the LED
  If GPIO.DigitalRead(kButtonPin) = GPIO.ON Then
    // Turn the pin on (give it power)
    GPIO.DigitalWrite(kLEDPin, GPIO.ON)
  Else
    GPIO.DigitalWrite(kLEDPin, GPIO.OFF)
  End If

  App.DoEvents(50) ' Small delay helps with CPU usage
Wend

```

## Transfer and Run the Xojo App

Start your SFTP app and connect to the Raspberry Pi. Transfer the LEDButton folder from the Linux build folder to the Pi.

On the Pi, open the Terminal (or ssh to it from your development machine). In the terminal, navigate to the location of the LEDButton folder and then CD to the folder:

```
cd LEDButton
```

Run the app:

```
sudo ./LEDButton
```

 Normally sudo is needed to access GPIO. The GPIO.SetupGPIOs method does provide access to some GPIO functionality without requiring sudo.

The LED should light up when you press the button. To stop the app, press Control-C.

Starting with Raspbian Jessie, you no longer have to use sudo to access GPIO. In order to not require sudo, you have to first set an environment variable before you run the app:

```
export WIRINGPI_GPIOMEM=1
```

The Xojo project is included with the Xojo examples and is located here:

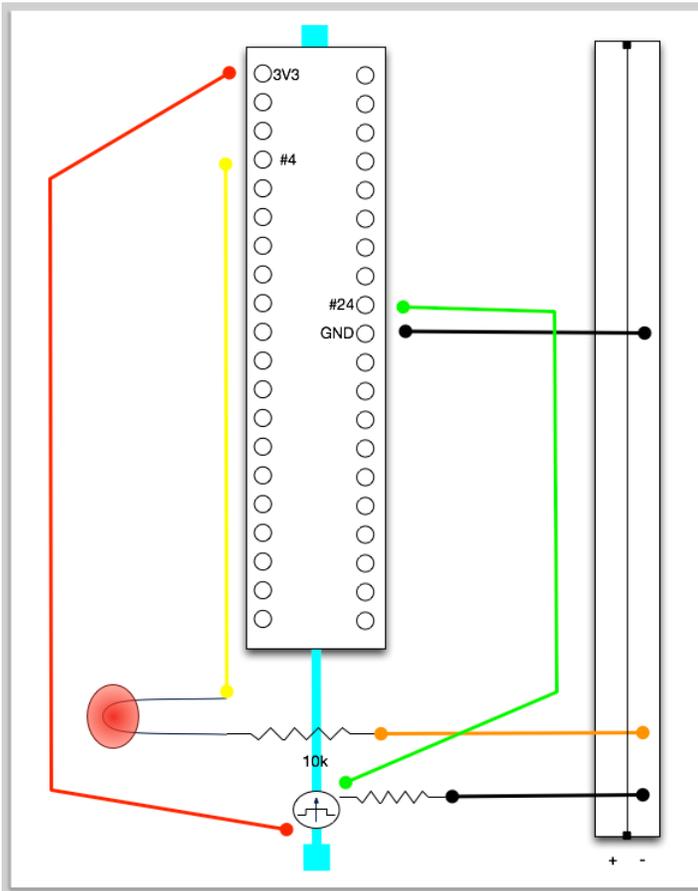
- Examples/Platform-Specific/Linux/RaspberryPi

## Video

Watch the step-by-step video:

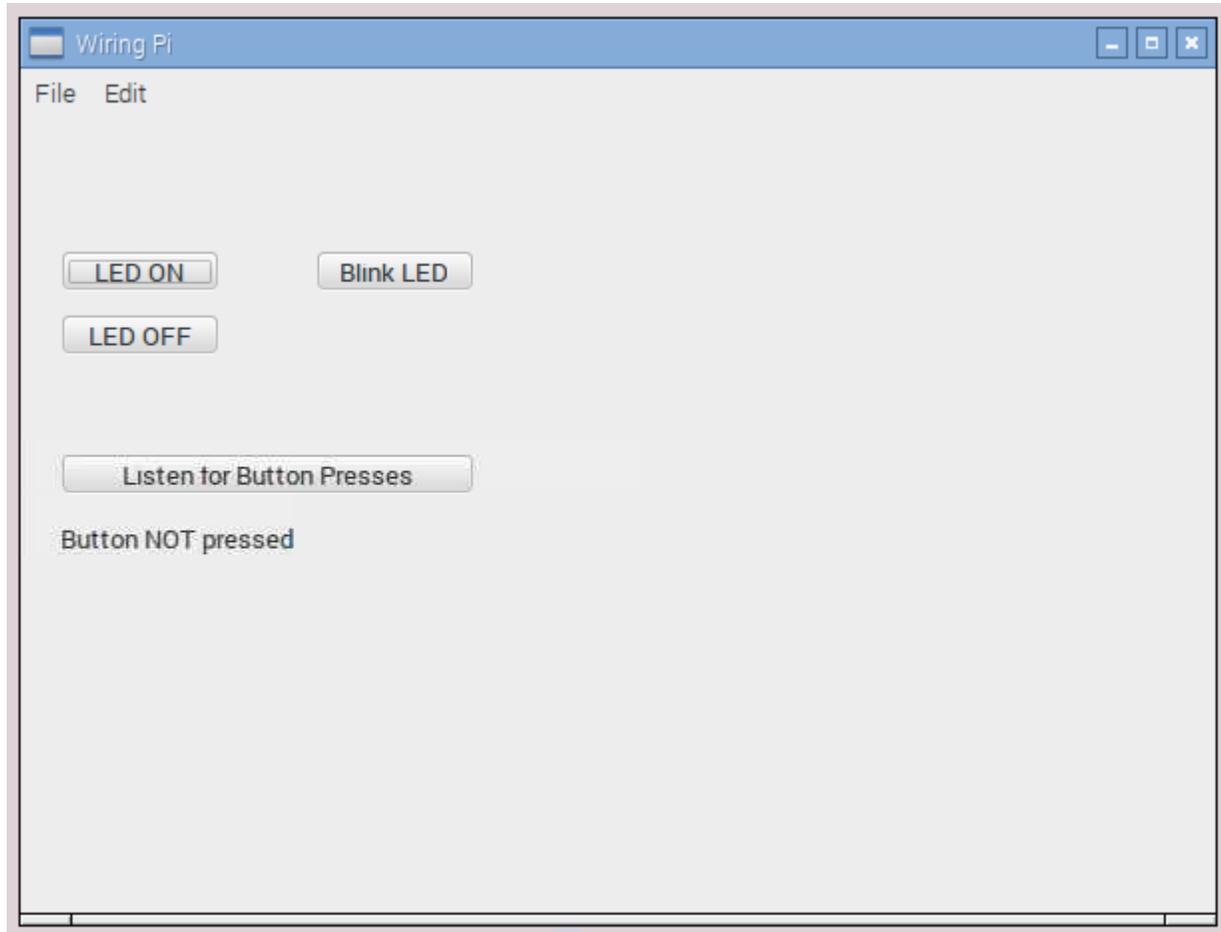
## Circuit Diagram

Here is a circuit diagram for reference:



# Desktop GPIO App

You can also create desktop apps that can control GPIO. This example project controls the LED and button circuits you created in the [Blinking LED Tutorial](#) and the [Button LED Tutorial](#).



The Xojo project is included with the Xojo examples and is located here:

- [Examples/Platform-Specific/Linux/RaspberryPi](#)

Use `gksudo` to start a desktop app from the command line with `sudo`:

```
gksudo ./WiringPi-UI
```

Or set the environment variable before starting the app from the command line:

```
export WIRINGPI_GPIOMEM=1
```

```
./WiringPi-UI
```

# Creating Crash Dumps on Windows

If you experience application crashes you may be asked by support to create a crash dump file. Crash dumps are created automatically by Windows if the following registry key is present:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\Windows Error Reporting\LocalDumps
```

In order to help support troubleshoot your problem please do the following:

- Create the registry key LocalDumps if it is not present already.
- Reproduce the problem (i.e. make the application crash).
- Locate the crash dump file in %LOCALAPPDATA%\CrashDumps. Note that if the crashing application runs under the System account, that resolves to C:\Windows\System32\config\systemprofile\AppData\Local\CrashDumps.
- Attach the crash dump file to a Feedback case.

This works on all versions of Windows beginning with Vista and Server 2008 (including Windows 7, Windows 8, Server 2012, etc.).

---

Source: [Helge-Klein](#)

# Internal Plugins

The items listed in this section are currently implemented as internal plugins in Xojo. **This is an implementation detail that is subject to change at any time.**

This information is needed by anyone who is building installer applications for the Windows build system introduced in version 2008 Release 2. In general, this build system produces a folder rather than a single .exe file, as was the case with previous versions. The folder includes the .exe file, plus a "Lib" folder that contains dlls for all the plugins that the application uses. Since a number of items are implemented as internal plugins, you may well see a Lib folder even when you have not installed any external plugins.

If the application uses no internal or external plugins, the build process produces a single .exe file.

If the application uses one or more of the plugins in a group, then you will get the .dll for the group.

**NOTE:** some high-level classes internally use functions implemented as internal plugins. For example, `HTTPSocket` and `POP3Socket` both use the MD5 function for authentication purpose thus causing the MD5 plugin to be included in the Lib folder.

- AddressBook
  - AddressBook
- Appearance Pak
  - BevelButton
  - PopupArrow
  - DisclosureTriangle
  - ProgressWheel
  - Imagewell
  - Separator
  - Placard
  - UpDownArrows
- Browser Plug-in
  - HTMLViewer
- Game Input
  - GameInputDevice
  - GameInputManager
  - GameInputElement
- Internet Encodings
  - EncodeBase64
  - DecodeURLComponent
  - DecodeBase64
  - EncodeQuotedPrintable
  - DecodeQuotedPrintable
  - EncodeURLComponent
- MD5
  - MD5
  - MD5Digest
- QuickTime
  - EditableMovie
  - QTGraphicsExporter

- MediaPlayer
- QSoundTrack
- QT3DAudio
- QTTrack
- QTEffect
- QTUserData
- QTEffectSequence
- QTVideoTrack
- RBScript
  - RBScript
- RegEx
  - RegEx
  - RegExOptions
  - RegExException
  - RegExSearchPatternException
  - RegExMatch
- Shell
  - Shell
  - ShellNotRunningException
  - ShellNotAvailableException
- Sockets
  - SSLSocket
- XML
  - XMLAttributeList
  - XMLNodeList
  - XMLContentModel
  - XMLReader
  - XMLDocument
  - XMLReaderException
  - XMLDOMException
  - XMLXsltHandler
  - XMLElement
  - XMLNamespaces
  - Anything else that starts with "XML".

# Keyboard Shortcuts

## General

Hold down Alt (Windows), Shift (on Linux) or Option (on OS X) to prevent loading the user interface state for the IDE. This prevents the loading of tabs and defaults Xojo to a standard window size.

## Code Editor

Add Shift to many of the these commands to also select the text.

In general, the Code Editor on OS X uses [standard Cocoa text editor keyboard shortcuts](#).

OS X	Windows/Linux	Description
Cmd + [Double-click on method]	Ctrl + [Double-click on method]	Navigate to the method that was double-clicked.
Cmd + '	Ctrl + '	Comment or uncomment the selected block of code. If not code is selected, then it works on the current line.
	Ctrl + m	Adds the line extension character "_" and moves you to a new line.
Cmd + \	Ctrl + \	Toggles the breakpoint on/off for the current line of code.
Shift + Return	Shift + Return	This will auto-complete a code block. For example, if you type "If True" and then press Shift-Return, the code editor will auto-complete to look like the following: <pre>If True Then   </pre> End If with the cursor on the line between the Then and the End If. This works with all code blocks, such as <a href="#">If...Then</a> , <a href="#">While...Wend</a> , <a href="#">For...Next</a> , <a href="#">Select Case</a> , etc.
Cmd + Left Arrow	Home	Moves the cursor to the beginning of the line.
Cmd + Right Arrow	End	Moves the cursor to the end of the line.
Ctrl + t		Transposes the characters at the cursor.
Ctrl + k		Deletes characters from the cursor to the end of the line.
Option + Backspace	Ctrl + Backspace	Deletes the previous word.
Cmd + Backspace		Deletes characters from the cursor to the beginning of the line.
Option + Right Arrow	Ctrl + Right Arrow	Moves forward one word at a time.
Option + Left Arrow	Ctrl + Left Arrow	Moves backward one word at a time.

OS X	Windows/Linux	Description
Option + Delete	Ctrl + Delete	Deletes the next word.
Cmd + Up Arrow or Home	Ctrl + Home	Go to first line of code.
Cmd + Down Arrow or End	Ctrl + End	Go to last line of code.

## Menus

### File Menu

OS X	Windows / Linux	Command
⌘-,		Open Preferences / Options
⌘-Q		Quit/Exit Xojo
⌘-N	Ctrl+N	New Project
⇧-⌘-N	Ctrl+Shift+N	New Workspace
⌘-T	Ctrl+T	New Tab
⌘-O	Ctrl+O	Open...
⌘-W	Ctrl+W	Close Tab
⇧-⌘-W	Ctrl+Shift+W	Close Window
⌘-S	Ctrl+S	Save
⇧-⌘-S	Ctrl+Shift+S	Save As...
⌘-P	Ctrl+P	Print...
⇧-⌘-P		Page Setup...

### Edit Menu

OS X	Windows / Linux	Command
⌘-Z	Ctrl+Z	Undo
⇧-⌘-Z	Ctrl+Y	Redo
⌘-X	Ctrl+X	Cut
⌘-C	Ctrl+C	Copy
⌘-V	Ctrl+V	Paste
⌫	⌫	Delete
⌘-D	Ctrl-D	Duplicate

OS X	Windows / Linux	Command
⌘-A	Ctrl-A	Select All
⇧-⌘-A	Ctrl+Shift+A	Deselect All
⌘-'	Ctrl+'	Comment
↵ (return)	↵ (return)	Set Default Value Of
⌘-F	Ctrl+F	Find

## View Menu

OS X	Windows / Linux	Command
⌘-E	Ctrl+E	Go to Layout / Code Editor
⌘-L	Ctrl+L	Library
⌘-I	Ctrl+I	Inspector
⌘-F		Enter Full Screen

## Insert Menu

OS X	Windows / Linux	Command
⌘-⌘-E	Ctrl+Shift+E	Event Handler
	Ctrl+Shift+H	Menu Handler
⌘-⌘-M	Ctrl+Shift+M	Method
⌘-⌘-N		Note
⌘-⌘-P	Ctrl+Shift+P	Property
⌘-⌘-C	Ctrl+Shift+C	Constant

## Project Menu

OS X	Windows / Linux	Command
⌘-R	Ctrl+R	Run
⌘-.	Ctrl+ESC	Stop Debugging
⌘-⌘-R	Ctrl+Alt+R?	Run Remotely
⇧-⌘-O	Ctrl+Shift+O	Step Over
⇧-⌘-I	Ctrl+Shift+I	Step Into
⇧-⌘-T	Ctrl+Shift+T	Step Out

OS X	Windows / Linux	Command
⌘-\	Ctrl+\	Turn On/Off Breakpoint
⌘-K	Ctrl+K	Analyze Project
⇧-⌘-B	Ctrl+B	Build Application
⌘-⌘-F		Go To Search
⇧-⌘-L	Ctrl+Shift+L	Go To Location

## Window Menu

OS X	Windows / Linux	Command
⌘-M		Minimize
⇧-⌘-]	Ctrl+Tab	Next Tab
⇧-⌘-[	Ctrl+Shift+Tab	Previous Tab
⌘-1 through ⌘-5		Select Window

## Help menu

OS X	Windows / Linux	Command
	F1	Language Reference

# Uniform Type Identifiers

Uniform Type Identifiers (UTIs) are used by the system to identify the type of files or in-memory data. For a file, the UTI is determined by the system with the help of its extension and its creator or its MIME type. As an example, OS X and iOS applications use UTIs to declare the format for data they place on a pasteboard. OS X apps use UTIs to declare the types of files that they are able to open.

## Usage in Xojo

In Xojo, you specify UTIs for files using File Type Sets. They can then be used to select specific types of file or to control icons used with files owned by your apps.

The Clipboard uses UTIs to identify its contents.

The SpotlightQuery class uses UTIs to search for specific files.

## Inheritance

Uniform Type Identifiers support multiple inheritance, i.e. a UTI can be a child of any number of other UTIs. It is said to conform to another UTI.

For example, the UTI for an AVI movie file is `public.avi` and it conforms to:

- `public.movie`: the base type for any movie
- `public.audiovisual-content`
- `public.data`: the base type for physical files or data
- `public.item`: the base type for the physical hierarchy
- `public.content`: the base type for all document content

## UTI System Tips

### Getting All the UTIs Declared on a System

In the Terminal application (in Applications→Utilities→Terminal), type the following command (case-sensitive) to display a list of all UTIs declared by the system and other applications:

```
mdimport -A
```

### Getting UTI and Other Metadata of a Given File

In the Terminal application (in Applications→Utilities→Terminal), type the following command (case-sensitive):

```
mdls (followed by a space)
```

then drag & drop the file you want information on onto the Terminal window and press the return key.

Among other metadata used by Spotlight, you should see the following for an AVI file:

```
kMDItemContentType          = "public.avi"
kMDItemContentTypeTree     = (
    "public.avi",
    "public.movie",
    "public.audiovisual-content",
    "public.data",
    "public.item",
    "public.content"
)
```

- kMDItemContentType contains the file's UTI
- kMDItemContentTypeTree is the list of all the UTIs the file conforms to.

## To Learn More

- [Uniform Type Identifier](#) in the Apple iOS Developer Library
- [Uniform Type Identifier](#) on Wikipedia
- [Uniform Type Identifier](#) in the Apple Mac Developer Library

# Desktop App Deployment

## Table of Contents

- [Windows](#)
  - [Installers](#)
  - [Zip](#)
- [OS X](#)
  - [Disk Images](#)
  - [Installers](#)
  - [Code Signing for GateKeeper](#)
  - [Mac App Store](#)
- [Linux](#)
  - [Generic Installer](#)
  - [Debian Installer](#)
  - [RedHat Installer](#)
  - [Zip](#)

## Windows

On Microsoft Windows, applications are deployed using installers. Any installer tool will work on your apps, including: InnoSetup, Advanced Installer, NSIS and InstallShield.

- [Sample InnoSetup script for Xojo apps](#)

**Note:** These installer tools all have to be run on Microsoft Windows in order to create a Windows installer. However, InnoSetup does work with WINE to allow it to be run on OS X or Linux.

## Installers

When you create your installer, you need to tell it to include all the files necessary to run the application. At a minimum, this includes the EXE file and the contents of its associated Libs and Resources folders.

For example, an application called Sliders would create a file called “Sliders.exe” and a folder called “Sliders Libs”. The Libs folder contains DLLs for libraries, plugins and other associated files needed by your application.

If your application has other support files or folders, such as a Resources folder, then make sure that your installer includes them as well.

### Setup.exe or MSI

Most installer tools allow you to create your installer as a Setup.exe file or as an MSI (Microsoft Installer) file. Either work fine, but MSI files have the advantage of being the current recommended method from Microsoft and can be used by IT departments for better control of installations. Choose what works best for your customers.

### Location and Shortcuts

Windows applications are installed to the Program Files folder. On 64-bit systems, your applications are installed to the Program Files (x86) folder (because they are 32-bit).

Windows users expect to have easy access to your application, so this means you should create easily accessible shortcuts. Your installer tool should provide you with the option of creating a shortcut for the user on the Desktop and in the Start Menu.

## Zip

For very simple distribution, you can Zip the application and its supporting files (such as the Libs folder). You can create a Zip by right-clicking on the parent folder in Windows Explorer and selecting Send To -> Compressed (zipped) Folder.

Once unzipped, the application can be run from any location.

Although this can be useful for testing purposes, it is not recommended for proper Windows application deployment.

### Microsoft Redistributable Files

Your applications require the Microsoft Visual C++ Runtime. The required files are included in the Libs folder (msvcp100.dll and msucr100.dll) of your applications for your convenience.

However, for increased Windows compatibility, you should instead consider including the Microsoft Visual C++ Runtime Redistributable Installer as part of your installer. To do this you do not include the msvcp100.dll and msucr100.dll files that are in the Libs folder and instead embed the Microsoft Visual C++ Runtime Redistributable as part of the installer. Most installer tools have a way to do this automatically.

The advantage of doing it this way is that the Visual C++ Runtime Redistributable will install its files into the Windows system folder, which allows them to be updated by Microsoft Windows Update for security and other reasons. If you leave your Visual C++ Runtime files in the Libs folder then they cannot be updated by Windows Update.

- [Download the Visual C++ Runtime Redistributable](#)

## OS X

Generally, OS X applications consist of a single App file, called the Application Bundle. You can embed other files into the Application Bundle because it is technically a folder that OS X treats as a file. In Finder, you can right-click on any App and select “Show Package Contents” to see the actual contents of the Application Bundle as a folder.

Of course, your application can still have separate files not in the Bundle, in which case you want to make sure that your application is in a folder of its own.

Since applications are technically a folder, you have to include them in some sort of container in order to distribute them, such as disk images (DMG), installers and even simple Zip files.

### Disk Images (DMG)

A disk image is a file that simulates a disk or drive. After downloading the file (and double-clicking it), it appears in the sidebar of the Finder as a drive. This is called “mounting” the disk image. When the user clicks on the drive in the Finder sidebar, they see your application and typically drag it to the Applications folder to install it.

To create a disk image, you can use the Disk Utility application included with OS X or try one of the many specialized disk image creation tools such as DMG Canvas.

A disk image is probably the most common way to install OS X applications, but keep in mind that some users (especially those new to OS X) may find the concept of mounting a drive and dragging a file to the Applications folder very confusing.

Also keep in mind that if the user mistakenly tries to run the Application directly from the disk image, it may not behave as expected because the disk image is read-only.

## Installers

OS X can also use an actual installer to install your application, but it is not common for applications distributed outside the Mac App Store. To create an installer you can use the PackageMaker tool (included with the OS X Development tools) or you can use the free [Packages](#) installer.

An installer gives you more control over permissions and other settings, but they can be much more difficult to create than a simple disk image. Also, remember that an installer on OS X (a pkg file) is actually a bundle so you have to distribute it in a disk image or a Zip.

## Zip

A Zip file is an archive of your application. A zip is easy to download and most users understand what they are. They can usually be unzipped by simply double-clicking on them, which reveals the application itself. The application can then be copied to the Applications folder.

You can create a Zip by right-clicking on your application in the Finder and selecting Compress.

## Code Signing for GateKeeper

OS X 10.8 introduced a feature called GateKeeper. GateKeeper is designed to provide a level of security for users installing applications. Essentially, if the application (or its installer) is not digitally signed using an Apple-provided certificate then OS X will not allow the application to be installed by default.

This can be overridden by the user right-clicking on the application (or installer) and selecting Open, but not many users will know about this.

This means you are probably going to want to digitally sign your OS X applications. To get a certificate, you have to join the Mac Developer Program (\$99). Your certificate is good for five years.

You use Certificate Utility in the Mac Dev Center to create a certificate (you can use the same certificate for all your applications).

Once you have followed the instructions to create the certificate and have installed it on your computer, you can use it to code sign your application using this command:

```
codesign -f -s "Developer ID Application: YourName" "YourXojoApp.app"
```

Code signing must be done as the absolute last step. If you modify anything inside your application bundle (such as Info.plist), you will invalidate the signature and have to code sign again.

 Note: If you are using an installer then you have to sign the installer separately using a special installer certificate.

## Mac App Store

The Mac App Store is a great way to distribute your OS X applications. People find it easy and convenient to purchase applications from the Mac App Store. Unfortunately, getting applications into the Mac App Store is not easy or convenient for the developer.

### Sandboxing

Sandboxing is used to restrict what your application is able to access. This serves as a security feature, because if an app were to become compromised for some reason then it would be unable to do as much damage as if it had full control of the computer.

Sandboxing works best with Cocoa applications, although it does work for Carbon applications (with some Apple restrictions).

### Certificates and Code Signing

To be able to submit an application to the Mac App Store for Apple to review, you need to first create your Mac App Store certificates using the online Certificate Tool that is part of the Mac Dev Center. You also need to create a bundle ID for your application.

When your application is complete, you then need to code sign your application (and all its dynamic libraries). You also need to create the installer and code sign that as well.

Finally you can create a submission using iTunes Connect, fill in all the required information and then upload your application using the Application Loader tool that is part of the OS X Development Tools.

Once you have submitted something to the Mac App Store, it can take several weeks before your application is approved by Apple and ready for sale.

### Validating The Apple ID

The above steps work great for any apps, including free apps, but if you are selling your app, you will want to prevent people from copying the purchased app to another computer and running it there (without having to log into their Apple ID). To do this you also need to verify the Apple ID.

Verifying an AppleID requires calling a Cocoa API. The code is far too involved to include here, but there is a sample project in the "Platform Specific" folder for OS X that you can reference.

## Linux

Linux applications also have a variety of deployment options. The simplest is to use GZip and let the user put the application wherever they want. You can also use a tool such as InstallJammer that creates a generic installer that works with a variety of Linux platforms. Lastly, you can create separate installers for each Linux distribution.

Common installer formats are deb (used by Debian and Ubuntu) and RPM (RedHat Package Maker) used by RedHat.

## Generic Installer

[InstallJammer](#) is an open-source product that has a simple user interface for creating an installer that works on a variety of Linux distributions.

Unfortunately, this tool is no longer being actively developed, but it still works well and is a good choice if you do not use Linux often enough to master creating dedicated installers.

## Debian Installer

Debian installers are used by Debian-based Linux distributions, such as Ubuntu or Linux Mint. They can be installed by the Synaptic Package Maker or from the terminal.

You create Debian installers using the `dpkg-deb` terminal application. Unfortunately, its usage is far more involved than can be discussed here. This tutorial describes how you can create a Debian package:

- [Debian Binary Package Building HOWTO](#)

## Redhat Installer

The Redhat installer format (RPM) is used by Redhat-based Linux distributions. You can create RPM installers using the `rpmbuild` terminal application. Unfortunately, its usage is far more involved than can be discussed here.

The Fedora Project does have a good walkthrough:

- [How to create an RPM package](#)

## Zip

A Zip file is an archive of your app. A zip is easy to download and most users understand what they are. They can usually be unzipped by simply double-clicking on them, which reveals the app itself. The app can then be copied to where the user wants it.

# Web App Deployment

You can create two types of web applications, Stand-alone and CGI. In addition to differences in how they operate, they are also deployed differently.

## Table of Contents

- [Stand-alone Application](#)
- [CGI Application](#)
  - [Apache](#)
  - [IIS](#)
- [Xojo Cloud](#)

## Stand-alone Application

When you create a stand-alone web application, you get a single application that consists of both your application and a standalone web server. This web server listens on the port specified in the Build settings.

To start a stand-alone web application, you simply run the application from the command line or terminal. This command runs a standalone web app on OS X or Linux:

```
./mywebapp
```

You can run multiple stand-alone web applications on the same server, but each web application has to be listening on a unique port. To make this easier, you can specify the port as a command-line parameter:

```
./mywebapp --port=8080
```

If you have the port set at 8080 for example, then you can access the web server using the URL followed by the port like this:

```
http://www.myserver.com:8080
```

## Command Line Parameters

You can use these command-line parameters to change settings when you launch the web application.

Command	Description
--Port	The port the web app listens to for a connection. This can be used to change the port that was used to build the web app.
--MaxSockets	The maximum number of allowed connected sockets. This is not the number of maximum users as the number of connections does not map one-to-one with the number of connected users. Do not set the value too high as it will increase memory and CPU usage. The default is 200. If you do not want any unsecured connections, set this value to 0.

Command	Description
--SecurePort	The port the web app listens to for a secure connection. This can be used to change the port that was used to build the web app.
--MaxSecureSockets	The maximum number of allowed secure connected sockets. This is not the number of maximum users as the number of connections does not map one-to-one with the number of connected users. Do not set the value too high as it will increase memory and CPU usage. The default is 200.
--NetworkInterfaceIndex	The index value (of NetworkInterfaces) to use as the NIC for connections.
--SecureNetworkInterfaceIndex	The index value (of NetworkInterfaces) to use as the NIC for secure connections.

## Using Port 80

In order to navigate to a stand-alone web application without specifying the port, you need to use port 80, the standard port used by web browsers. However, OS X and Linux do not allow you to use port 80 (or any port lower than 1024) unless you have root access. This means that on OS X and Linux you need to use the sudo command to start your web application if you are using port 80:

```
sudo ./mywebapp
```

**Note:** The sudo command prompts you for your user name and password.

## Background Processing

When you run your web application directly from the command line, it will continue running as long as you don't quit the command/terminal window or log out of the account. This technique does not work well on true servers or remote servers. In these cases you want to run the web application as a background process. On Windows this means, you would run the web app as a Windows Service. On Linux it means you would run it as a Daemon. On OS X, you can run it as a daemon or use the preferred launchd command to start it as a daemon.

## Deploying with SSL

In order to deploy a standalone web application so that it uses SSL, you first need an SSL certificate. You can use a self-signed certificate for local testing, but always purchase a real certificate for publicly available web sites. The certificate file used by Xojo is a text file containing the components of your SSL certificate. Generally, a self-signed certificate will have two components (the certificate and key file) while a purchased certificate will likely have three or more (certificate, key, CABundle or intermediates). Create the text file with the same name as your application (minus the app extension if any) and add the ".cert" extension. Then paste the contents of your certificate files in this order:

1. Certificate
2. CABundle
3. Private Key

Start each file on a new line.

This Xojo certificate file must be placed next to the built web app when it runs. This can be done using a Copy Files Build Step with Build Automation.

Now you can start your web app and tell it to listen for connections on a secure port. This uses the `secureport` and `maxsecuresockets` command-line parameters described in the preceding section. The secure port must be different than the (non-secure) port selected in the Shared Properties within Xojo. It also must be different than a port specified with the `port` command-line parameter.

For example, if you want to launch your web app on secure port 8081, use this command:

```
MyWebApp --secureport=8081
```

Once you have launched the app, you can connect to it in the browser like this (for a web app running locally):

```
https://127.0.0.1:8081
```

If you also want to prevent unsecured access to the web app, you can set the number of unsecured sockets to 0:

```
MyWebApp --secureport=8081 --maxsockets=0
```

## Windows Service

To run a web application as a Windows Service, you use the `sc` command from the Command Line to install the app as a service:

```
sc create MyWebAppSvc type= own start= auto binpath= c:\Path\To\Exe\mywebapp.exe
```

Now you can go to the Services Manger in the Control Panel and see the “MyWebAppSvc” service listed. Use Services Manager to Start, Stop or Pause the service.

## Daemon

To make your stand-alone web application run as a daemon background process, you call the `Daemonize` method, usually in the `App.Open` event:

```
#If Not TargetDebug Then
  If Not Daemonize Then
    Print("Unable to Daemonize app.")
    Quit
  End If
#Endif
```

When you now run the stand-alone web application from the terminal, it will immediately return you to the terminal prompt because the app is now running in the background.

This also works on OS X, but Apple prefers that you use the **launchd** command to start daemons. Using launchd means you have to create a Property List file with the specific settings.

For more information, refer to Apple's documentation on this:

- [Creating Launchd Jobs](#)

## CGI Application

When you build your web application as a CGI application, you get an executable file that communicates to your existing web server using CGI (common gateway interface). The way this works is that a small Perl CGI script is created when you build your web application. Its only purpose is to communicate between your web application and the web server. The Perl script starts your web application if it is not running and routes all communication between the web server to your web application and vice versa.

When you build your web application for CGI, you have the option of choosing a port or having it be chosen automatically. This is an internal port that is used for communication between the Perl script and your web application. Because this is internal communication, the firewall is not affected. The only requirement is that nothing else on the server is also using the port.

Nearly all web servers support CGI and Perl, but some are much easier to configure than others.

### Apache

The most common type of web server you will see is Apache, which is typically preinstalled on Linux servers. Most Apache servers have a specific location where CGI applications should be uploaded. This is often called the "cgi-bin" folder.

On a properly configured web server, deploying a web application is as simple as using your FTP client of choice to copy the entire web application folder's contents to the cgi-bin folder (be sure to enable Binary mode for the transfer). The files include:

- The web application itself
- The Resources folder
- The Libs folder
- config.cfg
- the Perl CGI file
- The .htaccess file (which may be hidden in the default view of your FTP software)

Then you can start the web application by accessing the cgi script:

```
http://www.myserver.com/mywebapp.cgi
```

### Troubleshooting

Unfortunately, not all servers are going to be properly configured to run Xojo web applications. You may find that you have to change the configuration yourself. These are some common areas to check. Unfortunately, step-by-step instructions are not possible because every installation of Apache is different, having configuration files in

different places with different web server users and different permissions. Use these tips as guidelines, but you'll still need an understanding of Apache and how it has been installed on the OS you are using.

- **Platform:** Verify that you uploaded the correct platform for your web application. Although you may be developing and testing on OS X, if you are using a Linux server you need to remember to make sure that you create a Linux build to upload.
- **32-bit Libraries:** Your application is a 32-bit application. To run it on a 64-bit version of Linux, you need to ensure the 32-bit compatibility libraries are installed. The command to do this varies depending on the Linux distribution. With Debian, you can use apt-get:
  - `# apt-get update`
  - `# apt-get install ia32-libs-multiarch`
- **Application Identifier:** Verify that the Application Identifier is unique. If there is another web app running on the server with the same Application Identifier, then your new app will not start.
- **Permissions:** Verify that your CGI application and all the libraries in the Libs folder have been copied to the server and have execute permissions (755 is best). You may also need to check the parent folder as well the config.cfg and Perl CGI script. Permissions of 777 are not secure and some web servers (that use seEXEC) will not run anything with that permission.
- **AddHandler:** If your web application displays the page source rather than running, you may need to add the command "AddHandler cgi-script .cgi" to your Apache configuration file or to the .htaccess file generated with your web application.
- **Internal Server Error:** If the Perl CGI script is having trouble starting you may see this error. Verify that Perl is installed correctly and permissions are correct.
- **Unable to Launch Application:** The Perl CGI script is unable to start your application. Usually you will see additional information such as "permission denied". Verify that your web application was uploaded using FTP Binary mode (and not ASCII, which can corrupt the application files).
- **Unable to Connect to Port:** The port may be blocked for some reason. Perhaps the firewall is blocking a local port or the port is in use by another application or service. If you have selected a port, make sure it is > 1024 and < 65536.
- **Web Server Logs:** Your web server logs may have additional useful information. Unfortunately, the location of these logs varies depending on the Linux distribution and version.

## IIS (Microsoft Internet Information Services)

Included with Windows is a web server called IIS. This web server is ideally suited for running simple HTML web sites, .NET web applications and other specific Microsoft web products.

By default, IIS does not have a Perl installation and does not work well with CGI. Although Xojo web applications do work with IIS, properly configuring it is an advanced task. You can refer to these instructions for assistance:

- [IIS Deployment](#)

Alternatively, you can instead configure Apache to run on Windows. These are the official instructions for installing Apache on Windows:

<http://httpd.apache.org/docs/2.2/platform/windows.html>

## Deploying with SSL

Because a CGI app uses a separate web server, you have to ensure your web server (typically Apache) is configured to work with SSL and your SSL certificate. You do not have to do anything specific with your Xojo CGI web app in order to deploy it as SSL.

## Xojo Cloud

If all this setup and configuration is too much to bother with, you should consider hosting your web application using Xojo Cloud. This service allows you to deploy a web application directly from Xojo in one step. With Xojo Cloud, you just check the “Xojo Cloud” target in Build Settings for your web project and then select it to specify the name of the application and the Xojo Cloud server on which to deploy. Then you simply click the Deploy button in the toolbar and your app is built and uploaded to your Xojo Cloud server.

### Control Panel

You can purchase a Xojo Cloud server from the Xojo Store. Once it is provisioned, you can access the Control Panel from your Accounts page on the Xojo web site.

The Control Panel displays your servers (including the IP address) and can perform these actions:

- Restart: Restarts the Xojo Cloud server.
- Rename: Give the server a new name.
- List Apps: Displays all the Xojo apps that have been deployed to the Xojo cloud server. From here you can Quit or Delete apps.
- Setup SSL: Walks you through the process of getting a CSR (Certificate Signing Request) to use to purchase an SSL Certificate that you can then use on your server.
- Delete: Deletes the Xojo Cloud server.
- PostgreSQL: Enables or disables PostgreSQL database server, providing you with a username and password.
- MySQL: Enables or disables MySQL database server, providing you with a username and password.
- Tunnel: Enables or disables an SSH tunnel (providing you with a username and password) to allow you to connect to PostgreSQL/MySQL from your own computer.

### Deployment

It is ridiculously easy to deploy a web app to Xojo Cloud: simply click the Deploy button on the toolbar in Xojo. This builds your app, uploads it to Xojo Cloud and launches it in your default web browser.

The built app's name has a suffix if the Stage Code (in the Shared Build Settings) is set to Development (-Dev), Alpha (-Alpha) or Beta (-Beta). This allows you to deploy apps for testing without affecting an app that is running in production. Set the Stage Code to “Final” to remove any suffix.

### PostgreSQL and MySQL Usage

Once you enable PostgreSQL or MySQL database on your Xojo Cloud server, connecting to the database in a deployed app works exactly as if the DB was on your local computer. There are no special steps to do.

If you want to access a DB on Xojo Cloud from your computer, you will have to enable the **Tunnel** and then connect to the tunnel from the computer. On OS X and Linux, you can use this command to access the tunnel:

```
ssh -L 5432:localhost:5432 dbadmin@ipaddress -N
```

If you are using Windows, you will need to use an external app such as PuTTY. There are a number of tutorials available on the internet, just put “putty ssh tunnel” into your favorite search engine.

**Firewall**

When using classes that have to communicate outside the server (database or socket classes, for example), you need to use the `XojoCloud.FirewallPort` class to first open the port.

```
Dim fwp As New XojoCloud.FirewallPort(587, XojoCloud.FirewallPort.Direction.Outgoing)
fwp.Open() ' This call is synchronous
If fwp.isOpen() Then
    ' Do what you need to do
    fwp.Close() ' Close port when done
End If
```

# IIS CGI Deployment

## Overview

Internet Information Services (IIS) is the name of the web server included with most versions of Microsoft Windows. This guide walks you through configuring IIS so that it can run Xojo web applications.

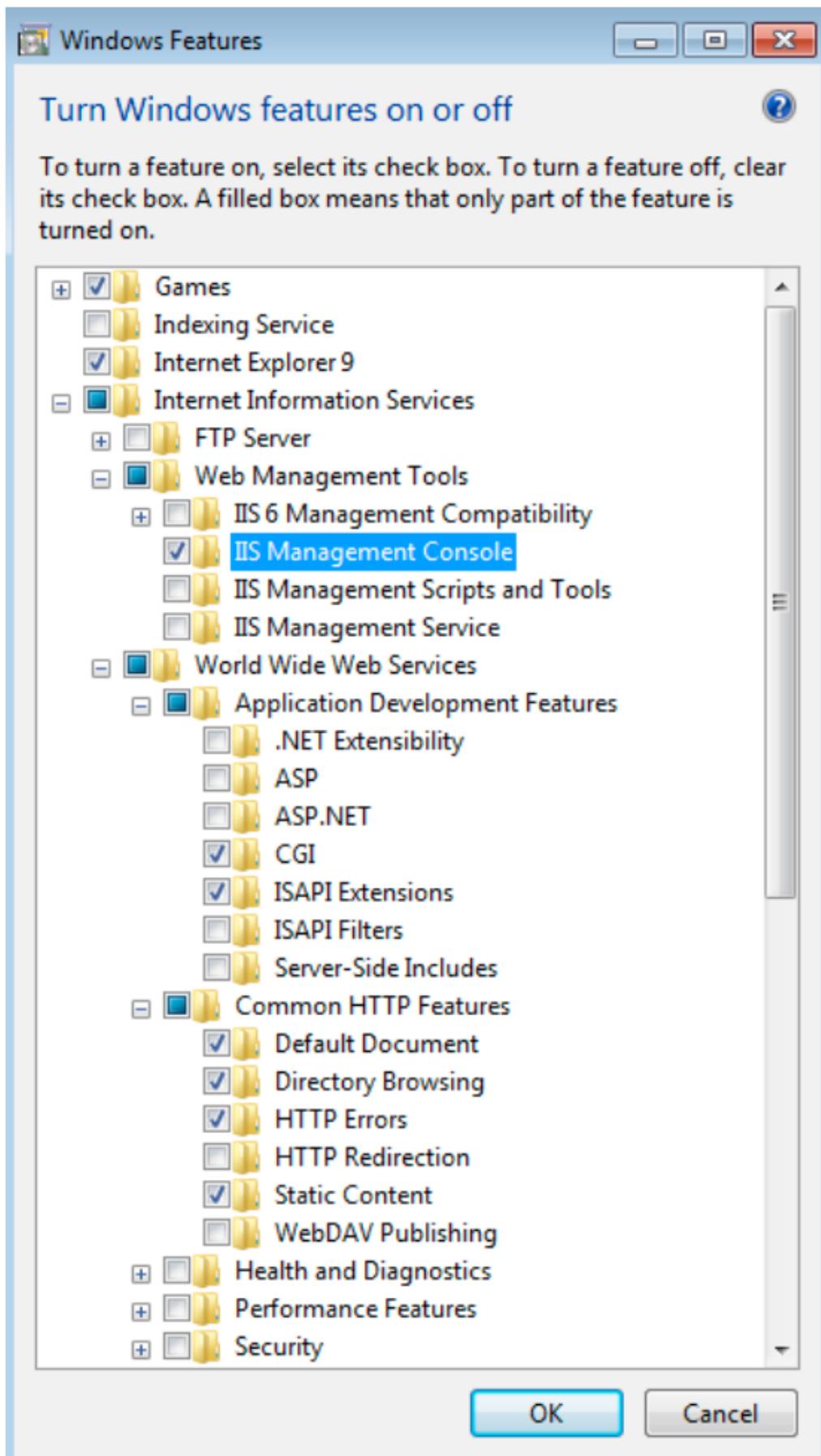
**Note:** We cannot provide support for configuring IIS (or any other web server) for use with Xojo web applications. If you require easy, one-click deployment of your Xojo web apps consider using Xojo Cloud.

## Installing IIS 7

If you are using Microsoft Windows Server, then IIS may already be installed. If so, you can skip to the next section. These steps are for installing IIS 7 on Windows 7.

The following steps are taken from Windows 7:

1. Open the Control Panel.
2. Click the Programs section.
3. In the Programs and Features section select “Turn Windows Features On or Off”.
4. Find the “Internet Information Services” section and check the items as shown below.



5. Click OK to install the selected components.

## Install ActiveState Perl

Xojo web applications use a Perl CGI script to communicate with the web server. ActivePerl is the de facto standard for a Perl installation for IIS. One downside of ActivePerl is that the 64-bit installer does not include the necessary ISAPI Extension, so you'll need to download and install the 32-bit version regardless of the version of Windows you are using. ActivePerl is available here: <http://www.activestate.com/activeperl>.

Go to the downloads area and download the most recent version of ActivePerl for Windows (x86).

1. Run the installer, accepting the defaults.
2. Run the following from the command line (as an administrator):
  1. `ap-iis-config add all`

## Adding a New Site

### Launch IIS Manager

To add a new web site to IIS, you need to use IIS Manager.

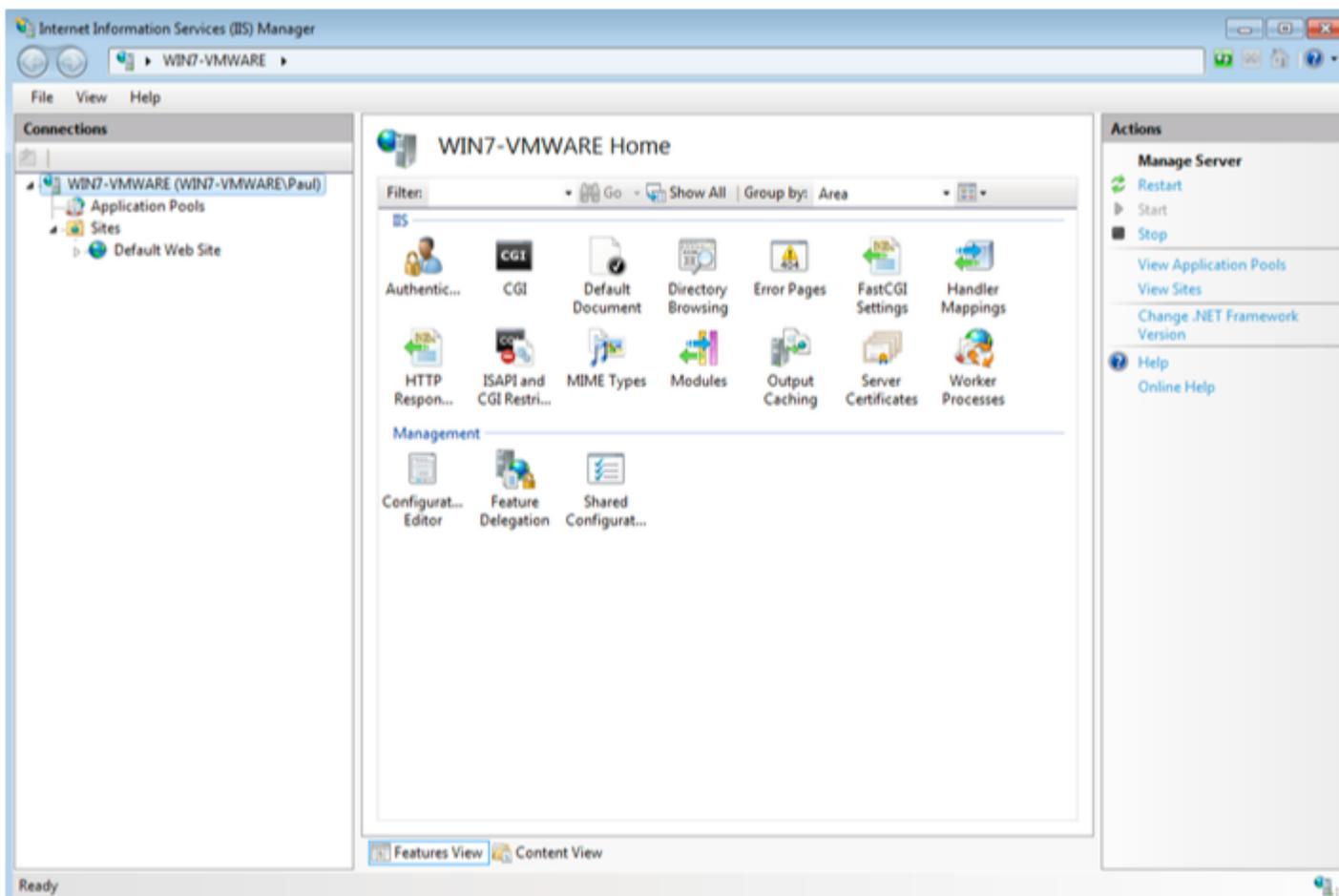
1. Open Control Panel
2. Click System and Security
3. Click Administration Tools
4. In the list, select "Internet Information Services (IIS) Manager"

### Add a Web Site

With IIS Manager running, you can now add a new web site to IIS:

1. Right Click "Sites" in the TreeView on the left hand side (you may need to expand it) of the screen and select "Add Web Site"
2. In the "Site name" field, enter the name of your site. This is mostly for reference purposes, so you can use something like RSWE. Note that the "Application Pool" value also changes. This is a special user just for running your app. Remember the value in this field (especially if you decide to change it).
3. Set the "Physical Path" property to the directory where your application lives. You typically make it a subdirectory of the IIS hosting directory at: `C:\inetpub\wwwroot\`. For this example use "`C:\inetpub\wwwroot\rswe`".
4. Click the "Test Settings..." button. You should see two items, which you do not need to change:
  1. Authentication: Pass-through authentication
  2. Authorization: Cannot verify access to path - This error will prevent your app from running. You'll fix this in the Permissions section.

5. In the Binding section, change the port number to an unused port such as 8080. If you want to use port 80, you'll need to remove "Default Web Site".
6. Assuming you have DNS set up for this machine, you could also enter a host name so users can type a name instead of an IP address.
7. Click OK to save the site.



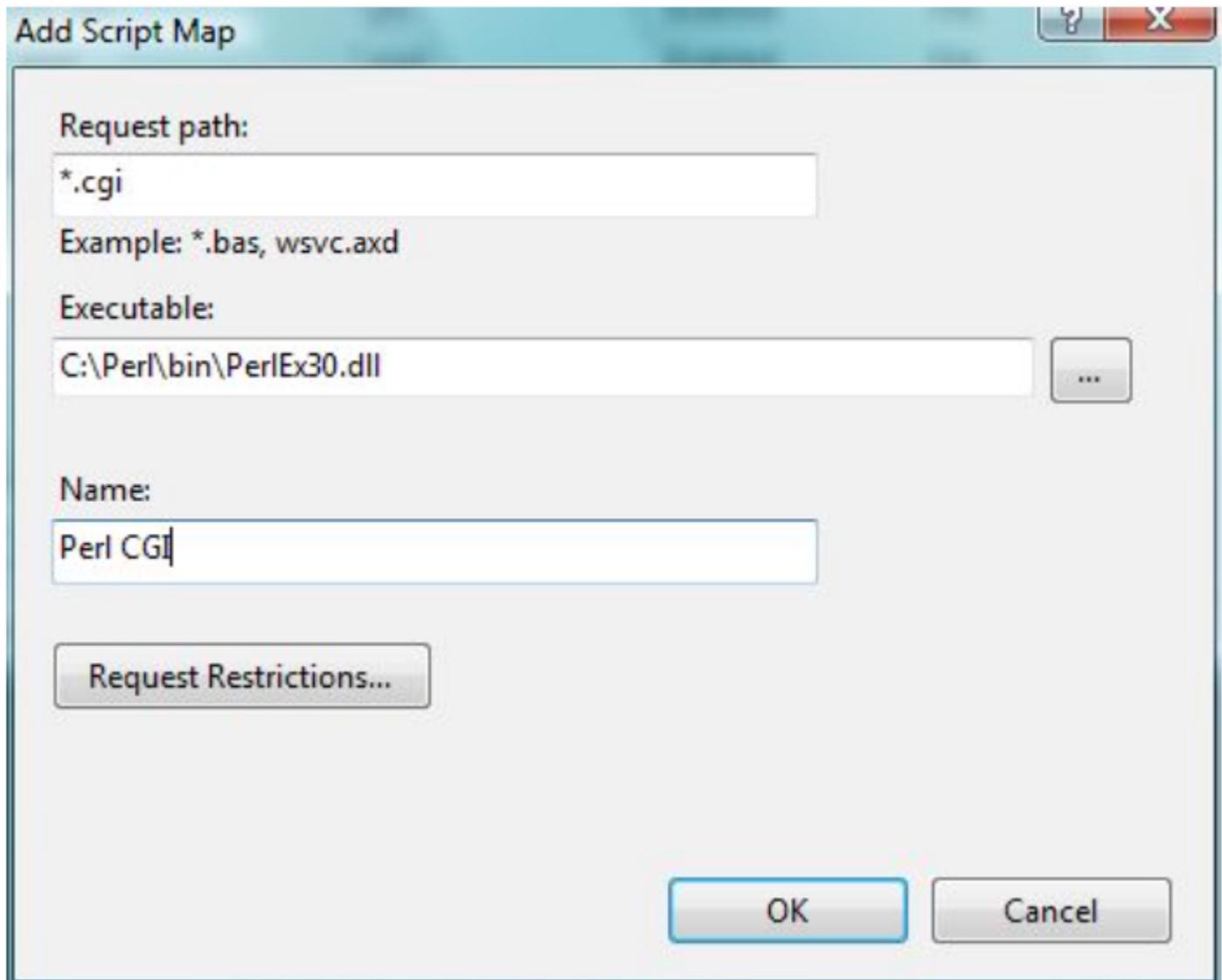
## IIS Configuration

In the left pane of IIS Manager:

1. Select your site
2. Double-click Handler Mappings
3. On the right sidebar, click Add Script Map

You need to configure the Add Script Map settings as shown here:

- Request Path: \*.cgi
- Executable: Browse to the Perl binary folder (by default, C:\Perl\bin) and select PerlEx30.dll
- Name Perl CGI

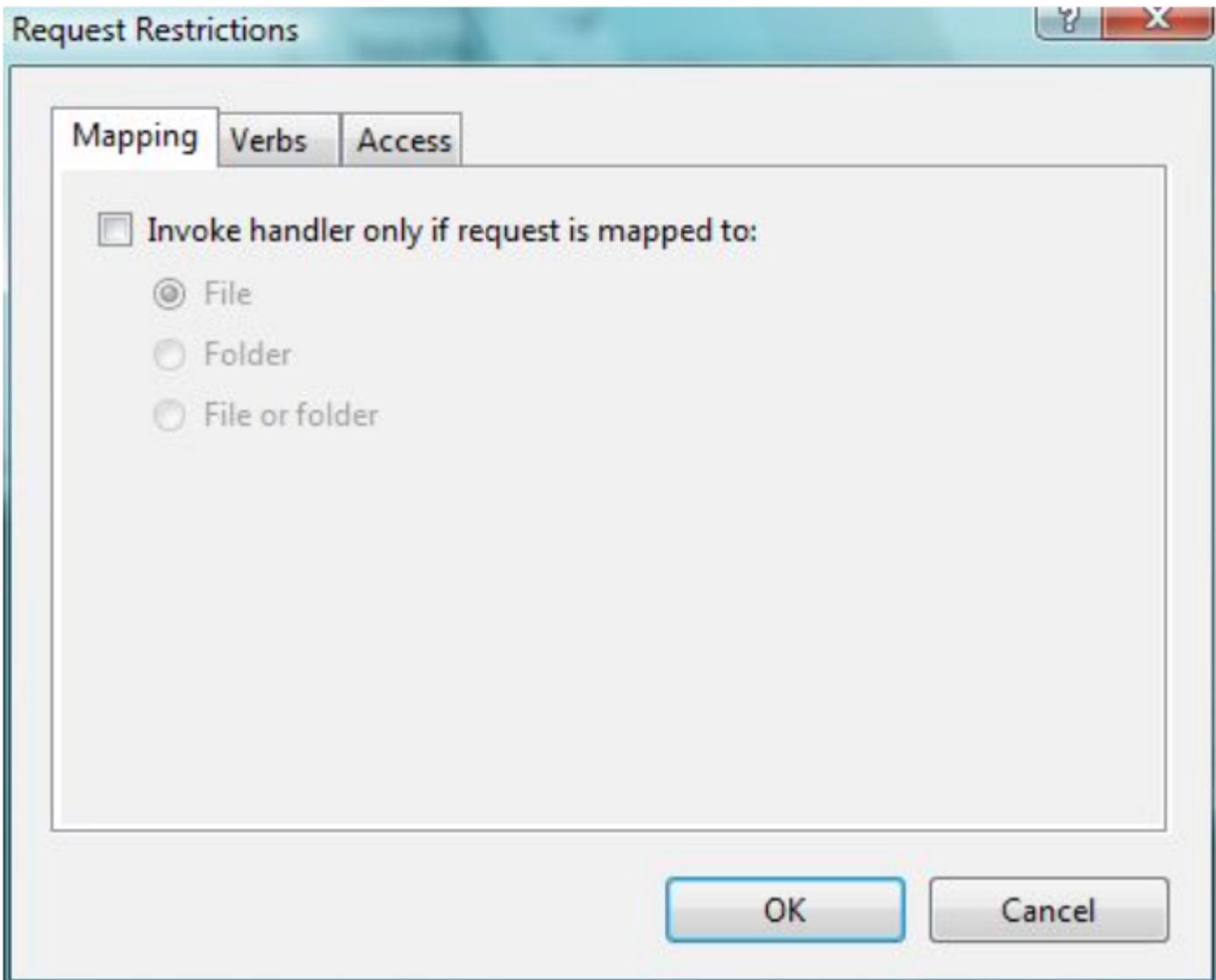


## Request Restrictions

On the window, click Request Restrictions and in its dialog box, make the following changes.

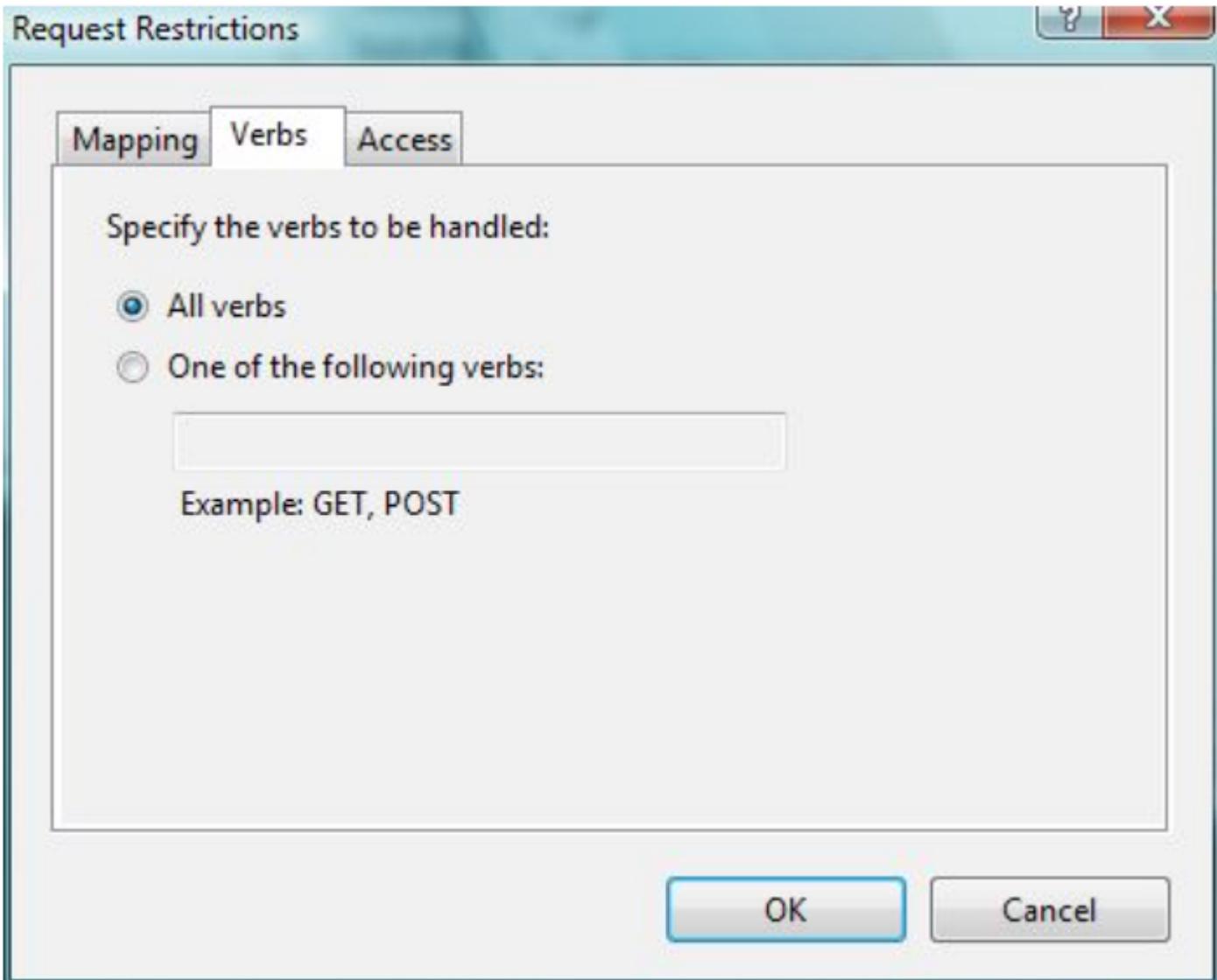
On the Mapping tab:

- Mapping: Check "Invoke handler only if request is mapped to" and select "File".



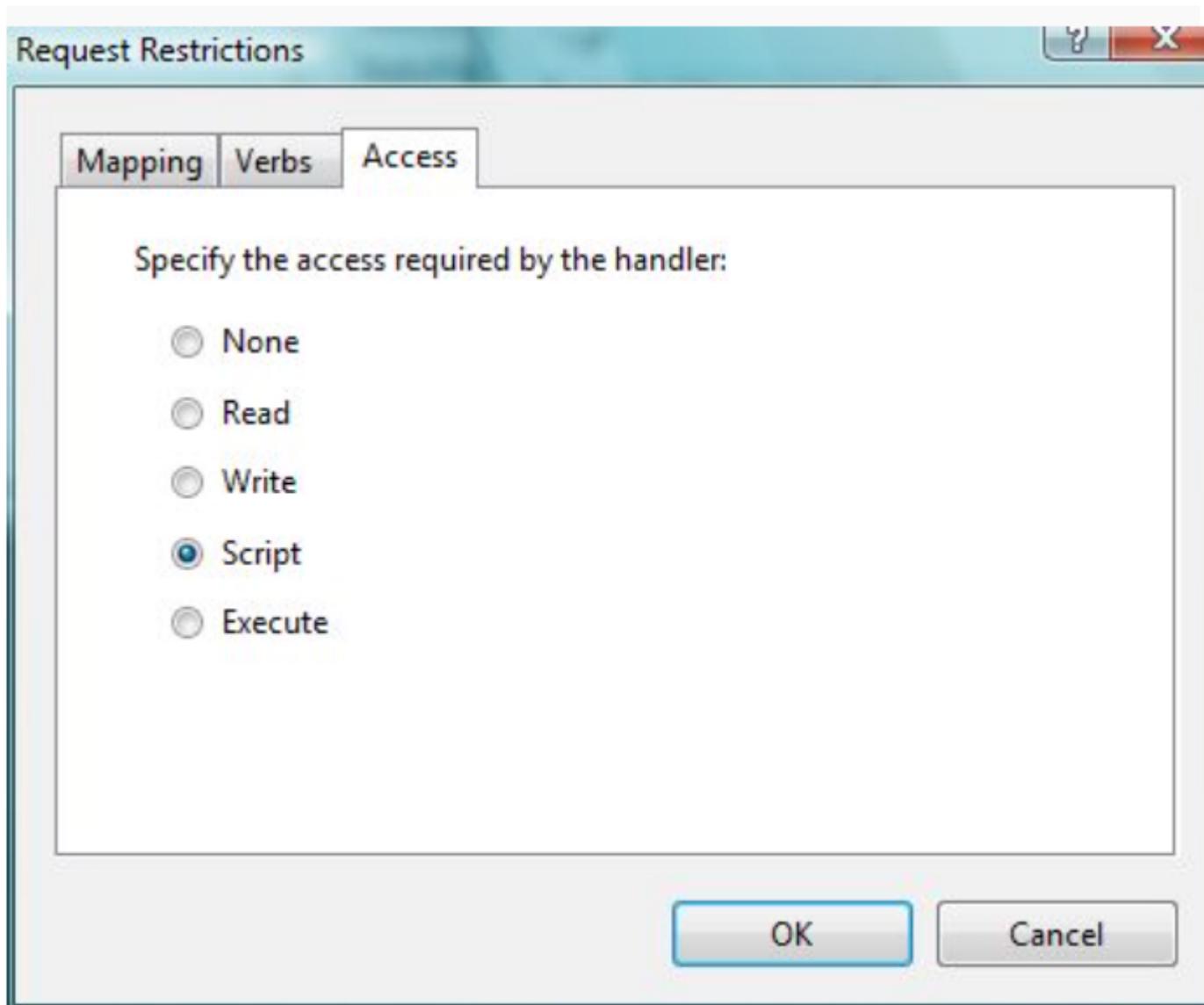
On the Verbs tab:

- Verbs: Select "All verbs"



On the Access tab

- Choose Script.



Click OK to close Request Restrictions and then click OK to close Add Script Map.

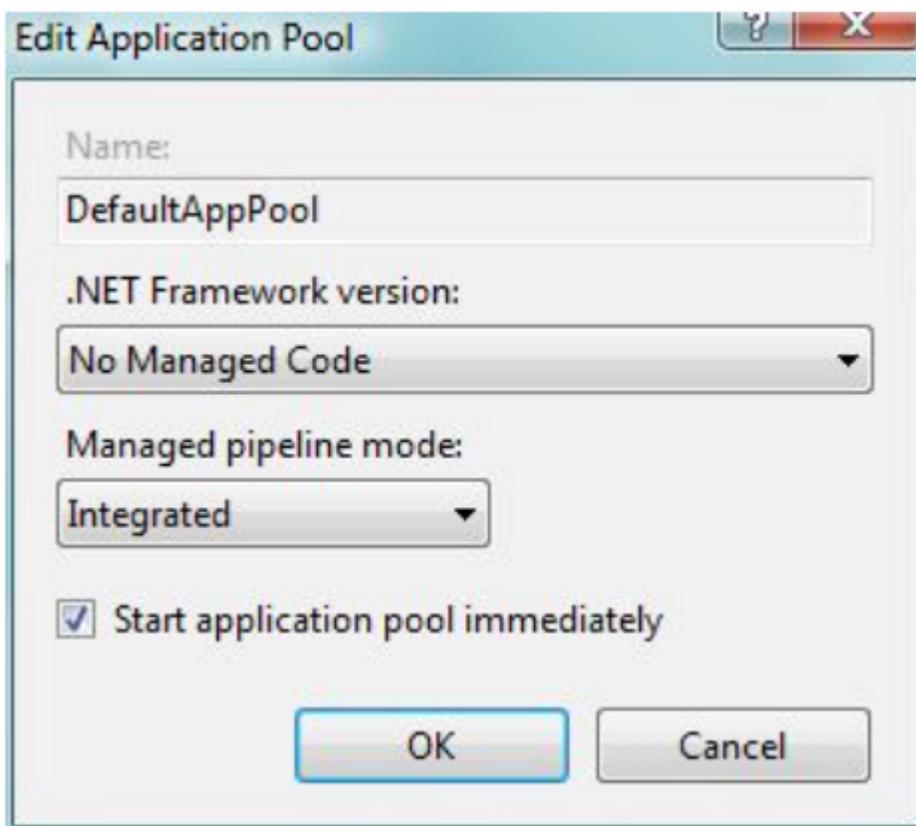
A dialog appears asking if you want to allow this ISAPA Extension. Choose Yes.

## Application Pools

In the left pane, select Application Pools.

Select your Site Name from the list of Application Pools and click Basic Settings to open the Edit Application Pool window to make these changes:

- .NET Framework version: No Managed Code
- Managed pipeline mode: Integrated



Click OK to close.

## Advanced Settings

Lastly, if you are on a 64-bit system:

- Click Advanced Settings
- Change Enable 32-bit Applications to True

## Application Pool Permissions

Your new site should appear in the TreeView on the left. Right-Click that site and select Explore. A window will appear where your web application will be placed, but first you need to set the correct permissions for this folder. You'll need the Application Pool name from above.

1. Right-Click inside the folder and select Properties.
2. Click the Security Tab
3. Click the Edit button.
4. Click the Add button.
5. In the "Enter the object names to select" field type IIS AppPool\RSWE
  1. (replace RSWE with your App Pool name if you used something different).
6. Click the Check Names button. If correct, the name you typed will be changed to just the name of your application pool and be underlined.

7. Click OK to dismiss the dialog.
8. Click OK to dismiss the Permissions dialog.
9. Click OK to dismiss the Properties dialog.

More information can be found about Application Pool Identities at: <http://learn.iis.net/page.aspx/624/application-pool-identities/>

## Deploy Your Application

Build your Xojo web application as a CGI application.

Copy it to the folder containing your web site (based on the above example, it is "c:\inetpub\wwwroot\webapp").

Launch your app from the web browser using this URL: <http://localhost:8080/iitestest.cgi>



Use the appropriate port that you have set in your web application build properties. 8080 is the default.

# Migrating from Visual Basic

Visual Basic (6 or earlier) and Visual Basic .NET use a language very similar to the Xojo language. You will notice that many of the commands are nearly the same, but there are differences as well.

## Similarities to Visual Basic

Visual Basic 6 (VB6) is no longer supported by Microsoft, which recommends you instead migrate to Visual Basic .NET (VB.NET). But Visual Basic .NET is large and complex, not to mention not cross-platform. Xojo is usually a better choice for Visual Basic 6 applications because it has the simplicity of VB6, but is a fully object-oriented language like VB.NET.

### Programming Language

To start with, the language syntax of VB is very similar to Xojo. You'll see familiar syntax for If..Then..Else, For..Next, While..Wend, Dim and many other commands. Someone who has used either VB6 or VB.NET will have no trouble understanding the Xojo programming language.

### Data Types

Although Xojo data types are not always named exactly the same as VB6 data types, all the equivalent types are there. For example, Integer is equivalent to a VB6 Long. Here is a mapping of some VB data types to Xojo data types:

VB Data Type	Xojo Data Type
Boolean	<u>Boolean</u>
Byte	<u>Byte</u>
Currency	<u>Currency</u>
Date	<u>Date_class</u>
Double	<u>Double</u>
Integer	<u>Int16</u>
Long	<u>Integer</u>
Object	Object
Single	<u>Single</u>
String	<u>Text</u>
Variant	<u>Auto</u>

## Controls

The default UI controls included with VB are, for the most part, also included with Xojo.

But Xojo also has several controls that are not included by default with VB. Of course, VB had plenty of additional, but Windows-specific, controls that could be added to its default setup and many of these controls can be added to Xojo using ActiveX, but they will also remain Windows-only.

Here is a list of some VB controls and their Xojo Desktop equivalents:

<b>VB Control</b>	<b>Xojo Desktop Control</b>	<b>Xojo Web Control</b>	<b>Xojo iOS Control</b>
PictureBox	Canvas ImageWell	WebCanvas WebImageView	iOSImageView iOSCanvas
Label	Label	WebLabel	iOSLabel
TextBox	TextField TextArea	WebTextField WebTextArea	iOSTextField iOSTextArea
Frame	GroupBox	WebGroupBox	n/a
CommandButton	PushButton BevelButton	WebButton	iOSButton
CheckBox	CheckBox	WebCheckBox	iOSSwitch
Listbox	Listbox PopupMenu	WebListBox	iOSTable
HScrollBar VScrollBar	ScrollBar	WebScrollbar	n/a
Timer	Timer	WebTimer	Timer
Shape	Oval Rectangle	WebOval WebRectangle	iOSOval iOSRectangle
WebBrowser	HTMLViewer	WebHTMLViewer	iOSHTMLViewer
TreeView	Listbox	n/a	n/a
Toolbar	Toolbar	WebToolbar	iOSToolbar
MediaPlayer	MoviePlayer	WebMoviePlayer	n/a

## Differences from Visual Basic

A big difference is that Xojo cannot create DLLs, ActiveX controls or any kind of shared libraries. Since these are all Windows-specific technologies, they are not useful for cross-platform applications.

Xojo can access DLLs and many ActiveX controls, but using them means your application will only run on Windows and cannot be cross-platform.

Of course, Xojo can easily create web applications, something VB6 cannot do.

## File I/O

File input and output in VB6 uses direct, path-based access to files. This is not something that works for cross-platform applications, so Xojo consolidates all file processing into a few classes: [FolderItem](#), [TextInputStream](#), [TextOutputStream](#) and [BinaryStream](#).

## Data Types

Xojo is a strongly typed programming language. VB6 (and older versions) would allow you to use a variable that had not been previously declared. It would infer a type based on a special character in its name (name\$ would be a String, for instance). Before trying to migrate VB6 code, you should use the `OPTION EXPLICIT` command to make sure that all your variables are declared.

# Visual Basic Migration Assistant

Visual Basic Migration Assistant (VBMA) is a free tool that can help you to begin migrating VB6 and VB.NET code to a Xojo desktop project. VBMA creates a Xojo project from the contents of your VB project. Specifically, it moves over forms, modules, classes and their code.

- [Download Visual Basic Migration Assistant](#)

## What it Does

VBMA takes the selected VB project and creates a Xojo XML project file containing the forms, modules, classes and source code from the VB project. The purpose of this tool is to get your project into Xojo so that you can work on it from a single place.

 VBMA does not create a working Xojo application from VB code.

Since VB forms do not exactly match Xojo windows, VBMA maps VB controls to their equivalent Xojo controls as it migrates the project.

Source code is not converted or modified in any way. The code is migrated to the Xojo project, but is completely commented out and is primarily for reference.

## For Best Results

VB 5 and 6 and VB.NET are supported. If you are using an older version of VB, upgrade your code to a later version before attempting to use VBMA.

Review and try to reduce your usage of 3rd party VB controls. Not only are they not cross-platform, but not all of them work with Xojo.

## Converting a Project

When you launch VBMA, you are presented with a wizard that walks you through the process.

1. After the first screen of instructions, you select the project to migrate.
2. Select the “Import Project” button and choose your VB project file. Alternatively, you can drag individual files to the List or you can use the “Add Item” button to select individual files.
3. Specify the encoding as necessary. This is the encoding/language that was used to create the VB project. This is especially important if your VB project uses any non-English filenames.
4. Click the Next button to go to the Control Mapping screen.
5. VBMA analyzes the VB project and displays the type of controls that it is using. In the Mapping screen, you can select the Xojo control to use for each VB control.
6. You can choose to save the Control Map to a file so that you can use it again for other projects.
7. Click the Migrate button to migrate the VB project to a Xojo XML project file. You will be prompted for a location to save the file.
8. When VBMA is finished it will attempt to have Xojo open the project file.
9. With your VB project in Xojo, you can now begin work on creating a Xojo version.

# Migrating from Microsoft Access

Microsoft Access is database software that runs on Windows and is part of specific versions of Office. It is often used to create specialized in-house database applications. But Access cannot create real, stand-alone applications. If you have an Access application and are running into its limitations, Xojo is a great choice to take your application to the next level.

## Similarities

Access has a form designer, database designer and a programming language (VBA: Visual Basic for Applications).

Xojo has all these components as well, but expands on each of them. It has a form designer with many more controls than Access provides and allows you to layout your user interface in any way you want. It uses SQLite as its built-in database and has a database designer for designing your tables. And of course, Xojo has a much more robust programming language.

## Migrating

Migrating an Access application is typically a three-step process where you migrate the database itself, the forms that are used to manipulate the data and the source code.

### Database

When migrating a Microsoft Access application, you first need to consider the database. If you are using the Access “Jet” database engine, you will most likely want to migrate it to another database engine. Although you can connect to a Jet database using ODBC or ADO on Windows, Jet is not a cross-platform database format. OS X can only connect to a Jet database using ODBC in read-only mode.

Your best bet in this case is to use [SQLite](#), which is much faster than the Access Jet database and is fully cross platform. You can easily migrate your database tables and data from an Access database to SQLite. This can be done using [ODBC](#), ADO or a variety of 3rd party products.

If your Access database is connecting to another database as its data source, then you can use the ODBC plugin and an ODBC driver to connect. Or you can use the built-in plugins for connecting to [PostgreSQL](#), [MySQL](#), Oracle and [Microsoft SQL Server](#).

### Forms

Your Access forms are likely used to edit data in tables. You can recreate these forms as Windows, Web Pages or Views (iOS) in your Xojo application.

### Source Code

Access is programmed using the Visual Basic for Applications language which is quite similar to the [Xojo programming language](#). You will have to rewrite your code, but at the same time will find the Xojo programming

language to be familiar in its syntax and commands.

## Learn More

To learn more about Xojo and how you can use it to replace Microsoft Access, check out these topics:

- [Getting Started](#)
- [Desktop App Tutorial](#)
- [Web App Tutorial](#)
- [iOS App Tutorial](#)
- [Using SQLite](#)

# Migrating from FileMaker

FileMaker is a database tool that runs on both Windows and OS X. It is often called the OS X version of Access. Much like with Access, FileMaker has its own database engine, form designer and scripting language.

## Migrating

Migrating a FileMaker application is typically a three-step process where you migrate the database itself, the forms that are used to manipulate the data and the scripting code.

### Database

When migrating a FileMaker application, you first need to consider the database. Although you can connect to a FileMaker database using ODBC, you will need to get the appropriate drivers. As an alternative, you can migrate your data to [SQLite](#) (a fast, cross-platform database) by first converting the FileMaker data to XML.

You can also connect to a FileMaker database by using Custom Web Publishing and an API.

### Forms

Your FileMaker forms are likely used to edit data in tables. You can recreate these forms as Windows (or Web Pages or iOS Views) in your Xojo application. In most cases you will use Label, TextField and TextArea to recreate form fields, but there are lots of other available controls as well.

Either way, you would likely have your application connect to the database at startup and then populate the form with the first record. Next and Previous buttons can be added to fetch and display the appropriate information from the database.

### Scripting Code

FileMaker is programmed using a scripting language that is somewhat similar to the Xojo programming language. You will have to rewrite your code, but at the same time will find the Xojo programming language to be familiar.

Here are some FileMaker commands and their Xojo equivalents:

FileMaker Command	Xojo Command
Exit Script	Return
Set Error Capture	<a href="#">Try..Catch..End Try</a>
Set Variable	<a href="#">Dim</a>
If..End If	<a href="#">If..Then..Else..End if</a>
Loop..End Loop	<a href="#">Loop..Until</a>
Go to Field	TextField.SetFocus

FileMaker Command	Xojo Command
Field assignment	TextField.Text = "value"

## FileMaker Library (Xojo.FM)

To make it easier for developers to transition from Xojo to FileMaker, there is an open-source library on GitHub that implements many of the FileMaker functions so that you can use them by name in Xojo.

To use the library, download the library from GitHub and open the project. In the Navigator, select the FM module and copy it. Then switch to your project and paste the module to it.

You can refer to any of the functions in the module by using the FM prefix.

## FileMaker Commands to Xojo Commands

This section highlights FileMaker functions that have an equivalent command or function built-in to Xojo. If there is no built-in equivalent, the corresponding function in Xojo.FM is noted.

### Aggregate Functions

FileMaker	Xojo	Xojo.FM
Average	n/a	FM.Average
Min	n/a	FM.Min
Max	n/a	FM.Max
Sum	n/a	FM.Sum

### Container Functions

Base64Decode, Base64Encode

FileMaker	Xojo	Xojo.FM
Base64Decode	DecodeBase64	FM.Base64Decode
Base64Encode	EncodeBase64	FM.Base64Encode

### Date Functions

Date, Day, DayOfWeek, DayOfYear, Month, Year

FileMaker	Xojo	Xojo.FM
Date	Xojo.Core.Date	FM.Date
Day	Xojo.Core.Date.Day	FM.Day
DayName	n/a	FM.DayName

<b>FileMaker</b>	<b>Xojo</b>	<b>Xojo.FM</b>
DayOfWeek	Xojo.Core.Date.DayOfWeek	FM.DayOfWeek
DayOfYear	Xojo.Core.Date.DayOfYear	FM.DayOfYear
Month	Xojo.Core.Date.Month	FM.Month
MonthName	n/a	FM.MonthName
WeekOfYear	n/a	FM.WeekOfYear
Year	Xojo.Core.Date.Year	FM.Year

## Financial Functions

FV, NPV, PMT, PV

## Get Functions

GetApplicationVersion, GetCurrentDate, GetCurrentTime, GetCurrentTimestamp, GetDesktopPath, GetDocumentsPath, GetPreferencesPath, GetScreenHeight, GetScreenWidth, GetSystemDrive, GetWindowContentHeight, GetWindowContentWidth, GetWindowHeight, GetWindowLeft, GetWindowTop, GetWindowWidth

<b>FileMaker</b>	<b>Xojo</b>	<b>Xojo.FM</b>
Get(ApplicationVersion)	XojoVersionString	FM.Get.ApplicationVersion
Get(CurrentDate)	Xojo.Core.Date.Now	FM.Get.CurrentDate
Get(CurrentTime)	Xojo.Core.Date.Now	FM.Get.CurrentTime
Get(CurrentTimestamp)	Xojo.Core.Date.Now	FM.Get.CurrentTimestamp
Get(DesktopPath)	SpecialFolder.Desktop	FM.Get.DesktopPath
Get(Device)	n/a	FM.Get.Device
Get/DocumentsPath)	SpecialFolder.Documents	FM.Get.DocumentsPath
Get/DocumentsPathListing)	n/a	FM.Get.DocumentsPathListing
Get(PreferencesPath)	n/a	FM.Get.PreferencesPath
Get(ScreenHeight)	Screen(0).Height	FM.Get.ScreenHeight
Get(ScreenScaleFactor)	n/a	FM.Get.ScreenScaleFactor
Get(ScreenWidth)	Screen(0).Width	FM.Get.ScreenWidth
Get(SystemDrive)	Volume(0)	FM.Get.SystemDrive
Get(SystemPlatform)	n/a	FM.Get.SystemPlatform

<b>FileMaker</b>	<b>Xojo</b>	<b>Xojo.FM</b>
Get(SystemVersion)	n/a	FM.Get.SystemVersion
Get(TemporaryPath)	n/a	FM.Get.TemporaryPath
Get(UUID)	n/a	n/a
Get(WindowContentHeight)	Window.Bounds.Height	FM.Get.WindowContentHeight
Get(WindowContentWidth)	Window.Bounds.Width	FM.Get.WindowContentWidth
Get(WindowHeight)	Window.Height	FM.Get.WindowHeight
Get(WindowLeft)	Window.Left	FM.Get.WindowLeft
Get(WindowTop)	Window.Top	FM.Get.WindowTop
Get(WindowWidth)	Window.Width	FM.Get.WindowWidth

## Logical Functions

<b>FileMaker</b>	<b>Xojo</b>	<b>Xojo.FM</b>
Case	Select...Case	
Choose	Select...Case	
Evaluate		FM.Evaluate
ExecuteSQL	Database.SQLSelect Database.SQLExecute	
If	If...Then If operator	

## Number Functions

<b>FileMaker</b>	<b>Xojo</b>	<b>Xojo.FM</b>
Abs	Math.Abs	
Ceiling	Math.Ceil	FM.Ceiling
Exp	Math.Exp	
Floor	Math.Floor	
Int	CType(value, Integer)	FM.Int
Ln	Math.Log	FM.Ln
Mod	Mod	

<b>FileMaker</b>	<b>Xojo</b>	<b>Xojo.FM</b>
Random	Math.RandomInt	FM.Random
Round	Math.Round	
Sign	Math.Sign	
Sqrt	Math.Sqrt	
Truncate	n/a	FM.Truncate

## Text Functions

<b>FileMaker</b>	<b>Xojo</b>	<b>Xojo.FM</b>
Char	Text.FromUnicodeCodePoint	FM.Char
Code	Text.Codepoints	FM.Code
Exact	Text.Compare	FM.Exact
Filter	n/a	FM.Filter
FilterValues	n/a	FM.FilterValues
GetAsDate	Xojo.Core.Date.FromText	FM.GetAsDate
GetAsNumber	Integer.FromText Integer.Parse Double.FromText Currency.FromText	FM.GetAsNumber
GetAsText	Integer.ToText Double.ToText Currency.ToText	FM.GetAsText
GetAsURLEncoded	EncodeURLComponent	FM.GetAsURLEncoded
GetValue	Arrays	
Left	Text.Left	FM.Left
LeftValues	n/a	FM.LeftValues
LeftWords	n/a	FM.LeftWords
Length	Text.Length	
Lower	Text.Lowercase	FM.Lower
Middle	Text.Mid	FM.Middle
MiddleValues	n/a	FM.MiddleValues
MiddleWords	n/a	FM.MiddleWords

<b>FileMaker</b>	<b>Xojo</b>	<b>Xojo.FM</b>
PatternCount	RegEx	
Position	Text.IndexOf	FM.Position
Proper	Text.Titlecase	FM.Proper
Quote	n/a	FM.Quote
Replace	Text.Replace	FM.Replace
Right	Text.Right	
RightValues	n/a	FM.RightValues
RightWords	n/a	FM.RightWords
Substitute	Text.ReplaceAll	FM.Substitute
Trim	Text.Trim	
TrimAll	n/a	FM.TrimAll
Upper	Text.Uppercase	FM.Upper
ValueCount	CountFields	FM.ValueCount
WordCount	n/a	FM.WordCount

## Text Formatting Functions

<b>FileMaker</b>	<b>Xojo</b>	<b>Xojo.FM</b>
RGB	Color.RGB	
TextColor	Change the TextColor property for the specific control.	FM.TextColor
TextFont		
TextSize		

## Time Functions

<b>FileMaker</b>	<b>Xojo</b>	<b>Xojo.FM</b>
Hour	Date.Hour	
Minute	Date.Minute	
Seconds	Date.Seconds	
Time	Date	
n/a	Ticks	
n/a	Microseconds	

## Trigonometric Functions

FileMaker	Xojo	Xojo.FM
Acos	Math.Acos	
Asin	Math.Asin	
Atan	Math.Atan	
Cos	Math.Cos	
Degrees	n/a	FM.Degrees
Pi	n/a	FM.Pi
Radians	n/a	FM.Radians
Sin	Math.Sin	
Tan	Math.Tan	

## Misc

FileMaker Command	Xojo Command	Notes
Base64Decode	<a href="#">DecodeBase64</a>	
Base64Encode	<a href="#">EncodeBase64</a>	
Date	Xojo.Core.Date	This is a class. To create a specific date:  <pre>Dim d As New Xojo.Core.Date(2014, 10, 10, Xojo.Core.TimeZone.Current)</pre>
Day	Xojo.Core.Date.Day	

## Learn More

To learn more about Xojo and how you can use it to replace Microsoft Access, check out these topics:

- [Getting Started](#)
- [Desktop App Tutorial](#)
- [Web App Tutorial](#)
- [iOS App Tutorial](#)
- [Using SQLite](#)
- [Develop iOS Apps that Integrate with FileMaker](#)

# Migrating from Visual FoxPro

Visual FoxPro (VFP) is a Windows programming tool made by Microsoft. It has been given its end-of-life and is no longer supported by Microsoft. VFP has its own tightly integrated database engine, a form designer and a programming language.

## Migrating

Migrating a Visual FoxPro application is typically a three-step process where you migrate the database itself, the forms that are used to manipulate the data and the scripting code.

### Database

When migrating a VFP application, you first need to consider the database. You can connect to a VFP database using ODBC on Windows, but it makes more sense to migrate your data to SQLite, which is a fast, cross-platform database.

### Forms

Your VFP forms are likely used to edit data in tables. You can recreate these forms as Windows (or web pages) using drag and drop just as you can with VFP.

For desktop applications, you may find the DataControl control to be a simple way to map your fields to database tables and columns without having to write a lot of code.

### Source Code

VFP is programmed using a proprietary language that is quite similar to the Xojo programming language. You will have to rewrite your code, but at the same time will find the Xojo programming language to be familiar.

Cully Technologies has a tool that can help you [migrate your Visual Fox Pro projects to Xojo](#).

## Language Syntax

The syntax of the two languages is different, but the concepts are quite similar. For example, to create an instance of a new class in VFP, you might use this code:

```
LOCAL oMyClass  
oMyClass = CREATEOBJECT("MyClass")
```

In Xojo you would write:

```
Dim oMyClass As New MyClass
```

The VFP MessageBox command is similar. Instead of writing this:

```
MessageBox("Hello, World!")
```

You would write this:

```
MsgBox("Hello, World!")
```

Here are some other VFP commands and their Xojo equivalents:

VFP Command	Xojo Command
ON ERROR	Exception
TRY..CATCH..END TRY	<a href="#">Try..Catch..End Try</a>
DO WHILE..ENDDO	<a href="#">While..Wend</a>
FOR EACH..ENDFOR	<a href="#">For Each..Next</a>
FOR..ENDFOR	<a href="#">For..Next</a>
IF..ENDIF	<a href="#">If..End If</a>
LOOP	<a href="#">Continue</a>
DECLARE	<a href="#">Dim</a>
DO CASE..END CASE	<a href="#">Select Case..End Select</a>



Special thanks to Kevin Cully of Cully Technologies for assistance with the information in this section.

## Learn More

To learn more about Xojo and how you can use it to replace Microsoft Access, check out these topics:

- [Getting Started](#)
- [Desktop App Tutorial](#)
- [Web App Tutorial](#)
- [iOS App Tutorial](#)
- [Using SQLite](#)

# Eddie's Electronics Web Service

In order to help you with testing a web service, the Eddie's Electronics web service is available for use with your apps. This web service is hosted on Xojo Cloud as a Xojo web app and uses a SQLite database for its data.

## Base URL

All web services calls are done through the base URL:

```
http://demos.xojo.com/EEWS/index.cgi/
```

## GetAllCustomers

HTTP Method: GET or POST

This call returns a JSON document containing the object "GetAllCustomers" which is a key-value store of a customer ID and its data, including FirstName, LastName, City, State and Zip.

### Sample Code

This code sends the web service request:

```
' Add Xojo.IO.HTTPSocket subclass to your project.  
' Drag it to a layout and name it MySocket.  
MySocket.Send("POST", "http://demos.xojo.com/EEWS/index.cgi/GetAllCustomers")
```

This code (in the PageReceived event handler of the socket added to your layout) processes the resulting Dictionary:

```
Dim customers As Xojo.Core.Dictionary  
customers = json.Value("GetAllCustomers")  
  
For Each entry As Xojo.Core.DictionaryEntry In customers  
  Dim custDict As Xojo.Core.Dictionary = entry.Value  
  Dim firstName As Text = custDict.Value("FirstName")  
Next
```

### Sample JSON Result

```
{  
  "GetAllCustomers":  
  {  
    "10174":  
    {  
      "FirstName": "Abdul",
```

```

        "LastName": "Mcconnell",
        "City": "Colorado Springs",
        "State": "CO",
        "Zip": "80935"
    },
    "10179":
    {
        "FirstName": "Abel",
        "LastName": "Alexander",
        "City": "Arcadia",
        "State": "IA",
        "Zip": "51430"
    }
}
}

```

## GetCustomer

HTTP Method: POST

This call returns all the details for a single customer. You specify the customer by providing a simple JSON document (as request content) containing the ID of the customer you want.

### Sample Xojo Code

This code (using the same socket used in the example above) sends a request for customer data:

```
Dim cust As New Xojo.Core.Dictionary
cust.Value("ID") = 10179
```

```
Dim json As Text
json = Xojo.Data.GenerateJSON(cust)
```

```
Dim data As MemoryBlock
data = Xojo.Core.TextEncoding.UTF8.ConvertTextToData(json)
```

```
MySocket.SetRequestContent(data, "application/x-www-form-urlencoded")
MySocket.Send("POST", "http://demos.xojo.com/EEWS/index.cgi/GetCustomer")
```

This code parses the resulting data:

```
Dim custInfo As Xojo.Core.Dictionary
custInfo = json.Value("GetCustomer")
```

```
Dim id As Integer
Dim firstName, lastName As Text
```

```
For Each entry As Xojo.Core.DictionaryEntry In custInfo
  Select Case entry.Key
    Case "ID"
      id = Integer.FromText(entry.Value)
    Case "FirstName"
      firstName = entry.Value
    Case "LastName"
      lastName = entry.Value
  End Select
Next
```

## Sample JSON Request

```
{"ID": 10179}
```

## Sample JSON Result

```
{
  "GetCustomer":
  {
    "ID": "10179",
    "FirstName": "Abel",
    "LastName": "Alexander",
    "City": "Arcadia",
    "State": "IA",
    "Zip": "51430",
    "Phone": "1-261-529-7025",
    "Email": "elit.sed@aliquamarcu.edu",
    "Photo": "Base64EncodedData",
    "Taxable": "0"
  }
}
```

# Using Inno Setup to Create a Windows Installer

Inno Setup is a free tool for creating Windows installers. It is a great way to create Windows installers for your desktop applications so that you can easily deploy them. The following script can be used with InnoSetup to create an installer for your desktop applications.

```
; Sample script for creating an installer for a desktop application
; To use this script, replace "XojoApp" with the name of your application.
```

```
[Setup]
; NOTE: The value of AppId uniquely identifies this application.
; Do not use the same AppId value in installers for other applications.
; (To generate a new GUID, click Tools | Generate GUID inside the IDE.)
AppId={{B104D606-3C81-45CF-A209-82C315FF3752}}
AppName=XojoApp
AppVerName=XojoApp 1.0
AppPublisher=XojoApp Company
AppPublisherURL=
AppSupportURL=
AppUpdatesURL=
DefaultDirName={pf}\XojoApp
DefaultGroupName=XojoApp
OutputDir=C:\Users\You\Development\XojoApp\Installer
OutputBaseFilename=SetupXojoApp
; If you have an End User License Agreement (EULA) that you want the user to agree to
before letting the install continue,
; put the path to it here.
LicenseFile=
Compression=lzma
SolidCompression=yes
```

```
[Languages]
Name: "english"; MessagesFile: "compiler:Default.isl"
```

```
[Tasks]
Name: "desktopicon"; Description: "{cm:CreateDesktopIcon}"; GroupDescription:
"{cm:AdditionalIcons}"; Flags: unchecked
```

; These directories will be created by the installer inside the DefaultDirName (defined above).

```
[Dirs]
Name: "{app}\XojoApp Libs"
```

```
; These are the files to include. By default you want to include the EXE and the Libs
folder
; but you can include any other files you like as well.
; Be sure to change the path to point to your built application.
```

---

  
[Files]

Source: "C:\Users\You\Development\XojoApp\BuildsFolder\Windows\XojoApp.exe"; DestDir: "{app}"; Flags: ignoreversion

Source: "C:\Users\You\Development\XojoApp\BuildsFolder\Windows\XojoApp Libs\\*";  
DestDir: "{app}\XojoApp Libs"; Flags: ignoreversion recursesubdirs createallsubdirs

; NOTE: Don't use "Flags: ignoreversion" on any shared system files

; Creates icons/links in the Start Menu and/or the desktop if the user chooses during installation.

## [Icons]

Name: "{group}\XojoApp"; Filename: "{app}\XojoApp.exe"

Name: "{commondesktop}\XojoApp"; Filename: "{app}\XojoApp.exe"; Tasks: desktopicon

; Give the user the option to run the app after the installation is finished.

## [Run]

Filename: "{app}\XojoApp.exe"; Description: "{cm:LaunchProgram,XojoApp}"; Flags: nowait postinstall skipifsilent

# Elevating User Access Control

Do you need to run a Xojo Application on Windows with Elevated User Access Control (UAC)?

This method creates a Visual Basic Script (VBS) to launch your program with Elevated Access Rights. It does not bypass the security of Windows! It causes the UAC dialog to appear so the user can agree to or enter administrator credentials. The *program* parameter needs to be the NativePath of the executable file.

This method can be used to launch Console apps to do things like update "protected" registry items and/or install programs.

```
Sub ExecuteWithUAC(program As String, args As String)
  Dim f As FolderItem
  Dim t As TextOutputStream
  Dim script As String = "Set objShell = CreateObject(""Shell.Application"")" +
  EndOfLine _
  + "objShell.ShellExecute ""<Program>"" , ""<Args>"" , """" , ""runas"" , 1" + EndOfLine

  Dim s As String
  f = GetTemporaryFolderItem
  f = GetFolderItem(f.NativePath + ".vbs")
  t = TextOutputStream.Create(f)
  s = ReplaceAll(script, "<Program>", program)
  s = ReplaceAll(s, "<Args>", ReplaceAll(args, chr(34), chr(34) + chr(34)))
  t.WriteLine(s)
  t.Close

  Dim sh As New Shell
  sh.Mode = 0
  sh.Timeout = 10000
  sh.Execute("Wscript.exe " + f.NativePath)
  f.Delete
End Sub
```

Thanks to Wayne Golding for this code!

# Xojo Reference Guide

The Xojo Reference Guide gives you fast access to information about the Xojo Programming Language and its frameworks. This includes commands, operators, constants, controls, classes and so forth.

If you are new to Xojo and want to learn about it in a more structured manner, you should instead start by reading the User Guide.

## Topics

- [Language](#)
  - [Commands](#)
  - [Compiler Constants](#)
  - [Conditional Compilation](#)
  - [Data Types](#)
  - [Literals](#)
  - [Operators](#)
  - [Pragma Directives](#)
  - [Reserved Words](#)
- [Framework](#)
  - [iOS Framework](#)
  - [Xojo Framework](#)
  - [Classic Framework](#)

# Language

The Xojo Programming Language is the same across all project types.

Topics described in this section:

- [Commands](#)
- [Data Types](#)
- [Literals](#)
- [Operators](#)
- [Compiler Constants](#)
- [Conditional Compilation](#)
- [Compiler Error Messages](#)
- [Pragma Directives](#)

# Commands

This section describes commands that are part of the Xojo language.

## Command Summary

Commands

[AddHandler](#)  
[App](#)  
[Array](#)  
[As](#)  
[Assigns](#)  
[ByRef](#)  
[ByVal](#)  
[Break](#)  
[Call](#)  
[Case](#)  
[Catch](#)  
[Const](#)  
[Continue](#)  
[End](#)  
[CurrentMethodName](#)  
[Declare](#)  
[Dim](#)  
[Do](#)  
[Do...Loop](#)  
[DownTo](#)  
[Each](#)  
[Else](#)  
[Elseif](#)  
[End](#)  
[Exit](#)  
[Extends](#)  
[False](#)  
[Finally](#)  
[For](#)  
[For...Next](#)  
[For Each...Next](#)  
[Global](#)  
[If](#)  
[If...Then...Else](#)  
[In](#)  
[Lib](#)  
[Line Continuation](#)  
[Loop](#)  
[Me](#)  
[Next](#)  
[Nil](#)  
[Optional](#)  
[ParamArray](#)  
[Raise](#)  
[RaiseEvent](#)  
[ReDim](#)  
[Rem](#)  
[RemoveHandler](#)  
[Return](#)  
[Select](#)  
[Select...Case](#)  
[Self](#)  
[Soft](#)  
[Static](#)  
[Step](#)  
[Super](#)  
[Then](#)  
[To](#)  
[True](#)  
[Try](#)  
[Try...Catch](#)  
[Ubound](#)  
[Until](#)  
[Using](#)  
[Wend](#)  
[While](#)  
[While...Wend](#)

The Xojo Language and its commands are all case-insensitive.

# AddHandler (Keyword)

Directs the handling of an event to a delegate method.

## Keyword Summary

Name	AddHandler
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">RemoveHandler</a> <a href="#">Timer</a>

## Syntax

```
AddHandler eventName, delegateMethod
```

### Parameters

<i>eventName</i>	The name of the event that you are handling.
<i>delegateMethod</i>	The name of the method that will be used to handle the event.

## Notes

Generally you will create a subclass to expose event handlers for you to implement. AddHandler is provides a way for you to instantiate a class in code and still have access to its event handlers.

The `delegateMethod` method declaration must consist of:

- A reference to the object being handled. This is the sender object and it must be the first parameter in the delegate and must be the correct type.
- The same parameters of the event you are handling should follow next in the same order with the same types. The parameter names can be changed.
- If the event you are handling returns a value, then the delegate method must also return the same value type.

A handler can handle any number of events, but an event can only have one event handler. To change the event handler for an event after you have added one, you must first remove the existing handler with [RemoveHandler](#). You can then add the new handler.

## Exceptions

RuntimeException	If you implement an event handler that is already implemented without first removing it using <a href="#">RemoveHandler</a> .
------------------	---

---

## Sample Code

This example lets you handle the `Timer.Action` event without creating a `Timer` subclass.

Start by adding a property to the layout:

```
MyTimer As Xojo.Core.Timer</code>
```

Next, add a `ProgressBar` to the layout. The `Timer` will simply update the `ProgressBar`.

In the `Open` event handler of the layout, instantiate the `Timer` and indicate that its `Action` event handler should be handled by a method, `TimerAction`, that you will add to the layout:

```
MyTimer = New Xojo.Core.Timer  
MyTimer.Period = 1000  
MyTimer.Mode = Xojo.Core.Timer.Modes.Multiple
```

```
AddHandler MyTimer.Action, AddressOf TimerAction</code>
```

Now add the `TimerAction` method to the layout:

```
Sub TimerAction(sender As Xojo.Core.Timer)  
  If ProgressBar1.CurrentValue < ProgressBar1.MaxValue Then  
    ProgressBar1.CurrentValue = ProgressBar1.CurrentValue + 1  
  Else  
    ' Stop Timer and Remove the handler  
    sender.Mode = Xojo.Core.Timer.Mode.Off  
    RemoveHandler MyTimer.Action, AddressOf TimerAction  
  End If  
End Sub
```

Remember that the first parameter to `TimerAction` must be of the type of the original object, in this case a `Timer`.

When you run the project, the `ProgressBar` is updated once per second.

# App

Returns the instance of the Application object used by the app. This varies by project type.

## Summary

Name	App
Type	Method
Project Types	All
Platforms	All
Related	<a href="#">Application</a> <a href="#">ConsoleApplication</a> <a href="#">iOSApplication</a> <a href="#">ServiceApplication</a> <a href="#">WebApplication</a>

## Notes

This method returns a specific instance of an application class, depending on the app type:

App Type	Class Instance
iOS	<a href="#">iOSApplication</a>
Desktop	<a href="#">Application</a>
Web	<a href="#">WebApplication</a>
Console	<a href="#">ConsoleApplication</a>

## Sample Code

In a desktop or web app, turns on AutoQuit so that the app quits when all its Windows are closed or all its Sessions are disconnected:

```
App.AutoQuit = True
```

# Array (Keyword)

Assigns a list of values to consecutive elements of a one-dimensional array.

## Keyword Summary

Name	Array
Type	Keyword
Project Types	All
Platforms	All
Related	Dim, <a href="#">Arrays</a>

## Syntax

```
arrayResult = Array(elementList)
```

### Parts

<i>arrayResult</i>	The name of the array to populate with the items in <i>elementList</i> .
<i>elementList</i>	A comma-delimited list of values that are used to populate the array. The data type of the elements should match the type of <i>arrayResult</i> or you will get a <code>TypeMismatch</code> error.

## Notes

Only one-dimensional arrays can be populated by `Array`.

`Array` provides the same functionality as separate assignment statements for each element of the array, beginning with element zero and proceeding consecutively. If there are more elements in the array than items in *elementList*, then the "extra" items in the array are not modified.

### Type Considerations

If *elementList* contains numeric data of different types and/or a mixture of signed and unsigned integers, `Array` will use the lowest common data type that accomodates all the elements.

For example, because hexadecimal numbers are unsigned integers (`UInteger`), the following code generates a `Type Mismatch` error:

```
Dim hexNums() As Integer
hexNums = Array(&h1, &h2)</code>
```

You need to be careful about the size of the array and whether or not the integer is signed. Instead you should declare the array as UInt32:

```
Dim hexNums() As UInt32
hexNums = Array(&h1, &h2)</code>
```

You can also choose to cast the elements to a specific type:

```
Dim hexNums() As Integer
hexNums = Array(Int32(&h1), &h2) // Once you convert one value type, the rest will
follow</code>
```

Similarly, this will work:

```
Dim hexNums() As integer
hexNums = Array(1, &h2)
```

## Sample Code

Populate arrays with teams and names:

```
Dim teams() As Text = Array("Red Sox", "Yankees", "Orioles", "Rays", "Blue Jays")
```

```
Dim names() As Text
names = Array("Bob", "Bill", "Brad", "Ben")
```

# As (Keyword)

Declares a variable or parameter name to be of a particular data type. Often used as part of these commands:

- [Dim](#)
- [For...Next](#)
- [For Each...Next](#)
- [Properties](#)
- [Methods](#)

## Keyword Summary

Name	As
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">For...Next</a> , <a href="#">For Each...Next</a> , <a href="#">Select...Case</a>

## Sample Code

Usage of As in For...Next and Dim commands:

```
For i As Integer = 1 To 10
```

```
    ...
```

```
Next
```

```
Dim value As Integer = 10
```

# Assigns (Keyword)

Used to indicate that a value will be passed via a parameter to a method with the Assignment operator.

## Keyword Summary

Name	Assigns
Type	Keyword
Project Types	All
Platforms	All
Related	Function, Sub

## Syntax

```
Assigns parameter As dataType
```

## Notes

Use the Assigns command when you want to assign a value to a parameter of a method with the equals sign rather than supply it as an argument.

The Assigns command can appear in a Method declaration dialog box for the only parameter passed to the method or for the last parameter, if more than one parameter is being passed. When Assigns is used, you use a different syntax when calling the method. The value for the Assigns parameter must be passed using the assignment operator rather than as an argument.

## Sample Code

The follow declaration requires the last parameter to be assigned rather than supplied in the parameter list:

```
Sub ChangeVolumn(a As Integer, b As Integer, Assigns c As Integer)
```

```
' Usage
```

```
ChangeValue(5, 4) = 10
```

```
' ChangeValue(5, 4, 10) would result in a syntax error
```

# ByRef (Keyword)

Used to pass a parameter by reference.

## Keyword Summary

Name	ByRef
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">ByValue</a> , Function, Sub

## Syntax

```
ByRef parameter
```

## Notes

To pass a parameter by reference, you precede a parameter name with the ByRef keyword in the parameter declaration for the method. When you pass information by reference, you actually pass a pointer to the object containing the information. The practical advantage of this technique is that any changes to the value of the parameter by the method are available to the method that called it.

 ByRef and ByVal are only applicable to simple data types (Integer, String, Double, etc.) All arrays and object types are reference types, effectively behaving as if ByRef was used with a simple data type.

## Sample Code

```
// A Powers method that multiplies a number it itself
Sub Powers(ByRef d As Double)
    d = d * d
End Sub
```

```
// use this in code like so
Dim value As Double = 3
Powers(3) // value now equals 9
```

# ByVal (Keyword)

Used to pass a parameter by value. This is the default for parameters so this is completely optional.

## Keyword Summary

Name	ByVal
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">ByRef</a> , <a href="#">Function</a> , <a href="#">Sub</a>

## Syntax

```
ByValue parameter
```

## Notes

Parameters passed by value (the default) are treated as local variables inside the method, just like variables that are created using the Dim statement. This means that you can modify the values of the parameters themselves rather than first assigning the parameter to a local variable. For example, if you pass a value in the parameter "x", you can increment or decrement the value of x rather than assigning the value of x to a local variable that is created using Dim.

ByRef and ByVal are only applicable to simple data types (Integer, String, Double, etc.) All arrays and object types are reference types, effectively behaving as if ByRef was used with a simple data type.

## Break (Keyword)

When running in debug mode (from the IDE), this command stops code execution and displays the debugger. It has no effect in a built application.

### Keyword Summary

Name	Break
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">About Debugging Using the Debugger</a>

### Syntax

```
Break
```

### Notes

The Break command has the same effect as setting a breakpoint in the Code Editor.

A common use of the Break command to display the debugger when a specific condition is met.

### Sample Code

Display the debugger when the loop counter reaches 5:

```
For i As Integer = 1 To 10
  If i = 5 Then Break // Display debugger
Next
```

# Call (Keyword)

Allows you to ignore the return value from a function call.

## Keyword Summary

Name	Call
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Methods</a>

## Syntax

```
Call functionName
```

## Notes

Use the Call command only when you want to call a function without using the value it returns. This should not often be the case since the return value is often useful, but it does enable you to call the function without having to create a local variable to store its result.

## Sample Code

```
Call CalculateSomeValueIgnoringReturnValue
```

# Case (Keyword)

Identifies a specify value that is matched in a Select...Case statement.

## Keyword Summary

Name	Case
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Select...Case</a> <a href="#">Controlling Code Flow</a>

## Sample Code

A Select...Case command:

```
Dim i As Integer = 5
Dim output As Text

Select Case i
Case Is < 4
    output = "Less than four"
Case 5
    output = "Five"
Else
    output = "Greater than five"
End Select
```

## Catch (Keyword)

Starts the Catch section of a [Try...Catch](#) code block.

### Keyword Summary

Name	Catch
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Try...Catch</a> <a href="#">Exception Handling</a>

### Sample Code

A Try...Catch command:

```
Try
  Dim a(5) As Integer
  a(6) = 45 ' Raises an OutOfBoundsException
Catch e As OutOfBoundException
  ' The above exception is caught here for you to handle
  ErrorLabel.Text = "Exceeded size of array"
End Try
```

# Const (Keyword)

Declares a value as a local constant.

## Keyword Summary

Name	Const
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Dim</a> <a href="#">Variables and Constants</a>

## Syntax

```
Const constantName [As datatype] = value
```

## Notes

Constants are assigned once and can never change. Local constants are only available within the method where there are declared.

You can optionally specify a data type for the constant value.

You can use Const anywhere within the method and can also declare constants within code blocks (such as For...Next, If...Then, etc.). Constants declared within code blocks are not accessible outside the code block.

You can also assign constants using expressions that consist of other constants or literal values.

### Project Item Constants

You can also create constants in a window, class, or module, as described in *User Guide Book 1: Fundamentals*. Constants that belong to windows, web pages, views or classes can be public (accessible throughout the application), protected (accessible only within the object that owns the constant and its subclasses), or private (accessible only within the object that owns it). Modules can also have global constants, accessible throughout the application.

## Sample Code

Some ways to use constants:

```
Const kAnswer As Integer = 42
```

```
Const kPi As Double = 3.14159265358979323846264338327950
```

```
' Constants declared from expressions
```

```
Const kNumber = 10 * 45 ' kNumber = 450
```

```
Const kName = "Bob " + "Roberts" ' kName = "Bob Roberts"
```

```
' Assign constant to a variable
```

```
Dim name As Text
```

```
name = kName
```

```
' Assign constant to a property
```

```
Label1.Text = kName
```

# Continue (Keyword)

Used to to continue execution with the next iteration of a loop, skipping any other code.

## Keyword Summary

Name	Continue
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Do...Loop</a> <a href="#">For...Next</a> <a href="#">For Each...Next</a> <a href="#">While...Wend</a> <a href="#">Repeating Code</a>

## Syntax

<code>Continue</code>
<code>Continue [For   While   Do]</code>
<code>Continue <i>forLoopCounter</i></code>

## Notes

The Continue statement enables you to jump to the end of a loop and continue execution without executing the lines of code between the Continue statement and the end of the loop. If there is ambiguity concerning which loop you mean, you can use the second or third syntaxes to clarify the situation. For example, if you have nested For loops and want to jump from a line in the innermost loop to the outermost loop, you can use the last syntax to identify the loop variable that controls the outermost loop.

## Sample Code

```
// Count the sum from 1 to 100, skipping the number 2 and 31
Dim sum As Integer

For i As Integer = 1 To 100
    If i = 2 Or i = 31 Then Continue

    sum = sum + i
Next
```

# CurrentMethodName (Method)

The name of the currently running method or event.

## Method Summary

Name	CurrentMethodName
Type	Method
Project Types	All
Platforms	All
Related	<a href="#">DebugLog</a>

## Syntax

```
result = CurrentMethodName
```

## Notes

You can use this for logging purposes to identify methods that were called.

# Declare (Keyword)

Used to call OS API methods.

## Keyword Summary

Name	Declare
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">CFStringRef</a> <a href="#">CString</a> <a href="#">iOSBlock</a> <a href="#">MemoryBlock</a> <a href="#">Ptr</a> <a href="#">WString</a>
Webinars	<a href="#">Simple iOS Declares</a>

## Syntax

```
[Soft] Declare Sub|Function name Lib libraryName [Alias aliasName] [Selector selectorName]  
([Parameters]) [As returnType]
```

<i>name</i>	The method or function name that will be used in your Xojo code. It is typically the name of the OS API.
<i>libraryName</i>	The Library (DLL, shared library or framework) containing <i>name</i> . This must be in quotes.
<i>aliasName</i>	(Optional) If the API call ( <i>name</i> ) has the same name as a built-in method (or you want a different name), declare the call with a different <i>name</i> and use the alias to refer to the actual API method name. The <i>aliasName</i> must be in quotes.
<i>selectorName</i>	(OS X and iOS) Used to specify the parameters to uniquely identify a Cocoa method. The parameters are separated by colons (no spaces) and must end in a colon. This is mutually exclusive with the Alias. This must be in quotes.
<i>parameters</i>	The parameters to pass to the <i>name/selectorName</i> . You can pass Nil to a parameter of type Ptr.
<i>returnType</i>	(Optional) The return type for the function.

## Notes



Incorrect Declares, such as Declares that use an incorrect library, method, parameter or selector will crash your Xojo app. You will need to review system logs (such as Console on OS X) to see

 what the OS reports as the cause of the crash.

For iOS, you can Declare into Cocoa Touch APIs.

Sub/Function names are always case-sensitive. For example, SetLocalTime works on Win32, but SetlocalTime does not.

You can use constants in place of *libraryName*.

Declare can be used to call external functions that return structure values.

You can pass Nil to a parameter of type Ptr, but passing Nil is not the same as passing a Nil MemoryBlock. The former is a constant and the latter requires a conversion at run-time. If the MemoryBlock is Nil, the conversion will raise a NilObjectException.

## Cocoa

The libraryname is automatically expanded to the full path of the library. For example, CoreMIDI would expand to "/System/Library/Frameworks/CoreMIDI.framework/CoreMIDI".

## Soft Declares

Soft declare statements are resolved at run-time only when your application tries to use the Declare.

In most cases, you should use a Soft declare in order to prevent your app from launching due to missing libraries or library functions.

For example rather than creating a declare to a specific version of LibC (on Linux) like this:

```
Declare Function getpid Lib "libc-2.3.2.so" () As Integer
```

You can instead use a soft declare and allow it to resolve to the appropriate version at run-time:

```
Soft Declare Function getpid Lib "libc" () As Integer
```

 By default, all Cocoa declares are "soft" even if you do not specify the Soft keyword.

## Alias Names

You can use the Alias keyword when you want to refer to the method by a different name:

```
Declare Function GetProcessID Lib "libc" Alias "getpid" As Integer
```

## Sample Code

Commonly used Declares:

```
// Gets a reference to an iOS class
Declare Function NSClassFromString Lib "Foundation" (classname As CFStringRef) As Ptr
// Usage: Dim NSClass As Ptr = NSClassFromString("NSClassName")
```

```
// Gets an instance of an iOS class
Declare Function alloc Lib "Foundation" Selector "alloc" (classRef As Ptr) As Ptr
// Usage: Dim instance As Ptr = alloc(NSClass)</code>
```

This code can be used to set the focus on a control:

```
Declare Sub becomeFirstResponder Lib "Foundation" Selector "becomeFirstResponder" (ref
As Ptr)
becomeFirstResponder(TextField1.Handle)
```

This code clears the focus:

```
Sub ClearFocus(extends c As iOSControl)
  Declare Sub resignFirstResponder Lib "Foundation" Selector "resignFirstResponder"
(ref As Ptr)
  resignFirstResponder(c.Handle)
End Sub</code>
```

Open a URL in the default app that can handle it:

```
Function ShowURL(url As Text) As Boolean
  // NSString* launchUrl = @"http://www.xojo.com/";
  // [[UIApplication sharedApplication] openURL:[NSURL URLWithString: launchUrl]];

  Declare Function NSClassFromString Lib "Foundation" (name As CFStringRef) As Ptr
  Declare Function sharedApplication Lib "UIKit" Selector "sharedApplication" (obj As
Ptr) As Ptr
  Dim sharedApp As Ptr = sharedApplication(NSClassFromString("UIApplication"))

  //
https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSURL\_Class/#//apple\_ref/occ/clm/NSURL/URLWithString:
  Declare Function URLWithString Lib "Foundation" Selector "URLWithString:" ( id As
Ptr, urlString As CFStringRef ) As Ptr
  Dim nsURL As Ptr = URLWithString(NSClassFromString("NSURL"), url)

  //
https://developer.apple.com/Library/ios/documentation/UIKit/Reference/UIApplication\_Class/index.html#//apple\_ref/occ/instm/UIApplication/openURL:
  Declare Function openURL Lib "UIKit" Selector "openURL:" (id As Ptr, nsurl As Ptr) As
Boolean
  Return openURL(sharedApp, nsURL)
End Function</code>
```

Check the scale for the main iOS screen:

```
Function mainScreenScale() As Double
  Declare function NSClassFromString Lib "Foundation" (aClassName As CFStringRef) As
```

```
Ptr
  #If Target32Bit Then
    Declare Function scale Lib "Foundation" Selector "scale" (classRef As Ptr) As
Single
  #Else
    Declare Function scale Lib "Foundation" Selector "scale" (classRef As Ptr) As
Double
  #Endif
  Declare Function mainScreen Lib "Foundation" Selector "mainScreen" (classRef As Ptr)
As Ptr

  Return scale(mainScreen(NSClassFromString("UIScreen")))
End Function</code>
```

Disable the Idle Timer to prevent the device from sleeping:

```
Function SetIdleTimerDisabled(disabled As Boolean)
  Declare Sub setIdleTimerDisabled Lib "UIKit" Selector "setIdleTimerDisabled:" (obj_id
As Ptr, disabled As Boolean)
  Declare function sharedApplication Lib "UIKit" Selector "sharedApplication" (clsRef
As Ptr) As Ptr
  Declare function NSClassFromString Lib "Foundation" (clsName As CFStringRef) As Ptr
  setIdleTimerDisabled(sharedApplication(NSClassFromString("UIApplication")), disabled)
End Function
```

# iOS Declare Example Projects

Also refer to the [Creating Simple iOS Declares](#) webinar.

All iOS Declare examples are located here: [Examples/iOS/Declares](#)

## AlertSheet

Demonstrates declares to show an iOS Alert Sheet.

## Base64Encoding

Demonstrates how to Base64Encode Text or a MemoryBlock and to decode back to Text.

## IconBadgeNumber

Demonstrates how to add a badge number to your app icon.

## iOSAlerts

Demonstrates how to show all the different types of iOS alerts, including text, password, login and more.

## SetFocus

Demonstrates how to set focus in a specific control.

## ShowURL

Demonstrates how to use the iOS URL handling to open URLs in Safari, Maps, Mail and Phone.

## Speak

Demonstrates how to use Text to Speech.

## TableSelectAndScroll

Demonstrates how to select a row in an [iOSTable](#) and scroll to it if it is not visible.

## UIButtonDeclares

Demonstrates how to access additional settings for Buttons.

## UIDatePicker

Demonstrates how to add and use the UIDatePicker control.

## UIDevice

Displays device information.

## UIScrollView

Demonstrates a scrollable view.

## OS X Declare Samples

### Current Logged-In User

```
Declare Function NSFullUserName Lib "Foundation" As CFStringRef</code>
```

### Center a Window

```
Declare Sub center Lib "Cocoa" Selector "center" (windowRef As Integer)
center(Self.Handle) // The Handle property of the Window</code>
```

### Miniaturize a Window to the Dock

```
Declare Sub miniaturize Lib "Cocoa" Selector "miniaturize:" (windowRef As Integer, id
As Ptr)
miniaturize(Self.Handle, Nil) // The Handle property of the Window
```

### Move a Window with Animation

```
Dim newRect As NSRect
```

```
// note that coordinates are upside down from what you are used to with Carbon
// 0,0 is at the bottom left
newRect.x = 10
newRect.y = Screen(0).Height - 30
newrect.width = Self.Width
newrect.height = Self.Height + 22
```

```
Declare Sub setFrameDisplayAnimate Lib "Cocoa" Selector "setFrame:display:animate:" _
(windowRef As Integer, rect As NSRect, display As Integer, animate As Integer)
setFrameDisplayAnimate(Self.Handle, newRect, 1, 1)
```

### Display Standard Cocoa About Window

```
Declare Function NSClassFromString Lib "Cocoa" (aClassName As CFStringRef) As Ptr
Declare Function sharedApplication Lib "Cocoa" Selector "sharedApplication" (receiver
As Ptr) As Ptr
```

```
Dim sA As Ptr = NSClassFromString("NSApplication")
sA = sharedApplication(sA)
```

```
Declare Sub OrderFrontStandardAboutPanel Lib "Cocoa" Selector
```

```
"orderFrontStandardAboutPanel:" _  
    (receiver As Ptr, ID As Ptr)
```

```
OrderFrontStandardAboutPanel(sA, Nil)
```

## Display a TextField with Rounded Corners

```
Declare Sub setBezelStyle Lib "Cocoa" Selector "setBezelStyle:" (handle As Integer,  
value As Integer)  
setBezelStyle(Me.Handle, 1)
```

## Play a Cocoa Notification Sound

```
Const CocoaLib = "Cocoa.framework"  
Soft Declare Function NSClassFromString Lib CocoaLib (aClassName As CFStringRef) As Ptr  
Soft Declare Function SoundNamed Lib CocoaLib Selector "soundNamed:" (ClsPtr As Ptr,  
name As CFStringRef) As Ptr  
Soft Declare Function Play Lib CocoaLib Selector "play" (instPtr As Ptr) As Boolean  
  
Dim NSSoundClassPtr As Ptr = NSClassFromString("NSSound")  
Dim notificationSound As Ptr = SoundNamed(NSSoundClassPtr, "Frog")  
Call Play(notificationSound)
```

# Windows Declare Samples

## Flash App Indicator in Task Bar

```
Declare Function FlashWindow Lib "User32" (HWND As Integer, invert As Boolean) As Boolean
Call FlashWindow(MyWindow.Handle, True) // flash once
```

# Dim

Declares a local variable for a specific data type.

## Keyword Summary

Name	Dim
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Arrays</a> <a href="#">ReDim</a> <a href="#">Variables and Constants</a>

## Syntax

```
Dim variableName As dataType [= initialValue]
```

```
Dim variableName As New dataType
```

```
Dim variableName, variableNameN As dataType [= initialValue]
```

Dim can also declare [arrays](#).

## Notes

Variable names can be any length. The first character of a variable must begin with a letter and can contain alphanumeric characters [A-Z, a-z, 0-9]. Subsequent characters can be any non-space Unicode character. All variable names are case-insensitive, so the names "FirstName" and "firstName" would refer to the same variable (the compiler may not currently enforce this for all Unicode characters). The underscore ( `_` ) cannot be used as the first character of a variable. For specific details, refer to [Unicode Standard Annex 31](#).

You can declare variables anywhere within the method. Some people prefer to declare all variables at the top of the method. Others prefer to declare variables near where they will be used. All variables are local to the specific method in which they are defined and cannot be accessed outside the method.

Variables declared within a code block (If...Then, For...Next, etc.) are unavailable (out of scope) outside of the code block.

## Sample Code

```
Dim a As Integer
Dim value As Integer = 42
```

```
Dim percent As Double = 0.50
Dim t As Text = "Hello"
Dim c As Color = &cFF4B51

Dim x, y, z As Integer = 10 // All variables are assigned the value 10

// Instantiate an object in a single statement
Dim d As New Dictionary

// Initialize a variable using a constant
Const kValue = 5
Dim value As Integer = kValue

// Initialize a variable using an enumeration (SecurityLevels)
Dim testEnum As SecurityLevels = SecurityLevels.Maximum
```

# Do (Keyword)

Indicates the start of a Do...Loop loop.

## Keyword Summary

Name	Do
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Do...Loop</a> <a href="#">Repeating Code</a>

## Sample Code

Loop until the counter reaches 100:

```
Dim x As Integer
Do
    x = x + 1
Loop Until x >= 100
```

# Do...Loop (Keyword)

Repeatedly executes a series of statements until the specified condition is True.

## Keyword Summary

Name	Do...Loop
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">For...Next</a> <a href="#">For Each...Next</a> <a href="#">While...Wend</a> <a href="#">Repeating Code</a>

## Syntax

```

Do [Until condition]
    statements
[Continue]
[Exit]
statements
Loop [Until condition]

```

<i>condition</i>	Any valid boolean expression. When the expression evaluates to True, the loop exits. When the expression is False, the statements in the loop are executed.
Continue	The Continue keyword causes the execution of code to skip directly to the Loop statement, skipping over any lines in between.
Exit	The Exit keyword immediately exits the loop so that the statement immediately following the Loop statement is executed.

## Notes

The Do...Loop statement continues to execute the statements within it until an Until condition evaluates to True or an Exit statement is reached.

The Until condition parts of the loop are optional, but if you do not include it you *have* to include an Exit statement

somewhere within the loop or you will have an **infinite loop**, causing your app to appear to get stuck or hang.

Variables declared within the Do...Loop are local to the loop and go out of scope when the loop ends.

## Sample Code

```
Dim x As integer
Do
    x = x + 1
Loop Until x = 100
// x = 100
```

```
x = 0
Do
    x = x + 1
    If x >= 100 Then Exit
Loop
// x = 100</code>
```

Using Exit:

```
Const kAttempts = 10
Dim attempt As Integer = 1
Dim output As Text

Do Until attempt > kAttempts
    Dim randomValue As Integer = Math.RandomInt(1, 10)
    If randomValue > 5 Then
        output = "Found a random value above 5 after " + attempt.ToText + " iterations."
        Exit
    End If
    attempt = attempt + 1
Loop

If attempt > kAttempts Then
    output = "Found NO random value above 5 after " + kAttempts.ToText + " iterations."
End If</code>
```

Using Continue:

```
Const kAttempts = 100
Dim matchCount As Integer
Dim attempt As Integer

Do
    attempt = attempt + 1
    Dim randomValue As Integer = Math.RandomInt(1, 10)
    If randomValue <= 5 Then
```

```
    Continue
End If
    matchCount = matchCount + 1
Loop Until attempt >= kAttempts

Dim output As Text
output = "Found " + matchCount.ToText + " random values above 5 within " +
kAttempts.ToText + " iterations."
```

## DownTo (Keyword)

Separates the beginning and ending values of a For...Next loop counter that counts down from the start value to the end value.

### Keyword Summary

Name	DownTo
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">For...Next</a>

### Sample Code

Count down to 1:

```
For i As Integer = 10 DownTo 1
    ...
Next
```

## Each (Keyword)

Specifies a value from the iterable item in a For Each...Next statement.

### Keyword Summary

Name	Each
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">For Each...Next</a>

### Sample Code

Loop through an array:

```
For Each i As Integer In values()  
    ...  
Next
```

## Else (Keyword)

Indicates the Else part of an If...Then...Else command or a non-matching section of a Select...Case command.

### Keyword Summary

Name	Else
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">If...Then...Else</a> <a href="#">Select...Case</a>

### Sample Code

Used in an If command:

```
If True Then
  // Your code
Else
  // Your code
End If
```

## Elseif (Keyword)

Indicates an Elseif part of an If...Then...Else command.

### Keyword Summary

Name	Elseif
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">If...Then...Else</a>

### Sample Code

```
If True Then
  // Your code
Elseif False Then
  // Your code
End If
```

## End (Keyword)

Indicates the end of a variety of code blocks.

## Keyword Summary

Name	End
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">If...Then...Else</a> <a href="#">Try...Catch</a>

## Syntax

```
End [If | Function | Select | Sub | Try]
```

## Notes

The type of block is an optional part of the command but is usually included for clarity.

## Sample Code

```
If True Then  
    // Your code  
End If
```

# Exit (Keyword)

Exits the current loop or method.

## Keyword Summary

Name	Exit
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Do...Loop</a> <a href="#">For...Next</a> <a href="#">For Each...Next</a> , <a href="#">While...Wend</a> <a href="#">Function</a> <a href="#">Sub</a>

## Syntax

Exit [For   Do   While]
Exit [forLoopCounter]
Exit [Sub   Function]

## Notes

If you use Exit without any optional keywords, it exists the current loop. If the Exit is not within a loop, it exits the method.

If you use Exit with the Do keyword, the Do loop it is in, even if it is also inside another type of loop, is exited.

If you use Exit with the While keyword, the While loop it is in, even if it is also inside another type of loop, is exited.

If you use Exit with the For keyword without passing a parameter, it exits the innermost For loop it is in, even if it is also inside another loop.

If you have nested For statements, you can pass the For keyword the variable that controls the loop you want to exit. For example, if you are testing all the elements of a two-dimensional array with nested For statements, you can use the loop variable for the outermost loop to exit at that point.

```
For i = 0 to 255
  For j = 0 to 255
    If myArray(i, j) = 23 then
```

```
    Exit For i
  End if
Next
Next
```

If you use Exit with the Sub or Function keywords, then it will exit the entire method or function as if you used Return. Exit Function cannot specify a return value, so if you use it to exit a function, the function will return the default value for the data type of the function.

## Sample Code

```
// Searches a ListBox for a specific value in the first column and
// when found, selects it and exits the loop
Dim search As Text = "text to find"

For i As Integer = 0 To ListBox1.ListCount - 1
  If ListBox.Cell(i, 0) = search Then
    ListBox1.Selected(i) = True
    Exit
  End If
Next
```

# Extends

Used in a module method declaration to indicate that the method is to be called using the dot operator ("."), as if it were an object method of the first parameter.

## Keyword Summary

Name	Extends
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Methods</a>

## Syntax

<i>Extends parameter As dataType</i>	
<i>parameter</i>	The name of the first parameter. If parameter is Nil, the Extends method is not executed, and a <a href="#">NilObjectException</a> is raised.
<i>dataType</i>	The data type (or class) for which the method can be called.

## Notes

The Extends keyword allows you to call a user-defined method as if it were part of a class. You use the Extends keyword only for the first parameter in the method declaration. Extends indicates that this parameter is to be used on the left side of the dot operator. The remaining parameters in the declaration, if any, are used normally.

Extension methods can be used with classes or other data types. For example, you can define methods that extend String, Color, and Integer data types. You can also extend arrays.

Methods declared in this way are sometimes called "class extension methods" even though they are not actually part of a class. You can use Extends only for methods in a module. Extends cannot be used to override another method. Extension methods are not virtual, since they are not part of a class.

Extension methods are an example of **syntactic sugar**. There is no difference in capability between an extension method and a normal module method; each simply specifies a particular calling syntax. (Many built-in methods are provided in both versions, so that either calling syntax will work.) Extension methods can make calling code more regular and readable, and if used consistently, can make it easier to remember which syntax to use. On the other hand, extension methods can make the dependency of a class on a module less obvious to a programmer.

---

## Examples

Extends can allow more readable and concise calling code. This example extends the Text data type with a function that determines whether one string contains another. In a module, create this function:

```
Function Contains(Extends t As Text, contained As Text) As Boolean
    Return t.InStr(Contained) >= 0
End Function
```

The Extends keyword indicates that the first parameter does not appear in the parameter list of the actual method call. Instead, it is used on the left side of the dot operator. The remaining parameter is used normally:

```
Dim t1 As Text = "Hello World"
Dim t2 As Text = "world"
If t1.Contains(t2) Then
    // Yes
End If
```

Without Extends, the method call would be functionally equivalent, but less clear in meaning:

```
If Contains(t1, t2) Then ... // what does each value mean?
```

# False

Assigns or tests the boolean value False. A comparison of two values that are not equal results in False.

## Keyword Summary

Name	True
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">And</a> <a href="#">Not</a> <a href="#">Or</a> <a href="#">Xor</a> <a href="#">True</a>

## Syntax

```
expression = False
```

## Sample Code

```
Dim b As Boolean  
b = True  
b = False
```

# Finally

Starts the Finally section of a Try...Catch code block.

## Keyword Summary

Name	Finally
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Try...Catch</a>

## Sample Code

```
Try
  Dim a(5) As Integer
  a(6) = 45 // Raises an OutOfBoundsException
Catch e As OutOfBoundException
  // The above exception is caught here for you to handle
  ErrorLabel.Text = "Exceeded size of array"
Finally
  // This code always runs
  a(0) = 100
End Try
```

# For

Starts a For...Next or For Each...Next loop.

## Keyword Summary

Name	For
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">For...Next</a> <a href="#">For Each...Next</a>

## Sample Code

```
For i As Integer = 1 To 10
    ...
Next
```

# For...Next

Loops over a collection of statements a specific number of times.

## Keyword Summary

Name	For...Next
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Do...Loop</a> <a href="#">For Each...Next</a> <a href="#">While...Wend</a> <a href="#">Repeating Code</a>

## Syntax

```

For counterValue [As datatype] = startValue To | DownTo endValue [Step stepValue]
    [Statements]
    [Continue [For]]
    [Exit [For]]
    [Statements]
Next [counterValue]

```

<i>counterValue</i>	A variable that is incremented or decremented every time through the loop. It can be an Integer, Single or Double. By default the <i>counterValue</i> is changed by 1 at the end of each loop (when the Next statement is reached or a Continue is invoked within the loop).
datatype	Specifies an optional datatype for the <i>counterValue</i> . This declares the variable for use only during the loop. The value goes out of scope when the loop ends.
<i>startValue</i>	The initial value of <i>counterValue</i> .
To   DownTo	Indicates whether the <i>counterValue</i> is incremented (To) or decremented (DownTo) each time through the loop.

<i>endValue</i>	The final value of <i>counterValue</i> . The loop ends if <i>counterValue</i> reaches a value beyond <i>endValue</i> at the time the Next or Continue is reached. This means that the loop ends once <i>counterValue</i> > <i>endValue</i> if <i>counterValue</i> is being incremented; the loop ends once <i>counterValue</i> < <i>endValue</i> if <i>counterValue</i> is being decremented. The <i>endValue</i> is re-evaluated in every loop iteration, so avoid invoking functions that return a value. If the function has a time-consuming operation, this could significantly delay the processing of your loop. Instead save the function result in its own variable and use that for <i>endValue</i> .
<i>stepValue</i>	The optional amount to increment or decrement <i>counterValue</i> each time through the loop. This defaults to 1. Using To indicates <i>counterValue</i> will be incremented; DownTo indicates <i>counterValue</i> will be decremented. If <i>stepValue</i> is negative, it causes <i>counterValue</i> to decrement, providing the same functionality as the DownTo keyword.
Continue	The Continue keyword causes the execution of code to skip directly to the Next statement, skipping over any lines in between.
Exit	The Exit keyword immediately exits the loop so that the statement immediately following the Next statement is executed.

## Notes

For...Next statements can be nested, but each For...Next statement must have its own counter variable.

When a For...Next loop is running, it blocks the user interface for your app. This prevents the user from being able to interact with the UI. For long-running tasks, use Threading to run the task in the background and allow the UI to remain responsive.

### Declaring the Counter Variable with the For Loop

The counter variable in a For statement can be declared inside the For statement rather than externally, as a local variable.

```
Dim i As Integer
For i = 0 To 10
    ...
Next
```

```
' Instead can be written as
For i As Integer = 0 To 10
    ...
Next
```

### Count Backwards

There are two ways you can write code to count from 5 down to 1:

```
For loopIndex As Integer = 5 DownTo 1
    ...
```

Next

```
For loopIndex As Integer = 5 To 1 Step -1
```

```
...
```

```
Next
```

### Counting By Steps

In addition to using -1 as a Step value to count backward, you can also use it to change counter by a different value in every loop iteration. For example, to count every other number:

### Exit and Continue

The Exit command is used to exit the For loop before it reaches its end counter. The Continue command is used to jump back to the start of the For loop skipping any subsequent code in the loop.

### Using Decimal Values as a Loop Counter

The following code shows the values 1.5, 2.5 and 3.5:

```
For dblValue As Double = 1.5 to 4
    MsgBox("Now at " + dblValue.ToText)
Next
```

### Counter Values After the Loop

If you declare the *counter* variable outside the loop, it retains its last value after the loop ends. If it finishes normally, i.e., it cycles through every element without encountering Exit, that value will be the end value plus 1 if you have not specified a Step value, or the end value plus Step if you have. If the loop exits early through Exit, the counter will retain the value it held at the point of exit.

```
Dim i As Integer
```

```
For i = 1 To 10
    // Some code
Next i
// i = 11
```

```
For i = 1 To 10 Step 2
    // Some code
Next i
// i = 12
```

```
For i = 10 DownTo 1
    // Some code
Next i
// i = 0
```

```
For i = 10 To 1 Step -2
    // Some code
```

```
Next i
// i = -1

Dim d As Double

For d = 1.0 To 10.0
    // Some code
Next d
// d = 11.0
```

```
For d = 1.0 To 10.0 Step 0.5
    // Some code
Next d
// d = 10.5
```

### Performance Tips

If start, end, or step are not constant values, they get evaluated every time the loop is executed, even if they evaluate to the same value for each iteration. Therefore, the loop:

```
For i As Integer = 0 To FontCount-1
    ...
Next
```

takes more time than:

```
Dim nFonts As Integer
nFonts = FontCount-1
For i = 0 To nFonts
    ...
Next
```

Be especially aware of this when the end value changes during the loop! Also, you can modify the counter variable inside the loop.

The example below shows both techniques. While it adds new elements to the array, but it will still iterate over all items:

```
Dim values() As Integer = Array(1, 5, 4, 8)
// The loop inserts 0 after values above 4 inside the values array:
For i As Integer = 0 To values.Ubound
    If values(i) > 4 Then
        values.Insert i+1, 100
        i = i + 1 // skip the next item because otherwise we'd add more values endlessly
    End If
Next i
```

### Troubleshooting

When looping through lists (such as ListBoxes or arrays) where you plan to remove some of the items, you should

loop from the end of the list to the beginning to avoid the countValue no longer matching the expected position in the new, smaller list.

## Sample Code

```
Dim days() As Text = Array("One", "Two", "Three", "Four", "Five")
Dim output As Text
For counter As Integer = 0 To days.UBound
    output = output + days(counter)
Next
// output = "OneTwoThreeFourFive"
```

```
Dim days() As Text = Array("One", "Two", "Three", "Four", "Five")
Dim output As Text
For counter As Integer = days.UBound DownTo 0
    output = output + days(counter)
Next
// output = "FiveFourThreeTwoOne"
```

Counting by Steps:

```
Dim output As Text
For i As Integer = 1 To 6 Step 2
    output = output + i.ToText + " "
Next
// output = "1 3 5 "
```

Using Exit:

```
Const kAttempts = 10
Dim output As Text

For attempt As Integer = 1 To kAttempts
    Dim randomValue As Integer = Math.RandomInt(1, 10)
    If randomValue > 5 Then
        output = "Found a random value above 5 after " + attempt.ToText + " iterations."
        Exit
    End If
Next
```

```
If attempt > kAttempts Then
    output = "Found NO random value above 5 after " + kAttempts.ToText + " iterations."
End If
```

Using Continue:

```
Const kAttempts = 100
```

```
Dim matchCount As Integer

For attempt As Integer = 1 To kAttempts
    Dim randomValue As Integer = Math.RandomInt(1, 10)
    If randomValue <= 5 Then
        Continue
    End If
    matchCount = matchCount + 1
Next

Dim output As Text
output = "Found " + matchCount.ToText + " random values above 5 within " +
kAttempts.ToText + " iterations."
```

# For Each...Next

Loops through the elements of a one-dimensional array.

## Keyword Summary

Name	For Each...Next
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Do...Loop</a> <a href="#">For...Next</a> <a href="#">While...Wend</a> <a href="#">Repeating Code</a>

## Syntax

```

For Each element [As datatype] In array
    [Statements]
    [Continue [For]]
    [Exit [For]]
    [Statements]
Next [counterValue]

```

<i>element</i>	<i>array</i> A variable of the same data type as <i>array</i> that refers to an element of <i>array</i> . The loop processes each value in <i>array</i> , but the specific order is not guaranteed.
<i>datatype</i>	Optional: The data type of the array element. It can be any one-dimensional array. If you declare the data type with the optional <i>As</i> clause, you do not have to do so with a <i>Dim</i> statement before the loop. The data type must match <i>array</i> 's data type.
<i>array</i>	A one-dimensional array whose data type must match <i>datatype</i> .
<a href="#">Continue</a>	The <i>Continue</i> keyword causes the execution of code to skip directly to the <i>Next</i> statement, skipping over any lines in between.
<a href="#">Exit</a>	The <i>Exit</i> keyword immediately exits the loop so that the statement immediately following the <i>Next</i> statement is executed.

## Notes

A For...Each loop does not guarantee that it will loop through the values in the array in index order. Do not make any assumptions of the traversal order as it is subject to change in the future.

As with other block statements, variables declared within the For...Each loop go out of scope when the loop finishes or exits.

## Sample Code

```
// Iterate through an array
Dim days() As Text = Array("One", "Two", "Three", "Four", "Five")
Dim output As Text
For Each d As Text In days
    output = output + d
Next
```

```
// Calculate the sum of values in an array
Dim values() As Double = Array(1.5, 5.5, 8.0, 45.0, 22.5)
Dim sum As Double
For Each d As Double In values
    sum = sum + d
Next
// sum = 82.5
```

Counting by Steps:

```
Dim output As Text
For i As Integer = 1 To 6 Step 2
    output = output + i.ToText + " "
Next
// output = "1 3 5 "
```

Using Exit:

```
Const kAttempts = 10
Dim attempt As Integer
Dim output As Text

For attempt = 1 To kAttempts
    Dim randomValue As Integer = Math.RandomInt(1, 10)
    If randomValue > 5 Then
        output = "Found a random value above 5 after " + attempt.ToText + " iterations."
        Exit
    End If
Next
```

---

```
If attempt > kAttempts Then
    output = "Found NO random value above 5 after " + kAttempts.ToText + " iterations."
End If
```

Using Continue:

```
Const kAttempts = 100
Dim matchCount As Integer

For attempt As Integer = 1 To kAttempts
    Dim randomValue As Integer = Math.RandomInt(1, 10)
    If randomValue <= 5 Then
        Continue
    End If
    matchCount = matchCount + 1
Next

Dim output As Text
output = "Found " + matchCount.ToText + " random values above 5 within " +
kAttempts.ToText + " iterations."
```

# Global

Identifies the global scope, making it possible to refer to items that might otherwise be ambiguous.

## Keyword Summary

Name	Global
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Using</a>

## Syntax

```
Global.identifier
```

## Notes

Global works similarly to Self, Me and Super and is most useful when working with namespaces. You use it to refer to a global identifier (such as a module or namespace) that might otherwise be inaccessible because they are hidden by local identifiers (i.e. methods or variables) in the current scope.

## Sample Code

Declare a local variable that

Access the classic Dictionary class within a method that has used Using to access Dictionary in Xojo.Core

```
Using Xojo.Core
Dim d As New Dictionary // Uses Xojo.Core.Dictionary
```

```
Dim d2 As New Global.Dictionary // Uses classic Dictionary class
```

This is a more advanced example that shows how a local variable can make a module inaccessible:

```
Module Utility
  Protected Version As Integer
End Module
```

```
Module AppStuff
  Protected Version As String
```

---

```
Protected Sub GetStuff()  
    // Get the value of Utility.Version and assign it to our local Version  
    Version = Str(Utility.Version)  
  
    // But now we have a local variable named utility, which shadows module Utility!  
    Dim utility As Color  
  
    // In order to reach the Utility.Version, we must explicitly back out to the global  
scope  
    Global.Utility.Version = Val(utility.Hue)  
  
    // The first assignment can also be rephrased like this:  
    Global.AppStuff.Version = Str(Global.Utility.Version)  
End Sub  
End Module
```

# If

Starts an If...Then...Else command.

## Keyword Summary

Name	If
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">If...Then...Else</a> <a href="#">If operator</a>

## Sample Code

```
If True Then  
    ' Your code  
End If
```

```
Dim i As Integer = 40  
If i > 40 Then  
    ' Your code  
End If
```

# If...Then...Else

Conditionally executes a group of statements depending on the value of boolean conditions.

## Keyword Summary

Name	If...Then...Else
Type	Command
Project Types	All
Platforms	All
Related	<a href="#">If</a> <a href="#">#If...#EndIf</a>

## Syntax

```

.....
If condition Then
.....
    statements
.....
[ElseIf condition Then]
.....
    statements
.....
[Else]
.....
    statements
.....
End [If]
.....
If condition Then statement [Else statement]
.....

```

<i>condition</i>	An expression that evaluates to True or False.
<i>statements</i>	One or more code statements that are executed.

## Notes

The statements after the If are run if the If condition evaluates to True.

The statements after the optional ElseIf are run if the ElseIf condition evaluates to True.

The statements after the optional Else are run if all the other conditions evaluated to False.

Only the first section of an If...Then...Else that evaluates to True is run. So if both an If and an ElseIf evaluates to

True, then only the statements in the If portion will run.

You can have a single-line If...Then...Else statement, but when doing so only a single statement is allowed for the If and Else portions.

Variables declared within sections of an If...Then...Else statement go out of scope when leaving the section.

## Sample Code

```
Dim i As Integer = 6
```

```
If i > 5 Then  
    i = i * 10  
Else  
    i = i - 1  
End If  
// i = 60
```

```
If i < 5 Then  
    i = i * 10  
ElseIf i = 6 Then  
    i = 100  
Else  
    i = i - 1  
End If  
// i = 100
```

```
Dim t As Text = "Hello, World!"  
If t.BeginsWith("Hello") Then  
    Label1.Text = t  
Else  
    Label1.Text = "Goodbye!"  
End If
```

```
i = 100  
If i >= 10 And i <= 100 Then  
    Label1.Text = "In range"  
Else  
    Label1.Text = "Out of range"  
End If
```

```
If i < 10 Or i > 100 Then  
    Label1.Text = "Out of range"  
Else  
    Label1.Text = "in range"  
End If
```

# In

Specifies the array in a For Each...Next statement.

## Keyword Summary

Name	In
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">For Each...Next</a>

## Sample Code

```
For Each i As Integer In values()  
    ...  
Next
```

# Lib

Specifies the library to use with a Declare statement.

## Keyword Summary

Name	Lib
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Declare</a>

## Sample Code

```
Declare Sub becomeFirstResponder Lib "Foundation.Framework" Selector  
"becomeFirstResponder" (obj_id As Ptr)
```

```
becomeFirstResponder(TextField1.Handle)
```

---

# Line Continuation

The underscore character, "\_", is used as the line continuation character to indicate that a line of code is being continued on the next line of text in the Code Editor.

## Notes

Use the underscore character as the last character in a line of text in the Code Editor to indicate that you are continuing the line of code on the next line in the Code Editor. The compiler will then treat the second line as a continuation of the first line. When you use the \_ keyword, the Code Editor indents the next line automatically, as is shown:

```
Dim saveChangesDialog as New MessageDialog
SaveChangesDialog.Explanation = "Hint: If you don't save, you will " _
    + "lose your work since your last Save."
```

If you need to break up a text string into two lines in the Code Editor, place the \_ keyword outside the quoted string, as in the example above.

To enter the \_ character and move to the next line automatically, press Ctrl+Enter on Windows and Linux; on Macintosh, press Option-Return.

# Loop

Indicates the end of a Do...Loop loop.

## Keyword Summary

Name	Loop
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Do...Loop</a>

## Sample Code

```
Dim x As Integer
Do
    x = x + 1
Loop Until x >= 100
```

# Me

Refers to the control that owns (or contains) the current event handler.

## Keyword Summary

Name	Me
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Self</a>

## Syntax

```
Me.MemberName
```

## Notes

In an event handler of a class (usually a control) that was added to a Window, WebPage, iOSView, ContainerControl or WebContainer, you use the Me prefix to refer to its own members rather than the members of the parent Window, WebPage, iOSView, ContainerControl or WebContainer.

For example, in the Open event handler of a Label, if you wanted to change the Label Text property, you would write this code:

```
Me.Text = "Hello"
```

If, in the Label's Open event handler, you tried to access the Text property without the Me prefix, your code would actually instead be referring to a (non-existent) Text property of its container (the Window, WebPage, etc.).

If you use **Me** outside of an event handler of a control in this manner, **Me** behaves as if you used **Self**. For clarity, you should limit your use of **Me** to only within event handlers used on containers as described above.

## Sample Code

```
// Code in the Open event handler of a Label that sets its Text property
Event Open()
    Me.Text = "Hello"
End Event
```

# Next

Defines the end of For..Next or For Each...Next loops.

## Keyword Summary

Name	Next
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">For...Next</a> <a href="#">For Each...Next</a>

## Sample Code

```
For i As Integer = 1 To 10
    ...
Next
```

# Nil

Indicates that an object is Nil (has no value).

## Keyword Summary

Name	Nil
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">ls</a>

## Syntax

```
expression = Nil
```

## Notes

Nil means no value. When an object variable is first declared, it is automatically initialized to Nil. Some functions return a Nil value under certain circumstances.

Nil is a constant of its own datatype, not an Object. It can convert itself to any object class, any array type, or any of the pointer types (Ptr, WindowPtr, CString, PString, WString, and CFStringRef).

You can pass Nil for any of the string parameter types above (CString, PString, WString), but not for String or Text. Passing Nil is not the same as passing the empty string, "". Passing Nil means you want to pass a null pointer. Passing "" means you want to pass a non-null pointer to the representation of "" for the declared format.

If you try to access an object that has a Nil value, a NilObjectException error is raised. If you don't handle the exception, your app will terminate. If you are testing the application in the IDE, you can turn on Project → Break on Exceptions to display the Debugger at the line of code that caused the runtime exception (even if the exception is handled).

An unhandled exception will display a generic error message before the application terminates. You can use the generic UnhandledException event handler in the App object in order to attempt to recover for otherwise unhandled exceptions.

You should use Try..Catch to handle possible exceptions in your code so that your app does not terminate unexpectedly.

## Sample Code

```
Dim f As New FolderItem("test.txt")
If f <> Nil Then
    // use the FolderItem
End If
```

# Optional

Used in a method parameter to indicate that the parameter is optional and may be excluded from the method call.

## Keyword Summary

Name	Optional
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Methods</a>

## Syntax

```
Optional parameterName As dataType
```

## Notes

Optional parameters must be defined at the end of the parameter list, after any required parameters. If the method call provides an argument for any one of a succession of optional parameters, it must provide arguments for all preceding optional parameters as well. Comma-separated gaps in the argument list are not allowed.

Optional does not provide a default value, so the default value becomes the default for the type or Nil.

## Sample Code

```
Sub MyMethod(a As Integer, b As Integer, Optional c As Integer)
```

```
// You can call the above method with or without the last parameter
```

```
MyMethod(5, 42, 10)
```

```
MyMethod(5, 42) // c will default to 0
```

```
// If you want to specify a default value, do that instead of using Optional
```

```
Sub MyMethod(a As Integer, b As Integer, c As Integer = 10)
```

# ParamArray

Used in a method parameter to indicate that an arbitrary number of parameters can be passed.

## Keyword Summary

Name	ParamArray
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">ByVal</a>

## Syntax

```
ParamArray parameterName As dataType
```

## Notes

This keyword allows you to have a method call that can take an arbitrary number of parameters. The parameters are available to the method in an array.

## Sample Code

```
Function AddNumbers(ParamArray nums As Integer) As Integer
    Dim i, total As Integer
    For Each i In nums
        total = total + i
    Next
    Return Total
End Function
```

```
// This function can now be called like this:
Dim sum As Integer
sum = AddNumbers(1, 2, 3, 4, 5)
// sum = 15
```

# Raise

Used to raise an exception to the next level in the calling chain.

## Keyword Summary

Name	Raise
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Try...Catch</a> <a href="#">Exception Handling</a>

## Syntax

```
Raise exceptionInstance
```

## Notes

The exceptionInstance must be an object that is a subclass of RuntimeException.

You can create your own subclasses of RuntimeException and use Raise to raise them for error conditions.

## Sample Code

Raise New RuntimeException

```
Dim e As New MyRuntimeException // subclass of RuntimeException  
e.Message = "My error message."  
Raise e
```

# RaiseEvent

Calls an event definition on a class.

## Keyword Summary

Name	RaiseEvent
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Events</a>

## Syntax

```
RaiseEvent eventDefinitionName[(parameters)]
```

## Notes

If the event definition requires parameters, they are included after eventDefinitionName.

This statement is optional as you can always call an event definition by its name. It is useful for situations where you have an event definition with the same name as a method on the class.

The only place you can use RaiseEvent (or call an Event) is on the class that contains the event definition. In particular, you cannot raise an event from an subclass, an instance of a control on a layout or from outside the class. If you need to be able to call an event, you will have to create a companion method on the initial class that raises the event.

## Sample Code

```
RaiseEvent MyEvent // MyEvent is an Event Definition on the class
```

# ReDim

Resizes an array previously declared with [Dim](#).

## Keyword Summary

Name	ReDim
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Arrays</a> <a href="#">Dim</a>

## Syntax

```
ReDim arrayName(upperBound)
```

```
ReDim arrayName(upperBoundDimension1, ...upperBoundDimensionN)
```

## Notes

The ReDim command can be used to increase or decrease the number of elements in a previously declared array.

When you ReDim an array to an unbounded size (using -1 as the upperBound), then all its elements are removed. It is faster to use ReDim in this manner to clear any array rather than looping through the array and calling the Remove method on each element.

When you increase the size of an array, any existing values are retained.

When you decrease the size of an array, existing values are retained if they are still within the new array bounds.

## Sample Code

```
Dim names() As Text // Create empty array
```

```
ReDim names(10) // Set its upper bound to 10
```

```
ReDim names((names.UBound + 10) // Add 10 additional array elements
```

```
ReDim names(-1) // Clear the array, removing all elements
```

# Rem

Used to add comments (remarks) to your code. You can also use the symbols // and '.

## Keyword Summary

Name	Rem
Type	Keyword
Project Types	All
Platforms	All
Related	

## Notes

All comments are ignored by the compiler and are not included in your built apps.

Comments can appear on separate lines or on the same line as code, provided they are to the right of the code.

## Sample Code

```
Rem This is a comment
// This is a comment
' This is a comment
```

```
Dim tax1 As Double // contains the tax rate
Dim tax2 As Double ' contains the tax rate
Dim tax2 As Double Rem contains the tax rate
```

# RemoveHandler

Removes an event handler that was added by AddHandler.

## Keyword Summary

Name	RemoveHandler
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">AddHandler</a>

## Syntax

```
RemoveHandler eventName, delegateMethod
```

### Parameters

<i>eventName</i>	The name of the event that you are handling.
<i>delegateMethod</i>	The name of the method that is used to handle the event.

## Notes

Always remove any handlers added with AddHandler when you are finished with them.

## Sample Code

This example lets you handle the Timer.Action event without creating a Timer subclass.

Start by adding a property to the layout:

```
MyTimer As Xojo.Core.Timer
```

Next, add a ProgressBar to the layout. The Timer will simply update the ProgressBar.

In the Open event handler of the layout, instantiate the Timer and indicate that its Action event handler should be handled by a method, TimerAction, that you will add to the layout:

```
MyTimer = New Xojo.Core.Timer
MyTimer.Period = 1000
MyTimer.Mode = Xojo.Core.Timer.Modes.Multiple
```

---

```
AddHandler MyTimer.Action, AddressOf TimerAction
```

Now add the TimerAction method to the layout:

```
Sub TimerAction(sender As Xojo.Core.Timer)
  If ProgressBar1.CurrentValue < ProgressBar1.MaxValue Then
    ProgressBar1.CurrentValue = ProgressBar1.CurrentValue + 1
  Else
    // Stop Timer and Remove the handler
    sender.Mode = Xojo.Core.Timer.Mode.Off
    RemoveHandler MyTimer.Action, AddressOf TimerAction
  End If
End Sub
```

Remember that the first parameter to TimerAction must be of the type of the original object, in this case a Timer.

When you run the project, the ProgressBar is updated once per second.

# Return

Exits the current method, returning to the calling method.

## Keyword Summary

Name	Return
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Methods</a> <a href="#">Event</a> <a href="#">Exit</a>

## Syntax

```
Return [value]
```

## Notes

Methods automatically return when you reach the end of the code in the method. You can use the Return command to exit a method early based on a condition.

You must specify a return value for functions and events that are declared to have a return value. The return value must match the type in the declaration.

The return value can be an array or a single value. If you want to return an array, put empty parentheses after the name of the data type in the Return Type area for the declaration in the Inspector. For example, if you want to return an array of Integers, this would be the return type:

```
Integer()
```

If you need to return a multi-dimensional array, place one fewer commas in the parentheses than dimensions. For example, to return a two-dimensional array of Doubles, use this as the return type:

```
Double(,)
```

## Sample Code

```
// Exit early from a method
Sub Test(value As Text)
    If value = "Hello" Then Return
```

```
    Label1.Text = value  
End Sub
```

```
// Return from a function  
Function calc(value As Integer) As Integer  
    value = value * 2  
    Return value  
End Sub
```

# Select

Starts a Select...Case statement.

## Keyword Summary

Name	Select
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Select...Case</a>

## Sample Code

```
Dim i As Integer = 5
Dim output As Text

Select Case i
Case Is < 4
    output = "Less than four"
Case 5
    output = "Five"
Else
    output = "Greater than five"
End Select
```

# Select...Case

Used to execute one set of statements out of a group based on the value of an expression.

## Keyword Summary

Name	Select...Case
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">If...Then...Else</a> <a href="#">Controlling Code Flow</a>

## Syntax

```
Select Case expression
.....
Case [Is] expression-n | Case expression-n, expression-n | Case expression-n To expression-n
.....
statements
.....
[Else]
.....
statements
.....
[Case Else]
.....
statements
.....
End [Select]
```

<i>expression</i>	An expression that evaluates to to a value that can be compared.
<i>expression-n</i>	An expression or list of expressions that are compared to <i>expression</i> . The data type must match <i>expression</i> . This can be a single value, a comma-delimited list of values, function(s) that returns a value, a range of values using the To keyword, or an expression that uses the Is or ISA keywords.
statements	One or more code statements that are executed.

## Notes

Select...Case is useful when there are several possible conditions to check. Unlike an If statement, Select...Case exists as soon as it finds a matching Case expression to use. If there are no Case expressions that match, then the

statements in the Else or Case Else sections are used.

**i** There can only be a single Else or Case Else section.

Variables declared inside a Case block are local to the block and go out of scope when leaving the block.

## Example Case Expressions

```
Case 2, 4, 6, 8 // several values
Case 2 To 5 // range of values from 2 to 5 (inclusive)
Case 2 To 5, 7,9,11 // Both separate values and range
Case myFunction(x) // a Function
Case Is >= 42 // greater than/equal to operator
Case Is <19 // less than operator
Case IsA Label // tests whether an object is a Label
```

## Sample Code

```
Dim dayNumber As Integer
Dim day As Text
dayNumber = 3
```

```
Select Case dayNumber
Case 2
    day = "Monday"
Case 3
    day = "Tuesday"
Case 4
    day = "Wednesday"
Case 5
    day = "Thursday"
Case 6
    day = "Friday"
Else
    day = "Weekend."
End Select
// day = "Tuesday"
```

# Self

Refers to the primary (main) class containing the code.

## Keyword Summary

Name	Self
Type	Keyword
Project Types	All
Platforms	All
Related	<u>Me</u>

## Syntax

```
Self.MemberName
```

## Notes

By default, references to members (properties, methods, etc.) of a class use the Self prefix by default. For example, to call a method on a class, you can just use the method name:

```
MyMethod(123)
```

Alternatively, you can also use the Self prefix if you need additional clarity:

```
Self.MyMethod(123)
```

Self is related to the Me prefix, which is specifically used to refer to members within the event handler of a class added to a Window, WebPage, iOSView, ContainerControl or WebContainer.

## Sample Code

```
// Code in the Open event handler of an iOSView that sets its Title property
Event Open()
    Self.Title = "Hello"
End Event
```

# Soft

Indicates that a Declare should be resolved when used at run-time rather than when the app is launched.

## Keyword Summary

Name	Soft
Type	Keyword
Project Types	All
Platforms	All
Related	<u>Declare</u>

# Static

Declares a local variable for a specific data type. When using Static, the value for the variable is remembered when the method that declared it is called again.

## Keyword Summary

Name	Static
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Arrays</a> <a href="#">Dim</a>

## Syntax

```
Static variableName As dataType [= initialValue]
```

```
Static variableName As New dataType
```

```
Static variableName, variableNameN As dataType [= initialValue]
```

Static can also declare [arrays](#).

## Notes

Essentially, Static creates a locally-scoped global variable. The variable you create retains its value permanently, but it can only be accessed from within the method that declared it.

Use of Static can result in a performance improvement for a value that is assigned by a function that takes noticeable time to calculate.

## Sample Code

```
Static age As integer
age = 33
```

```
Static firstName, lastName As Text
```

```
// This function returns an ever-increasing number
Function SerialNumber As Integer
    Static currentSerialNumber As Integer = 0
    currentSerialNumber = currentSerialNumber + 1
```

Return currentSerialNumber  
End Function

# Step

Specifies the counting increment (or decrement) in For...Next loops.

## Keyword Summary

Name	Step
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">For...Next</a>

## Sample Code

```
For i As Integer = 1 To 10 Step 2
    ...
Next
```

# Super

Used in a subclass to access methods in its superclass that have been overridden by the subclass.

## Keyword Summary

Name	Super
Type	Keyword
Project Types	All
Platforms	All
Related	Class

## Syntax

```
Super.methodname
```

## Notes

Super can only be used to access methods of the superclass, not properties, because only methods can be overridden.

If you add a Constructor to a control subclass, the Code Editor will automatically add a call to the Constructor of the Super. Do not remove this as it may cause the control subclass to not work as expected.

# Then

Indicates the Then part of an If...Then...Else statement.

## Keyword Summary

Name	Then
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">If...Then...Else</a>

## Sample Code

```
If True Then  
    // Your code  
End If
```

# To

Separates the beginning and ending values of a For...Next loop counter or a value of a match range in a Select...Case.

## Keyword Summary

Name	To
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">For...Next</a> <a href="#">Select...Case</a>

## Sample Code

```
For i As Integer = 1 To 10
    ...
Next
```

```
Dim value As Integer = 10
Select Case value
    Case 1 To 5
        ...
    Case 10 To 20
        ...
    Else
End Select
```

# True

Assigns or tests the boolean value True. A comparison of two values that are equal results in True.

## Keyword Summary

Name	True
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">And</a> <a href="#">Not</a> <a href="#">Or</a> <a href="#">Xor</a> <a href="#">False</a>

## Syntax

```
expression = True
```

## Sample Code

```
Dim b As Boolean  
b = True
```

# Try

Starts a [Try...Catch](#) code block.

## Keyword Summary

Name	Try
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Try...Catch</a>

## Sample Code

```
Try
  Dim a(5) As Integer
  a(6) = 45 // Raises an OutOfBoundsException
Catch e As OutOfBoundException
  // The above exception is caught here for you to handle
  ErrorLabel.Text = "Exceeded size of array"
End Try
```

# Try...Catch

Used to handle exceptions raised in your code.

## Keyword Summary

Name	Try...Catch
Type	Command
Project Types	All
Platforms	All
Related	<a href="#">Exceptions</a> <a href="#">Raise</a> <a href="#">Exception Handling</a>

## Syntax

```

Try
    // Your code
Catch [exceptionParameter] [As exceptionType]
    // Your code
[Catch [exceptionParameter] [As exceptionType]
[Finally]
End Try

```

You can specify an exception parameter and exception type so that you can catch specific exceptions.

## Notes

Only exceptions that are raised in code within the Try statement can be caught by the Catch statement. If you do not catch an exception that was raised, the exception is passed up the calling stack until it eventually reaches the UnhandledException event for the application (if implemented). If the exception is left completely unhandled, then your application terminates.

You can have multiple Catch blocks to catch different types of exceptions

You can nest Try...Catch commands. If an inner Try...Catch does not handle an exception, then the next outer Try...Catch block will attempt to handle it.

You can also have an optional Finally block (after all the Catch blocks). A Finally block executes regardless of whether an exception was raised. Note that the Finally block does not get called if your method returns or exits in

the Try or Catch blocks.

You should try to avoid catching the `RuntimeException` superclass as this could cause prevent threads from properly being killed or apps from quitting if an exception is raised during those events. If you must catch `RuntimeException`, then be sure to re-raise the exception if it is not one that you need. For example:

```
Try
  Dim d As Date
  Dim t As Text = d.ToText
Catch e As RuntimeException
  If e IsA NilObjectException Then
    // your code
  Else
    // Re-raise the exception for the framework
    Raise e
  End If
End Try
```

## Sample Code

```
Try
  Dim a(5) As Integer
  a(6) = 45 // Raises an OutOfBoundsException
Catch e As OutOfBoundsException
  // The above exception is caught here for you to handle
  ErrorLabel.Text = "Exceeded size of array"
End Try
```

# Ubound

Returns the upper bound of the array (the index of the last element).

## Method Summary

Name	Ubound
Type	Method
Project Types	All
Platforms	All
Related	<a href="#">Arrays</a> <a href="#">Array UBound method</a>

## Syntax

```
result = Ubound(array, [dimension])
```

## Notes

The Ubound function is used to determine the last element of an array, but it can also be used to determine the size of an array. It may appear at first that the last element number and the size of the array are the same but in fact they are not. All arrays have a zero element. In some cases element zero is used and in other cases it is not. You will need to keep this in mind when using the Ubound function to determine the number of values you have in the array. For example, if the array is zero-based, then element zero is used to store a value and you will have to add one to the value returned by the Ubound function to get the number of values in the array.

For **multi-dimensional arrays**, Ubound returns the index of the last element of the dimension you specify, or, if you do not specify a dimension, it returns the value for the first dimension. The first dimension is numbered 1.

# Until

Specifies a condition that ends a Do...Loop loop.

## Keyword Summary

Name	Until
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Do...Loop</a>

## Sample Code

```
Dim x As Integer
Do
    x = x + 1
Loop Until x >= 100
```

# Using

Makes all of the public declarations in a module (or namespace) available for inside the scope of the code as if it were defined there.

## Keyword Summary

Name	Using
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Global</a> <a href="#">Namespaces</a>

## Syntax

```
Using [Global.]identifier1.identifier2
```

```
Using [Global.]identifier
```

## Notes

The only legal target for a Using statement is a module (namespace). The first identifier will be resolved recursively, from inner to outer scope as usual. If you would rather skip the recursive search, you may use the Global variable.

When Using is used in code, it obeys code block scoping rules.

You can also use Using with classes and modules (Insert->Using Clause). When used in this manner, the Using applies to the entire class or module and all code contained within it.

## Sample Code

Access the Dictionary without using its full namespace:

```
Using Xojo.Core
Dim d As New Dictionary
```

Avoid the Xojo namespace:

```
Using Xojo
Dim d As New Core.Dictionary
```

Code block scoping:

```
If True Then  
    Using Xojo.Core  
    Dim d As New Dictionary  
End If
```

```
Dim d As Xojo.Core.Dictionary // Using statement is out of scope here
```

# Wend

Indicates the end of a While...Wend loop.

## Keyword Summary

Name	Wend
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">While...Wend</a>

## Sample Code

```
Dim x As Integer
While x < 100
    x = x + 1
Wend
```

# While

Starts a While...Wend loop.

## Keyword Summary

Name	While
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">While...Wend</a>

## Sample Code

```
Dim x As Integer
While x < 100
    x = x + 1
Wend
```

# While...Wend

Repeatedly executes a series of statements while the specified condition is True.

## Keyword Summary

Name	While...Wend
Type	Keyword
Project Types	All
Platforms	All
Related	<a href="#">Do...Loop</a> <a href="#">For...Next</a> <a href="#">For Each...Next</a> <a href="#">Controlling Code Flow</a>

## Syntax

```

While condition
    Statements
    [Continue]
    [Exit [While]]
Wend
  
```

<i>condition</i>	Any valid boolean expression. When the expression evaluates to True, the statements in the loop are executed. When the condition evaluates to False, the loop ends and any code following the Wend command is executed.
Continue	The Continue keyword causes the execution of code to skip directly to the Wend statement, skipping over any lines in between.
Exit	The Exit keyword immediately exits the loop so that the statement immediately following the Wend statement is executed.

## Notes

While statements can be nested to any level, with each Wend statement matching its corresponding While statement.

Variables declared within a While loop are local to the While loop and go out of scope when the loop ends.

## Sample Code

```
Dim x As integer
While x < 100
    x = x + 1
Wend
' x = 100</code>
```

Using Exit:

```
Const kAttempts = 10
Dim attempt As Integer = 1
Dim output As Text

While attempt <= kAttempts
    Dim randomValue As Integer = Math.RandomInt(1, 10)
    If randomValue > 5 Then
        output = "Found a random value above 5 after " + attempt.ToText + " iterations."
        Exit
    End If
    attempt = attempt + 1
Wend

If attempt > kAttempts Then
    output = "Found NO random value above 5 after " + kAttempts.ToText + " iterations."
End If</code>
```

Using Continue:

```
Const kAttempts = 100
Dim matchCount As Integer
Dim attempt As Integer

While attempt < kAttempts
    attempt = attempt + 1
    Dim randomValue As Integer = Math.RandomInt(1, 10)
    If randomValue <= 5 Then
        Continue
    End If
    matchCount = matchCount + 1
Wend

Dim output As Text
output = "Found " + matchCount.ToText + " random values above 5 within " +
kAttempts.ToText + " iterations."
```

# Compiler Constants

You can use Compiler Constants to isolate code for target-specific or other reasons.

Target32Bit	True if the app is built and running as a 32-bit executable.
Target64Bit	True if the app is built and running as a 64-bit executable.
TargetARM*	True if building an app for an ARM-compatible CPU or the app is running on an ARM-compatible CPU.
TargetBigEndian	Indicates the type of endianness used by the underlying CPU.
TargetCocoa	Always True for OS X apps.
TargetConsole	True is building or running a console app.
TargetDesktop	True if building or running as a desktop app.
TargetiOS	True if building or running as an iOS app.
TargetLinux	True if building a Linux app or the app is running on Linux.
TargetLittleEndian	Indicates the type of endianness used by the underlying CPU.
TargetMacOS	True if building an OS X app or the app is running on OS X.
TargetRemoteDebugger	True if the app is running via the Remote Debugger.
TargetWeb	True if building or running as a web app.
TargetWin32	True if running on any version of Windows (even 64-bit).
TargetX86	True if building an app for an Intel x86-compatible CPU or the app is running on an Intel x86-compatible CPU.
TargetXojoCloud	True if building a web app for Xojo Cloud or for a web app that is running on Xojo Cloud.
DebugBuild	True if running the app in the IDE.
XojoVersion	The major and minor version of Xojo as a Double in the form 20xx.yy, where xx is the 2-digit year and yy is the release number. For example, Xojo 2014 Release 3.2 would return 2014.32.
XojoVersionString	The version of Xojo as Text. For example, Xojo 2014 Release 3.2 would return "2014r3.2".
CurrentMethodName	Contains the fully-qualified name of the method (or event) where it is used. For example, in Window1's Open event handler it contains "Window1.Open". In the Action event handler of Button1 on Window1, it is "Window1.Button1.Action".

\* Not yet available

## Sample Code

```
#If TargetDesktop Then
  Dim c As New Clipboard
#Endif

#If DebugBuild Then
  // Do extra logging
#Endif

// Save settings
If TargetWin32 Then
  // Use the Registry
ElseIf TargetMacOS Then
  // Use a plist
ElseIf TargetLinux
  // Use XML
End If

Label1.Text = "Made with Xojo " + XojoVersionString
```

# Conditional Compilation (#If...#Endif)

Conditional compilation allows you to isolate code based on a constant to ensure it gets included or to prevent it from being included when building the project. These commands are prefixed with the "#" symbol.

## Syntax

```
#If constantExpression [Then]
    // target-specific code
[#Else]
    // other target-specific code
[#ElseIf constantExpression]
    // other target-specific code
#Endif

#If TargetBoolean Then <one line of target-specific code>
```

## Notes

You can use any of the built-in [Compiler Constants](#) or any of your own constants with conditional compilation.

#If statements can be written on one line (without a #Endif) if there are no #Else or #Elseif statements. When written on one line, the "Then" keyword is required.

Use conditional compilation to isolate target-specific code such as API calls, file handling, UI, etc.

The block of code that is evaluated by conditional compilation is included in the build. The rest of the code that is not evaluated is not included in the build.

## Sample Code

```
// Set path separator
Dim separator As Text
#If TargetWin32 Then
    separator = "\"
#Else
    separator = "/"
#Endif
```

# Data Types

This section covers the intrinsic data types. These are data types that are part of the Xojo language and are not classes.

- [Arrays](#)
- [Auto](#)
- [Boolean](#)
- [Color](#)
- [Currency](#)
- [Double](#)
- [Integer](#)
- [Single](#)
- [Structure](#)
- [Text](#)

# Arrays

An array is an indexed collection of data for a specific data type. Arrays themselves are not a data type, but any data type can be defined as an array using the [Dim](#) statement.

## Item Summary

Name	Arrays
Type	Concept
Methods	<a href="#">Append</a> , <a href="#">IndexOf</a> , <a href="#">Insert</a> , <a href="#">Pop</a> , <a href="#">Remove</a> , <a href="#">Shuffle</a> , <a href="#">Sort</a> , <a href="#">SortWith</a> , <a href="#">UBound</a>
Targets	All
Platforms	All
Related	<a href="#">Array</a> , <a href="#">Dim</a> , <a href="#">ReDim</a> , <a href="#">Dictionary</a> , <a href="#">ParamArray</a> , <a href="#">Text.Split</a> , <a href="#">Text.Join</a>

## Syntax

```
Dim arrayName() As dataType
```

```
Dim arrayName(), arrayNameN() As dataType [= Array(value1, value2, ..., valueN)]
```

## Notes

All arrays are indexed starting at position 0.

Arrays are created by specifying the upper bound for the array when you Dim it. Because arrays all start at position 0, an array has one more element than the number you use as the upper bound. For example, this creates an array with 5 elements, which you can then access using the index:

```
Dim aNames(4) As Text
aNames(0) = "Bob"
aNames(1) = "Bill"
aNames(2) = "Ben"
aNames(3) = "Brad"
aNames(4) = "Bart"
```

Constants can be used when declaring the upper bound of an array:

```
Const kBound As Integer = 4
Dim aNames(kBound)
```

You can also create empty arrays that do not specify a size. At run-time, your code can add elements to it or you can choose to ReDim it to a specific size. These are the two ways to create an empty array:

```
Dim aFirstNames() As Text
Dim aLastNames(-1) As Text
```

If you try to access an element of an array that does not exist, you will get an `OutOfRangeException`. Here are some examples that would raise the exception:

```
aNames(5) = "Tom" // this exceeds the upper bound of aNames
```

```
aFirstName(1) = "Bill" // aFirstName is still empty
```

Use the methods `Add`, `Insert` to add additional elements to arrays. These methods are the only ways to add elements to empty arrays.

## Multi-Dimensional Arrays

A multi-dimensional array has elements in 2 or more dimensions. A table would be an example of a 2-dimensional array because it has columns and rows.

Most of the array methods are not supported for multi-dimensional arrays.

You declare multi-dimensional arrays by specifying the upper bound for each dimension:

```
// A 5x5 table
Dim table(4, 4)
```

## Methods

### Append(value)

Appends a new element to the end of a one-dimensional array, increasing the size of the array by one. The *value* must match the data type of the array.

#### Sample Code

```
Dim names() As Text
names.Append("Bob")
```

### IndexOf(value, startIndex = 0)

Searches sequentially through a one-dimensional array and returns the index of the specified value. The value must match the data type of the array. Specify a *startIndex* to start the search at the specified index in the array.

#### Exceptions

<code>OutOfRangeException</code>	If <i>startIndex</i> is outside the bounds of the array.
----------------------------------	--

## Sample Code

```
// Search for a value in the array
Dim days() As Text = Array("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")

Dim thursdayIndex As Integer
thursdayIndex = days.IndexOf("Thursday") // thursdayIndex = 3
```

---

## Insert(index As Integer, value)

Inserts a new element into a one-dimensional array at the specific index. The value must match the data type of the array.

### Exceptions

OutOfRangeException	If <i>index</i> is outside the bounds of the array.
---------------------	---

## Sample Code

```
Dim names() As Text = Array("Bob", "Tom", "Jim")
names.Insert(2, "Joe")

// names() = "Bob", "Tom", "Joe", "Jim"
```

---

## Pop As value

Removes the last element of the array and returns its value. The return type is the same data type as the array.

### Notes

Append and Pop can be used together to treat an array as a stack. The Append method pushes a new element onto the end of the array/stack and the Pop method removes the last element from the array/stack and returns its value.

### Exceptions

OutOfRangeException	If the array is empty (has no elements).
---------------------	--

## Sample Code

```
// Push three values onto an array and then Pops off the values
Dim stack() As Text
stack.Append("One")
stack.Append("Two")
stack.Append("Three")

Dim value As Text
```

```
value = stack.Pop // value = "Three"  
value = stack.Pop // value = "Two"  
value = stack.Pop // value = "One"
```

---

## Remove(index As Integer)

Removes the element at position *index* from a one-dimensional array, adjusting down all elements after it. The size of the array is reduced by one.

### Exceptions

OutOfBoundsException	If <i>index</i> is outside the bounds of the array.
----------------------	---

### Sample Code

```
Dim names() As Text = Array("Bob", "Tom", "Jim")  
names.Remove(1)  
  
// names() = "Bob", "Jim"
```

---

## Shuffle

Randomly rearranges the elements of an array.

### Notes

A [Fisher-Yates shuffle](#) is used. There is no ability to control the seed. An element of the original array has a roughly equal chance of appearing in any position of the array after the shuffle, regardless of the size and number of dimensions of the array.

### Sample Code

```
// Shuffle a one-dimensional array  
Dim values() As Integer = Array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)  
  
values.Shuffle  
  
Dim result As Text  
For i As Integer = 0 To values.UBound  
    result = result + i.ToString  
Next  
  
// Shuffle a two-dimensional array  
Dim myArray(5, 5) As Text  
For i As Integer = 0 To 5  
    For j As Integer = 0 To 5
```

```
    myArray(i, j) = i.ToText + j.ToText
Next
Next

myArray.Shuffle
```

---

## Sort

Does an ascending sort of the elements of a one-dimensional array.

### Notes

You can only use Sort with arrays for these data types: [Text](#), [Single](#), [Double](#), [Integer](#) (and all related Integer types) and String.

### Sample Code

```
// Sort an array
Dim names() As Text = Array("Jim", "Bob", "Jack")
names.Sort

// names = "Bob", "Jack", "Jim"

// For a descending sort, simply access the array in reverse order
Dim reverseNames() As Text
For i As Integer = names.UBound DownTo 0
    reverseNames.Append(names(i))
Next

// reverseNames = "Jim", "Jack", "Bob"</code>
```

---

## Sort(sortMethod As Delegate)

 Available in 2015r3

This method is available for all non-structure single dimension arrays. You create and pass a delegate method to do the sorting.

### Notes

The delegate has two parameters, value1 and value2 (which are the same type as the array elements), and returns an integer. The delegate function should return a positive value if it considers value1 to be greater than value2, zero if it considers value1 and value2 to be equal, and a negative value if it considers value1 to be less than value2. An element that is less than another element will have a lower index in the array once sorting is finished.

Like the current Sort function, this modifies the array in place.

Delegate behavior:

- The delegate function must provide a stable ordering of the elements it is comparing. Failing to do so can result in infinite loops or other undefined behavior.
- The delegate function must not mutate or examine at the contents of the array while it is being sorted. Doing so will result in undefined behavior.
- The delegate must not be Nil. If Nil is passed, a NilObjectException is raised.
- The delegate function should not raise exceptions. If an exception is raised, the array is still valid and contains the same values but the order of elements is undefined.

### Sample Code

Sort an array of Dates:

```
Function DateCompare(value1 As Xojo.Core.Date, value2 As Xojo.Core.Date) As Integer
    ' This assumes the array is populated with non-Nil dates
    If value1.SecondsFrom1970 > value2.SecondsFrom1970 Then Return 1
    If value1.SecondsFrom1970 < value2.SecondsFrom1970 Then Return -1
    Return 0
End Function
```

```
Dim myArray() As Xojo.Core.Date
myArray.Append(Xojo.Core.Date.Now)
myArray.Append(New Xojo.Core.Date(2015, 8, 1, Xojo.Core.TimeZone.Current))
myArray.Append(New Xojo.Core.Date(2014, 4, 1, Xojo.Core.TimeZone.Current))
myArray.Append(New Xojo.Core.Date(2016, 11, 1, Xojo.Core.TimeZone.Current))
```

```
myArray.Sort(AddressOf DateCompare)
' The array is now sorted
```

---

## SortWith(withArray1, ..., withArrayN)

Does an ascending sort of one or more one-dimensional arrays in the same order as the base array.

### Notes

You can only use Sort with base arrays for these data types: [Text](#), [Single](#), [Double](#), [Integer](#) (and all related Integer types) and String.

Results are undefined when the elements of the base array are not unique, such as when they all consist of the same integer value.

In addition to using SortWith for sorting an array of objects by one of its properties, you can also use the new [Sort method](#) that takes a Delegate sorting method. You can copy each object's property value into a base array and then use SortWith on the base array, providing the object array as a *withArray*.

### Sample Code

```
// Sort two related arrays
Dim names() As Text = Array("Mozart", "Bing", "Jackson", "Flintstone")
Dim zips() As Text = Array("04101", "04240", "04123", "04092")
```

```
names.SortWith(zips)

// names() = "Bing", "Flintstone", "Jackson", "Mozart"
// zips = "04240", "04092", "04123", "04101"

Sort an array of objects using one of its values:

// Person is a class containing a LastName property
// Assume there is an array called People() As Person that contains
// a collection of Person objects
// Save the last names into their own array
Dim lastNames() As Text
For i As Integer = 0 To People.UBound
    lastNames.Append(People(i).LastName)
Next

// Now sort the last names and provide the people array
lastNames.SortWith(People)

// The People array is now sorted to match the last names
```

---

## UBound

Returns the upper bound of the array (the index of the last element).

### Notes

You will most often use UBound with [For..Next](#) loops.

Also refer to the [Ubound](#) command.

### Sample Code

```
Dim names() As Text = Array("Bob", "Tom", "Jack")

For i As Integer = 0 To names.UBound
    // Add to a table or list
    iOSTable1.AddRow(names(i))
Next
```

# Auto (Data Type)

The Auto type can store and retrieve any type value, but cannot convert to other types.

## Item Summary

Name	Auto
Type	Data Type
Inherits	n/a
Implements	n/a
Targets	All
Platforms	All
Related	<a href="#">Introspection</a>

## Notes

Once a value is placed into an Auto variable, it is treated as if it is that type of the value. For example, if you assign an Integer to an Auto variable, you can not later use it as a Text. You'll have to first assign it to an Integer and then convert it to a Text using `intValue.ToString`.

You can use [Introspection](#) to check the type of the value contained in an Auto variable.

You can assign a value of a different type to an Auto that already had a value.

## Sample Code

Simple Auto examples:

```
' Add an Auto containing an integer
Dim num As Auto
num = 42

Dim sum As Integer
sum = num + 10

' Display an Auto containing an integer
Dim numInt As Integer = num
Dim output As Text
output = numInt.ToString ' output contains "42"

' Convert an Auto containing a Double to Text
Dim num As Auto = 5.5
```

```
Dim t As Text = CType(num, Double).ToText
```

Check the type of an Auto variable:

```
Dim autoVar As Auto = 42  
Dim info As Xojo.Introspection.TypeInfo  
info = Xojo.Introspection.GetType(autoVar)
```

```
Label1.Text = "Type: " + info.Name ' Displays Int32
```

# Boolean

Boolean is a data type used to store True or False. The default value is False.

## Data Type Summary

Name	Boolean
Type	Data Type
Targets	All
Platforms	All
Related	True, False, Dim, <u>Not</u> , <u>And</u> , <u>Or</u>

## Example

Declares boolean variables and set their values to True:

```
Dim isTaxable As Boolean  
isTaxable = True
```

```
Dim DocumentModified As Boolean = True
```

Enable a TextField:

```
TextField1.Enabled = True
```

# Color

The data type for storing color values with an optional alpha component.

## Data Type Summary

Name	Color
Type	Data Type
Constants	<ul style="list-style-type: none"> <li>Black</li> <li>Blue</li> <li>Brown</li> <li>Clear</li> <li>Cyan</li> <li>DarkGray</li> <li>Gray</li> <li>Green</li> <li>LightGray</li> <li>Magenta</li> <li>Orange</li> <li>Purple</li> <li>Teal</li> <li>White</li> <li>Yellow</li> </ul>
Methods	<ul style="list-style-type: none"> <li><u>Alpha</u></li> <li><u>Blue</u></li> <li><u>Cyan</u></li> <li><u>Green</u></li> <li><u>HSV</u></li> <li><u>HSVA</u></li> <li><u>Hue</u></li> <li><u>Magenta</u></li> <li><u>Red</u></li> <li><u>RGB</u></li> <li><u>RGBA</u></li> <li><u>Saturation</u></li> <li><u>Value</u></li> <li><u>Yellow</u></li> </ul>
Targets	All
Platforms	All
Related	<a href="#">iOSGraphics</a>

## Notes



Desktop, Web and Console projects can use the [CMY](#) global function to set the color using cyan,

**i** magenta and yellow elements.

Colors are often used to assign colors to properties of type `Color`. Use the `RGB` or `HSV` methods to assign a color or use the format:

```
&cRRGGBB
```

In the above, `RR` is the `RGB` value of Red in hexadecimal, `GG` is the value Green in hexadecimal, and `BB` is the value of Blue in hexadecimal. You can right-click in the Code Editor and select `Insert Color` to choose a color using the color picker, which will insert the hexadecimal value at the cursor position in the editor.

Desktop, Web and Console projects can use the `Str` global function to get the hexadecimal `RGB` value of a `Color`:

```
Dim c As Color = RGB(255, 100, 50)
Dim hex As String = Str(c)
```

## Alpha Channel Support

The `Color` type has a read-only `Alpha` property. The alpha channel is the transparency of the color represented as an integer between 0 (opaque) and 255 (transparent).

The `Color` literal syntax can optionally have two more hexadecimal digits at the end representing the alpha channel (for example, "&cFF00007F"). If the color literal has only 3 channels of information, the alpha component defaults to 0.

```
&cRRGGBBAA
```

The related methods (`RGB`, `HSV` and `CMY`) can also take an `Alpha` parameter.

## Platform Availability

Alpha support is available on all platforms, but requires `GDI+` to be enabled on Windows. To enable `GDI+`, set the `UseGDIPlus` property on the `Application` class to `True`:

```
App.UseGDIPlus = True
```

The `CMY` global function is not available for `iOS` projects.

## Sample Code

Set colors using literal values:

```
Dim blue As Color = Color.RGB(0, 0, 255)
Dim blueTransparant = Color.RGB(0, 0, 255, 255)
```

## Constants

Black	&c000000	
Blue	&c0000FF	
Brown	&c996633	
Clear	&c000000FF	
Cyan	&c00FFFF	
DarkGray	&c555555	
Gray	&c808080	
Green	&c00FF00	
LightGray	&cAAAAAA	
Magenta	&cFF00FF	
Orange	&cFF8000	
Purple	&c800080	
Red	&cFF0000	
Teal	&c008080	
White	&cFFFFFF	
Yellow	&cFFFF00	

## Sample Code

```
MyLabel.TextColor = Color.Red
```

## Methods

### Alpha As Integer (read-only)

The alpha channel is the translucency of the color represented as an integer between 0 (opaque) and 255 (transparent).

### Blue As Integer (read-only)

The amount (0-255) of blue in the color (RGB).

### Cyan As Double (read-only)

The value of Cyan (0-1) in the color (CMY).

---

## Green As Integer (read-only)

The amount (0-255) of green in the color (RGB).

---

## HSV(h As Double, s As Double, v As Double) As Color

Create a Color from hue, saturation and value.

---

## HSVA(h As Double, s As Double, v As Double, a As Integer) As Color

Create a Color from hue, saturation, value and an alpha channel.

### Notes

The alpha channel is the translucency of the color represented as an integer between 0 (opaque) and 255 (transparent).

---

## Hue As Double (read-only)

The value of Hue (0-1) in the color (HSV).

---

## Magenta As Double (read-only)

The value of Magenta (0-1) in the color (CMY).

---

## Red As Integer (read-only)

The amount (0-255) of red in the color (RGB).

---

## RGB(r As Integer, g As Integer, b As Integer) As Color

Create a Color from red, green and blue.

### Sample Code

Set a Color to red:

```
Dim red As Color = Color.RGB(255, 0, 0)
```

---

## RGBA(r As Integer, g As Integer, b As Integer, a As Integer) As Color

Create a Color from red, green, blue and alpha channel.

### Notes

The alpha channel is the translucency of the color represented as an integer between 0 (opaque) and 255 (transparent).

### Sample Code

Set a Color to red with full transparency:

```
Dim red As Color = Color.RGBA(255, 0, 0, 100)
```

---

## Saturation As Double (read-only)

The value of Saturation (0-1) in the color (HSV).

---

## Value As Double (read-only)

The value of Value (0-1) in the color (HSV).

---

## Yellow As Double (read-only)

The value of Yellow (0-1) in the color (CMY).

# Currency

A decimal number that holds 15 digits to the left of the decimal point and 4 digits to the right. The default value of a Currency is 0.0.

## Data Type Summary

Name	Currency
Type	Data Type
Bytes	8
Range	-922337203685477.5808 to 922337203685477.5807
Methods	<a href="#">ToText</a>
Shared Methods	<a href="#">FromText</a>
Targets	All
Platforms	All
Related	<a href="#">Double</a>

## Notes

Currency is a 64-bit fixed-point decimal. This means that Currency does not have some of the rounding issues that are inherent in floating-point numbers stored in Double.

Use Currency instead of Double when storing monetary values. Since Double is a [standard double-precision floating-point number](#), it is unable to store some specific decimal values. The Currency type avoids these issues, which can be particularly important with rounding and when dealing with money.

## Methods

### ToText(Optional locale As Locale) As Text

Converts a currency value to a Text value using the specified locale.

#### Notes

If no locale is specified, then [Locale.Raw](#) is used.

#### Sample Code

Convert Currency to Text:

```
Dim c As Currency = 123.45
```

```
Dim t As Text
t = c.ToText ' t = "123.45"
```

## Shared Methods

### FromText(theText As Text, Optional locale As Locale) As Currency

Converts a Text value containing a number that can be represented as a currency to a Currency value.

#### Notes

If no locale is specified, then [Locale.Raw](#) is used.

#### Exceptions

RuntimeException	If the Text value does not contain a valid currency for the locale.
------------------	---

#### Sample Code

Convert values to the Currency type:

```
Dim userValue As Text
userValue = "123.45"
```

```
Dim c As Currency
c = Currency.FromText(userValue)
```

```
Dim locale As New Xojo.Core.Locale("en-US")
Dim value As Currency
```

```
userValue = "$123.45"
value = Currency.FromText(userValue, locale) ' value = 123.45
```

# Double

A double-precision floating-point number. The default value of a Double is 0.0.

## Data Type Summary

Name	Double
Type	Data Type
Bytes	8
Range	Max value: $\pm 1.79769313486231570814527423731704357e+308$ Min towards 0: $\pm 4.94065645841246544176568792868221372e-324$
Methods	<a href="#">Equals</a> <a href="#">ToText</a>
Shared Methods	<a href="#">FromText</a> <a href="#">Parse</a>
Targets	All
Platforms	All
Related	<a href="#">Currency</a> <a href="#">Single</a>

## Notes

Double is an IEEE double-precision, floating-point value. This means it is speedy but has some limits for values it can contain. For more information, refer to the wikipedia page about [floating point](#).

In most situations you should use Currency when dealing with monetary values.

## Methods

### Equals(numValue As Double, maxUlp As Integer) As Boolean

Compares two floating point values within a specified tolerance.

#### Parameters

<i>numValue</i>	The numeric expression being converted to the Double value.
<i>maxUlp</i>	The number of units in the last position that are used to denote the acceptable range in the test of equality. The default is 1. If you pass zero, then the test is the same as with =.

#### Returns

True if the Double is equal to *numValue* within the tolerance specified by *maxUlp*.

## Notes

Use this method rather than `=` when you need to determine whether two floating point numbers are close enough in value to be considered “equal.” This can be used to account for the imprecision of floating point division on computers, for example. It allows for a user-specified rounding error.

For *maxUlp*s, the last position refers to the the last byte in the binary representation of the mantissa. *maxUlp*s is the amount of difference between the last byte of the 2 numbers that is still acceptable. For example, consider these 2 numbers:

```
3.1415926535897932 ' last byte value is 0x18
3.141592653589795 ' last byte value is 0x1C
```

A *maxUlp*s of 3 will result in "not equal", while a *maxUlp*s of 4 will result in "equal".

## Sample Code

Compare two values:

```
Dim Pi As Double = 3.14159265358979323846264338327950
If Pi.Equals(22 / 7, 1) Then
    ' Close enough!
Else
    ' Not close enough
End if
```

## ToText(Optional locale As Locale, format As Text = "") As Text

Converts a Double value to a Text value using the optional locale and format.

## Notes

If no locale is specified, then Locale.Raw is used.

Refer to Unicode Number Format Patterns for a list of formats.

## Sample Code

Convert Double values to Text:

```
Dim d As Double = 123.45
```

```
Dim t As Text
t = d.ToText ' t = "123.45"
```

```
Dim n As Double = 1239.4567
Dim t As Text = n.ToText(Locale.Current, "#,###.##") ' t = 1,239.46
```

## Shared Methods

### FromText(theText As Text, Optional locale As Locale) As Double

Converts a Text value that containing a number that can be represented as a double to a Double.

#### Notes

If no locale is specified, then Locale.Raw is used.

#### Sample Code

Convert a Text value to a Double:

```
Dim userValue As Text
userValue = "123.45"
```

```
Dim d As Double
d = Double.FromText(userValue)
```

---

### Parse(theText As Text, Optional locale As Locale) As Double

Converts theText to a Double value.

#### Notes

If no locale is specified, then Locale.Raw is used.

Numbers are converted only if they are found at the beginning of the text. Any numbers that follow a non-numeric value are ignored. Empty text returns 0.

#### Sample Code

```
Dim d As Double
d = Double.Parse("123ABC")
// d = 123
```

# Integer

Used to store integer values. The default value is 0. Generally you will use the Integer data type (equivalent to Int32 on 32-bit builds or Int64 on 64-bit builds when available) or UInteger (equivalent to UInt32 on 32-bit builds or UInt64 on 64-bit builds when available). There are also other size-specific integer data types that are available, primarily for dealing with external OS APIs.

## Data Type Summary

Name	Integer
Type	Data Type
Bytes	4
Range	-2,147,483,648 to 2,147,483,647
Methods	<a href="#">ToBinary</a> <a href="#">ToHex</a> <a href="#">ToOctal</a> <a href="#">ToText</a>
Shared Methods	<a href="#">FromBinary</a> <a href="#">FromHex</a> <a href="#">FromOctal</a> <a href="#">FromText</a> <a href="#">Parse</a>
Targets	All
Platforms	All
Related	<a href="#">Dim</a> <a href="#">Static</a> <a href="#">Declare</a> <a href="#">Integer (size-specific)</a>

Name	UInteger
Type	Data Type
Bytes	4
Range	0 to 4,294,967,295

## Notes

If you assign a value that is larger (or smaller) than what the specific Integer type can hold, then the value will "overflow". This means the value will wrap around to the corresponding largest or smallest value and continue from there.

## Methods

### ToBinary(Optional minimumDigits As Integer) As Text

Converts the integer value to a Text containing its binary representation.

#### Sample Code

Convert an Integer value to the binary text value:

```
Dim i As Integer = 8
Dim binary As Text = i.ToBinary(4) ' binary = "1000"
```

---

### ToHex(Optional minimumDigits As Integer) As Text

Converts the integer value to a Text containing its hexadecimal representation.

#### Sample Code

Convert an Integer value to the hexadecimal text value:

```
Dim i As Integer = 255
Dim hex As Text = i.ToHex(4) // hex = "00FF"
```

---

### ToOctal(Optional minimumDigits As Integer) As Text

Converts the integer value to a Text containing its octal representation.

#### Sample Code

Convert an Integer value to the octal text value:

```
Dim i As Integer = 10
Dim octal As Text = i.ToOctal(4) // octal = "0012"
```

---

### ToText(Optional locale As Locale, format As Text = "") As Text

Converts the Integer to a Text value.

#### Notes

If no locale is specified, then [Locale.Raw](#) is used.

Refer to the [Unicode Technical Standard #35, appendix G \(Number Format Patterns\)](#) for information on how to specify a format.

## Sample Code

Convert an Integer value to Text:

```
Dim i As Integer = 42
Dim t As Text = "The number is " + i.ToString
```

```
' Add to an Integer directly and convert the new value
Dim n As Integer = 5
Dim t As Text = Integer(n+1).ToString
```

## Shared Methods

### FromBinary(theText As Text) As Integer

Converts a text form of a binary number to an Integer.

#### Sample Code

Convert "0110" to the integer value of 6:

```
Dim value As Integer
value = Integer.FromBinary("0110") ' value = 6
```

---

### FromHex(theText As Text) As Integer

Converts a text form of a hexadecimal number to an Integer.

#### Sample Code

Convert "ff" to the integer value of 255:

```
Dim value As Integer
value = Integer.FromHex("ff") ' value = 255
```

---

### FromOctal(theText As Text) As Integer

Converts a text form of an octal number to an Integer.

#### Sample Code

Convert "755" to the Integer value 493:

```
Dim value As Integer
value = Integer.FromOctal("755") ' value = 493
```

---

## FromText(theText As Text, Optional locale As Locale) As Integer

Converts a text form of a decimal number to an Integer. Specify a locale to convert thousands separator.

### Notes

If no locale is specified, then Locale.Raw is used.

### Exceptions

<u>BadDataException</u>	When <i>theText</i> contains anything other than an integer. Use <u>Parse</u> if you need to parse text that might contain non-numeric data.
-------------------------	--

### Sample Code

Convert a number in text to an integer:

```
Dim value As Integer
value = Integer.FromText("42") ' value = 42
```

This code converts a number using the US locale:

```
Dim locale As New Xojo.Core.Locale("en-US")
Dim value As Integer
value = Integer.FromText("1,234", locale) ' value = 1234
```

You can also use an exception to catch invalid data:

```
' Exception raised for invalid text, so you can handle your own
' special cases.
Dim value As Integer
Try
    value = Integer.FromText("123ABC")
Catch e As BadDataException
    value1 = -1 ' if data is invalid, just use -1
End Try</code>
```

---

## Parse(theText As Text, Optional locale As Locale) As Integer

Converts theText to an Integer value.

### Notes

If no locale is specified, then Locale.Raw is used.

Numbers are converted only if they are found at the beginning of the text. Any numbers that follow a non-numeric value are ignored. Empty text returns 0.

## Sample Code

Convert Text to an Integer:

```
Dim i As Integer
i = Integer.Parse("123ABC")
' i = 123
```

# Text

The Text type is used to store text with a defined encoding. It is a modern replacement for String, which did not require an encoding. By using Text instead of String, you can be assured that your text data always has a known encoding and is never treated as just a collection of raw data. All classes, methods and properties in the Xojo framework use the Text type. You can convert Text data to a String, if necessary.

## Data Type Summary

Name	Text
Type	Data Type
Constants	<a href="#">CompareCaseSensitive</a>
Methods	<a href="#">BeginsWith</a> , <a href="#">Characters</a> <a href="#">Codepoints</a> <a href="#">Compare</a> <a href="#">Empty</a> <a href="#">EndsWith</a> <a href="#">IndexOf</a> <a href="#">Left</a> <a href="#">Length</a> <a href="#">Lowercase</a> <a href="#">Mid</a> <a href="#">Replace</a> <a href="#">ReplaceAll</a> <a href="#">Right</a> <a href="#">Split</a> <a href="#">TitleCase</a> <a href="#">ToCString</a> <a href="#">Trim</a> <a href="#">TrimLeft</a> <a href="#">TrimRight</a> <a href="#">Uppercase</a>
Shared Methods	<a href="#">FromUnicodeCodepoint</a> <a href="#">FromCString</a> <a href="#">Join</a>
Targets	All
Platforms	All
Related	<a href="#">TextEncoding</a>

## Sample Code

Assign text to a Label:

```
Dim t As Text
t = "Hello, World"
MyLabel.Text = t
```

Text is available in all project types, so you can also use it in place of String. A Text value can be converted to a String, so code like this works:

```
Dim t As Text = "Hello, World!"
MsgBox(t) ' MsgBox takes a String, but this works because Text can be converted to String
```

You can also convert a String with a known encoding to a Text using the `String.ToText` method:

```
Dim s As String = "Hello"
Dim t As Text = s.ToText ' t = "Hello"</code>
```

## Notes

### Converting Text to and From Bytes

To get the bytes for the Text (using a [MemoryBlock](#)), you call [TextEncoding.ConvertTextToData](#) using a specific encoding.

To convert bytes in a [MemoryBlock](#) to Text, you call [TextEncoding.ConvertDataToText](#), specifying the encoding.

### Comparing Text vs. Strings

Text is abstract - a series of characters.

Bytes are concrete - a series of bits.

There are lots of different ways to encode characters into bytes. Most of them are very limited, only defining encodings for some characters, and even when they define encodings for the same characters, they often use different bytes.

The only encodings which can represent every character are the Unicode encodings: UTF-8, UTF-16, UTF-32.

The old String type tries to represent either text or bytes or both, and as a result it's complicated and confusing. With Text, this is now very simple: Text is characters, and if you want to convert to or from an array of bytes (or an old-fashioned String), you have to be clear about the encoding you intend to use.

When you say that you want to write an ASCII string to a serial port - well, you are actually writing bytes to the serial port, because you are doing something concrete, something that interchanges with other programs or machines. So you would convert the text to bytes, and you would do so using the ASCII encoding. Conversely, you can translate some bytes, contained in a String or a MemoryBlock, up to a Text value by specifying the encoding that was used to generate them.

### Technical Information

The Text type is an immutable series of [Unicode scalar values](#).

The documentation is very deliberate in its use of the terms character, code point, and scalar value. A character, in this context, refers to an [extended grapheme cluster](#) (also known as a user-perceived character). The terms [code point](#) and scalar value retain the meaning defined in the Unicode standard.

All of the APIs on the Text type operate in characters. For example, if the APIs worked in terms of Unicode code points, it would be possible to corrupt data using Left/Mid/Right if the positions happened to be in the middle of a composed character or grapheme cluster. Working in characters also avoids situations where the length of 'é' can be either 1 or 2.

Many of the functions in this API optionally take locales because different locales can have special rules for casing and comparing. The default behavior being to perform the operation in a locale-insensitive manner. Functions that perform comparisons also take option flags that specify how to perform the comparison (e.g. case sensitively). These flags are bit flags that are combined via the bitwise Or operator. If the combination of options is invalid, an exception is thrown.

## Constants

CompareCaseSensitive	1
----------------------	---

## Methods

**BeginsWith(other As Text, options As Integer = 0, Optional locale As [Locale](#) = Nil) As Boolean**

Determines whether the beginning of this Text instance matches the *other* text when compared using the specified comparison *options* and *locale*.

### Parameters

<i>other</i>	This text is matched with the beginning of the text.
<i>options</i>	(Optional) Use CompareCaseSensitive constant to have case-sensitive comparisons.
<i>locale</i>	(Optional) Used to specify a <a href="#">Xojo.Core.Locale</a> to use for the comparison.

### Return Value

Returns True if *other* matches the beginning of the text, False if it does not.

### Notes

By default this performs a case-insensitive comparison. To do a case-sensitive comparison, supply the CompareCaseSensitive constant to the options parameter.

By default comparisons are done in an invariant locale (i.e. not dependent on the user's preferences). The locale

parameter can be used to specify an explicit locale to do comparisons in.

### Exceptions

<code>RuntimeException</code>	When <i>options</i> are invalid (currently not 0 or 1).
<code>InvalidArgumentException</code>	When <i>other</i> is an empty text value.

### Sample Code

Check the beginning characters of some text:

```
Dim t As Text
t = "All we have to decide is what to do with the time that is given to us."

If t.BeginsWith("All") Then
    Label1.Text = "Text starts with 'All'."
End If
```

## Characters As Iterable

Returns an iterator that yields a Text value for each character, in order of first to last.

### Sample Code

Reverse the Text:

```
Dim t As Text = "Hello, World!"
Dim reverse As Text

For Each c As Text In t.Characters
    reverse = c + reverse
Next

' reverse = "!dlroW ,olleH"</code>
```

## Codepoints As Iterable

Returns an iterator that yields integer values for each Unicode scalar value that comprises the text.

### Sample Code

Look for Unicode 65:

```
For Each codePoint As UInt32 In myText.Codepoints
    If codePoint = 65 Then
```

```

    ' It is "A"
End If
Next</code>

```

## Compare(other As Text, Optional options As integer = 0, Optional locale As Locale = Nil) As Integer

Compares a text value with another text value. A non-empty text is always greater than an empty text. By default, a case-insensitive comparison is done.

### Parameters

<i>other</i>	The text to compare with the original text.
<i>options</i>	(Optional) Comparison options. Use constant <code>CompareCaseSensitive</code> for case-sensitive comparisons.
<i>locale</i>	(Optional) The locale to use for comparisons. By default an invariant locale (not dependent on the system settings) is used.

### Return Value

Returns a negative integer if the value is less than *other*, 0 if the two values are equal, and a positive integer if *other* is greater than the value.

### Notes

By default this performs a case-insensitive comparison. To do a case-sensitive comparison, supply the `CompareCaseSensitive` constant to the options parameter.

By default comparisons are done in an invariant locale (i.e. not dependent on the user's preferences). The locale parameter can be used to specify an explicit locale to do comparisons in.

### Exceptions

RuntimeException	When the specified <i>options</i> are invalid.
------------------	--

### Sample Code

Compare two text values:

```

Dim dog As Text = "Dog"
Dim cat As Text = "Cat"

Dim result As Integer
result = dog.Compare(cat)

```

```
' result > 0
```

---

## Empty As Boolean

Returns whether or not the text has contents. This will always be as fast or faster than checking length against zero.

### Return Value

Returns True when the text is empty, False when it is not.

### Sample Code

Check if a text value is empty:

```
Dim t As Text = ""
```

```
If t.Empty Then
```

```
    t = "Hello"
```

```
End If
```

---

## EndsWith(other As Text, Optional options As integer = 0, Optional locale As Locale = Nil) As Boolean

Determines if the text ends with *other* text.

### Parameters

<i>other</i>	The text to compare with the original text.
<i>options</i>	(Optional) Comparison options. Use constant CompareCaseSensitive for case-sensitive comparisons.
<i>locale</i>	(Optional) The locale to use for comparisons. By default an invariant locale (not dependent on the system settings) is used.

### Return Value

Returns True if the text ends with *other*, False if it does not.

### Notes

By default this performs a case-insensitive comparison. To do a case-sensitive comparison, supply the CompareCaseSensitive constant to the compareOptions parameter.

By default comparisons are done in an invariant locale (i.e. not dependent on the user's preferences). The locale parameter can be used to specify an explicit locale to do comparisons in.

## Exceptions

RuntimeException	If the options specified are invalid.
<a href="#">InvalidArgumentException</a>	If other is an empty text value

## IndexOf(other As Text, Optional compareOptions As Integer = 0, Optional locale As Locale = Nil) As Integer

Finds the position of *other* within the text.

## Parameters

<i>other</i>	The text to find within the original text.
<i>compareOptions</i>	(Optional) Comparison options. Use constant <code>CompareCaseSensitive</code> for case-sensitive comparisons.
<i>locale</i>	(Optional) The locale to use for comparisons. By default an invariant locale (not dependent on the system settings) is used.

## Return Value

Returns the zero-based location of *other* within the text. If *other* is not found, returns -1.

## Notes

By default this performs a case-insensitive comparison. To do a case-sensitive comparison, supply the `CompareCaseSensitive` constant to the `compareOptions` parameter.

By default comparisons are done in an invariant locale (i.e. not dependent on the user's preferences). The `locale` parameter can be used to specify an explicit locale to do comparisons in.

## Exceptions

RuntimeException	If the specified <i>options</i> are invalid.
<a href="#">InvalidArgumentException</a>	If <i>other</i> is an empty Text value.

## IndexOf(startPosition As Integer, other As Text, Optional compareOptions As Integer = 0, Optional locale As Locale = Nil) As Integer

Returns the zero-based location of given text value in this text starting at `startPosition`. If it's not found, returns -1.

## Parameters

<i>startPosition</i>	The zero-based starting position to start the search.
<i>other</i>	The text to compare with the original text.
<i>options</i>	(Optional) Comparison options. Use constant <code>CompareCaseSensitive</code> for case-sensitive comparisons.
<i>locale</i>	(Optional) The locale to use for comparisons. By default an invariant locale (not dependent on the system settings) is used.

## Return Value

Returns the zero-based location of *other* within the text. If *other* is not found, returns -1.

## Notes

By default this performs a case-insensitive comparison. To do a case-sensitive comparison, supply the `CompareCaseSensitive` constant to the `compareOptions` parameter.

By default comparisons are done in an invariant locale (i.e. not dependent on the user's preferences). The `locale` parameter can be used to specify an explicit locale to do comparisons in.

## Exceptions

<code>RuntimeException</code>	If the specified <i>options</i> are invalid.
<code>InvalidArgumentException</code>	If <i>other</i> is an empty Text value.
<code>OutOfBoundsException</code>	If <i>startPosition</i> is less than 0.
<code>OutOfBoundsException</code>	If <i>startPosition</i> is greater than the Text length.

## Left(count As Integer) As Text

Returns the first *count* characters of the Text value.

## Parameters

<i>count</i>	The number of characters to get.
--------------	----------------------------------

## Exceptions

<code>OutOfBoundsException</code>	If <i>maximumLength</i> is less than 0. If <i>count</i> is greater than <code>Text.Length</code> .
-----------------------------------	---

## Sample Code

Get the 5 left-most characters of the text:

```
Dim t As Text = "Hello, World!"  
Dim hello As Text = t.Left(5) ' hello = "Hello"
```

---

## Length As Integer

The number of characters.

## Sample Code

Get the length of some text:

```
Dim t As Text = "Hello, World!"  
Dim length As Integer = t.Length ' length = 13
```

---

## Lowercase(Optional locale As Locale = Nil) As Text

Creates a new text value that has its characters lowercased. If *locale* is non-*Nil*, it uses the locale's rules when performing the operation.

## Parameters

<i>locale</i>	Optional parameter specify specific locale rules to use.
---------------	--

## Sample Code

Set text to lower case:

```
Dim t As Text = "Hello, World!"  
t = t.Lowercase ' t = "hello, world!"
```

---

## Mid(start As Integer) As Text

An overload of *Mid* that returns all of the characters from *start* to the end of the text. The start position is a zero-based.

## Parameters

<i>start</i>	The Integer start position (0-based) to get the text.
--------------	---

## Exceptions

<i>OutOfBoundsException</i>	If <i>start</i> is less than 0.
-----------------------------	---------------------------------

OutOfBoundsException	If <i>start</i> is greater than the length of the text.
----------------------	---

### Sample Code

Get "World!" from text:

```
Dim t As Text = "Hello, World!"
t = t.Mid(7) ' t = "World!"
```

## Mid(start As Integer, length As Integer) As Text

Gets a portion of the characters in this text value. The *start* position is a zero-based. If the text value is shorter than the requested *length* of characters, the remaining text starting at the start is returned.

### Parameters

<i>start</i>	The Integer start position (0-based) to get the text.
<i>length</i>	The length of characters to get.

### Exceptions

OutOfBoundsException	If <i>start</i> is less than 0. If <i>start</i> is greater than Text.Length. If <i>length</i> is less than 0. If <i>start</i> + <i>length</i> is greater than Text.Length.
----------------------	---

### Sample Code

Get "World" from text:

```
Dim t As Text = "Hello, World!"
t = t.Mid(7, 5) ' t = "World"
```

## Replace(find As Text, replace As Text, Optional compareOptions As integer = 0, Optional locale As Locale = Nil) As Text

Creates a new value by replacing the first instances of the find parameter's value with the replacement parameter's value. If the input does not contain the requested value, nothing is replaced and the input is returned.

### Parameters

<i>find</i>	The text to find.
<i>replace</i>	The text to replace.
<i>options</i>	(Optional) Use CompareCaseSensitive constant to have case-sensitive comparisons.

<i>locale</i>	(Optional) Used to specify a <a href="#">Xojo.Core.Locale</a> to use for the comparison.
---------------	--

### Notes

By default this performs a case-insensitive comparison. To do a case-sensitive comparison, supply the `CompareCaseSensitive` constant to the `compareOptions` parameter.

By default comparisons are done in an invariant locale (i.e. not dependent on the user's preferences). The `locale` parameter can be used to specify an explicit locale to do comparisons in.

### Exceptions

<code>RuntimeException</code>	If the <i>options</i> specified are invalid.
<code>InvalidArgumentException</code>	If <i>find</i> is an empty text value.

### Sample Code

Replace "World" with "Mars":

```
Dim t As Text = "Hello, World!"
```

```
Dim newText As Text
newText = t.Replace("World", "Mars") ' newText = "Hello, Mars!"
```

---

## ReplaceAll(*find* As Text, *replacement* As Text, *Optional compareOptions* As Integer = 0, *Optional locale* As [Locale](#) = Nil) As Text

Creates a new value by replacing all instances of the `find` parameter's value with the `replacement` parameter's value. If the input does not contain the requested value, nothing is replaced and the input is returned.

### Parameters

<i>find</i>	The text to find.
<i>replacement</i>	The text to replace.
<i>options</i>	(Optional) Use <code>CompareCaseSensitive</code> constant to have case-sensitive comparisons.
<i>locale</i>	(Optional) Used to specify a <a href="#">Xojo.Core.Locale</a> to use for the comparison.

### Notes

By default this performs a case-insensitive comparison. To do a case-sensitive comparison, supply the `CompareCaseSensitive` constant to the `compareOptions` parameter.

By default comparisons are done in an invariant locale (i.e. not dependent on the user's preferences). The `locale`

parameter can be used to specify an explicit locale to do comparisons in.

### Exceptions

RuntimeException	If the <i>options</i> specified are invalid.
InvalidArgumentException	If <i>find</i> is an empty text value.

### Sample Code

Replaces all instances of "l" with "1":

```
Dim t As Text = "Hello, World!"
```

```
Dim newText As Text
newText = t.ReplaceAll("l", "1") ' newText = "HeLLo, WorLd!"
```

## Right(count As Integer) As Text

Returns the last *count* characters of the Text value.

### Parameters

<i>count</i>	The number of characters to get.
--------------	----------------------------------

### Exceptions

OutOfBoundsException	If <i>maxLength</i> is less than 0. If <i>count</i> is greater than Text.Length.
----------------------	---

### Sample Code

Get "World!" from text:

```
Dim t As Text = "Hello, World!"
t = t.Right(6) ' t = "World!"
```

## Split As Text()

Creates an array of the characters in the Text.

### Sample Code

Splits the text into an array with one element for each character:

```
Dim t As Text = "Hello, World!"
```

```
Dim chars() As Text
```

```
chars = t.Split
```

## Split(separator as Text, Optional compareOptions As Integer = 0, Optional locale As Locale = Nil) As Text()

Creates an array of the portions of the Text that are delimited by *separator*. If *separator* is the same as the text, an empty array is returned. If the text does not contain *separator*, an array containing only the original text value is returned.

### Parameters

<i>separator</i>	The separator value used to split the text.
<i>compareOptions</i>	(Optional) Comparison options. Use constant CompareCaseSensitive for case-sensitive comparisons.
<i>locale</i>	(Optional) The locale to use for comparisons. By default an invariant locale (not dependent on the system settings) is used.

### Notes

By default this performs a case-insensitive comparison. To do a case-sensitive comparison, supply the CompareCaseSensitive constant to the compareOptions parameter.

By default comparisons are done in an invariant locale (i.e. not dependent on the user's preferences). The locale parameter can be used to specify an explicit locale to do comparisons in.

### Exceptions

RuntimeException	If the <i>options</i> specified are invalid.
<u>InvalidArgumentException</u>	If <i>separator</i> is an empty text value.

### Sample Code

Split the text into "Hello" and "World":

```
Dim t As Text = "Hello World"
```

```
Dim words() As Text
words = t.Split(" ")
```

## TitleCase(Optional locale As Locale = Nil) As Text

Creates a new text value that has its characters titlecased. If the locale parameter is non-Nil, it will use that locale's rules when performing the operation.

## Parameters

<i>locale</i>	(Optional) The locale to use for comparisons. By default an invariant locale (not dependent on the system settings) is used.
---------------	--

## Sample Code

Convert text to "Hello, World!":

```
Dim t As Text = "hello, world!"
t = t.TitleCase ' t = "Hello, World!"
```

## ToCString(encoding As TextEncoding) As CString

Creates a CString from a Text with a specific encoding. The CString is immutable and is its own entity with the lifetime not tied to the source text value. This is provided to make dealing with declares easier.

## Parameters

<i>encoding</i>	The encoding to use to interpret the Text.
-----------------	--

## Notes

In general, CString is for use with [Declare](#) commands and [MemoryBlocks](#).

## Exceptions

NilObjectException	If <i>encoding</i> is Nil.
UnsupportedFormatException	If the text is cannot be represented in the given encoding. For example, Emoji is not representable in the ASCII encoding.

## Sample Code

Convert text to CString:

```
Dim t As Text = "Hello, World!"
Dim cs As CString = t.ToCString(Xojo.Core.TextEncoding.UTF8)
```

## Trim As Text

Trims whitespace, as defined in the Unicode standard, from the beginning and end of the text. If the value is empty or consists entirely of whitespace, an empty Text is returned.

## Sample Code

Removes beginning and ending white space:

```
Dim t As Text = " Hello, World! "
```

---

```
t = t.Trim ' t = "Hello, World!"
```

## TrimLeft As Text

Trims whitespace, as defined in the Unicode standard, from the beginning. If the value is empty or consists entirely of whitespace, an empty Text is returned.

### Sample Code

Removes beginning white space:

```
Dim t As Text = " Hello, World! "
```

---

```
t = t.TrimLeft ' t = "Hello, World! "
```

## TrimRight As Text

Trims whitespace, as defined in the Unicode standard, from end of the text. If the value is empty or consists entirely of whitespace, an empty Text is returned.

Removes ending white space:

```
Dim t As Text = " Hello, World! "
```

---

```
t = t.TrimRight ' t = " Hello, World!"
```

## Uppercase(Optional locale As Locale = Nil) As Text

Creates a new text value that has its characters lowercased. If the locale parameter is non-Nil, it will use that locale's rules when performing the operation.

### Parameters

<i>locale</i>	Optional parameter specify specific locale rules to use.
---------------	--

### Sample Code

Set text to upper case:

```
Dim t As Text = "Hello, World!"
```

---

```
t = t.Uppercase ' t = "HELLO, WORLD!"
```

## Shared Methods

### FromUnicodeCodepoint(*codepoint* As UInt32) As Text

Creates a new Text object from a single Unicode code point decimal value.

#### Notes

Visit [Unicode Lookup](#) to get decimal values to use with this method.

#### Parameters

<i>codepoint</i>	The Unicode codepoint (decimal) to create as text.
------------------	--

#### Exceptions

UnsupportedFormatException	If <i>codepoint</i> is not a Unicode scalar value (i.e. it is a high-surrogate code point or a low-surrogate code point).
----------------------------	---

#### Sample Code

Get the EndOfLine character:

```
Dim EOL As Text = Text.FromUnicodeCodepoint(10)
```

### FromCString(*str* As CString, encoding as [TextEncoding](#)) As Text

Creates a new Text object from a CString, interpreting it using the given encoding.

 Not yet available.

#### Parameters

<i>str</i>	The C string to convert to Text.
<i>encoding</i>	The encoding to use to interpret the C string.

#### Exceptions

NilObjectException	If <i>str</i> is Nil.
NilObjectException	If <i>encoding</i> is Nil.
UnsupportedFormatException	If <i>str</i> is not valid for the given encoding.

### Join(*items()* As Text, *separator* As Text) As Text

Creates a new Text object by concatenating each item in the items array together. If separator is not empty, it will be inserted between each item when performing the concatenation. If the items array is empty, an empty Text

value is returned.

### Parameters

<i>items()</i>	The array to join into a single text.
<i>separator</i>	The separator used to separate each element in the array when joined into the text.

### Exceptions

NilObjectException	If <i>items</i> is Nil.
--------------------	-------------------------

### Sample Code

Create a text from the array containing "Hello" and "World":

```
Dim words() As Text = Array("Hello", "World")
```

```
Dim newText As Text  
newText = Text.Join(words, ",") ' newText = "Hello,World"</code>
```

# Advanced Data Types

These data types are primarily used in conjunction with Declare commands that interface to OS APIs, which usually require information to be passed using a specific data type in order to work properly.

- [CFStringRef](#)
- [CString](#)
- [Delegate](#)
- [Integer \(size-specific\)](#)
- [OSType](#)
- [Ptr](#)
- [Single](#)
- [WString](#)

# CFStringRef (Data Type)

For use with OS X and iOS API calls.

## Item Summary

Name	CFStringRef
Type	Data Type
Inherits	n/a
Implements	n/a
Targets	All
Platforms	All
Related	<a href="#">Declare</a>

## Notes

CFStringRef implicitly convert to String or Text when assigned to String or Text variables.

### Memory Management

The Xojo framework handles memory management of CFStringRef objects smartly.

Based on rules explained under [Apple's Create Rule](#), a call to a declared function will either retain (CFRetain) the retrieved value or not. Once the CFStringRef object goes out of scope (i.e. it's no longer referenced by Xojo code), it will be released by calling CFRelease.

That means that you usually do not have to worry about proper Retain/Release calls for CFStringRef objects you retrieve using declares.

# CString (Data Type)

A null-terminated String.

## Item Summary

Name	CString
Type	Data Type
Targets	All
Platforms	All
Related	<a href="#">Declare</a> <a href="#">MemoryBlock</a> <a href="#">Text</a>

## Notes

Generally, CString is for use with [Declares](#) and [MemoryBlocks](#).

### Text

To convert a Text value to a CString, call [Text.ToCString](#). Remember, this will only convert the Text up to the first Null (0) byte value.

To convert a CString to a Text value, call [Text.FromCString](#).

### String

You can assign a String variable to a CString and it will be terminated with a null automatically. Remember, this will only convert the String up to the first Null (0) byte value.

CString implicitly converts to String when assigned to String variables.

# Delegate (Data Type)

Represents a specific method.

## Item Summary

Name	Delegate
Type	Data Type
Inherits	n/a
Implements	n/a
Methods	Invoke
Targets	All
Platforms	All
Related	<a href="#">AddressOf</a> <a href="#">AddHandler</a> <a href="#">WeakAddressOf</a>

## Notes

A Delegate data type is an object representing a specific method. It is a function pointer with a method signature.

Delegates decouple interface from implementation in a similar way to events or interfaces. This decoupling allows you to treat a method implementation as a variable that is changeable based on run-time conditions. They represent methods that are callable without knowledge of the target object. You can change the function the delegate points to on the fly.

A Delegate can be declared in either a module or a class. You use the Insert → Delegate menu command or the Add button in the Code Editor to create a Delegate entry.

A Delegate must have a name and can have optional parameters and a return type, which must match the method to which it delegates.

You cannot directly create a variable of type Delegate. You can only create variables of the type of Delegate that you actually added to the module or class.

## Constructors

### Constructor(p As Ptr)

Creates a delegate based on the supplied pointer, which is assumed to be correct for the delegate type.

## Sample Code

Assume there is a Delegate declared as SimpleProc:

```
Dim pp As Ptr = AnOSFunctionThatReturnsAFunctionPointer()  
Dim sp As New SimpleProc(pp)  
sp.Invoke()
```

## Methods

### Invoke([optional parameters] As [optional type])

Calls the method pointed to by the delegate. You must specify the same parameters (and return value) as the method to which the delegate refers.

## Integer (size-specific)

Used to store integer values using specific size types. The default value is 0. Generally you will use the `Integer` data type (equivalent to `Int32` on 32-bit builds or `Int64` on 64-bit builds when available) or `UInteger` (equivalent to `UInt32` on 32-bit builds or `UInt64` on 64-bit builds when available). This page describes the other size-specific integer data types that are available, primarily for dealing with external OS APIs.

### Data Type Summary

Name	Integer (size-specific)
Type	Data Type
Bytes	varies (see below)
Range	varies (see below)
Methods	<a href="#">ToBinary</a> <a href="#">ToHex</a> <a href="#">ToOctal</a> <a href="#">ToText</a>
Shared Methods	<a href="#">FromBinary</a> <a href="#">FromHex</a> <a href="#">FromOctal</a> <a href="#">FromText</a> <a href="#">Parse</a>
Targets	All
Platforms	All
Related	<a href="#">Dim</a> <a href="#">Static</a> <a href="#">Declare</a>

Name	UInteger
Type	Data Type
Bytes	4
Range	0 to 4,294,967,295

### Size-Specific Integer Types

Integer Data Type	Bytes Used	Number Range
Int8	1	-128 to 127
Int16	2	-32,768 to 32,767

Integer Data Type	Bytes Used	Number Range
Int32	4	-2,147,483,648 to 2,147,483,647
Int64	8	-2 <sup>63</sup> to 2 <sup>63</sup> -1 -9,223,372,036,854,775,808 to +9,223,372,036,854,775,807
UInt8 or Byte	1	0 to 255
UInt16	2	0 to 65,535
UInt32	4	0 to 4,294,967,295
UInt64	8	0 to 2 <sup>64</sup> -1 (18,446,744,073,709,551,615)

## Notes

If you assign a value that is larger (or smaller) than what the specific Integer type can hold, then the value will "overflow". This means the value will wrap around to the corresponding largest or smallest value and continue from there.

# OSType

A 4-byte integer known as a "fourcharcode," sometimes used on OS X and iOS. If you pass a string of four characters via this data type, it is automatically converted into a four char code Integer. Also, OSType does an implicit type conversion to String when you assign an OSType to a String variable. Note that the String "" is converted to the OSType "????".

## Item Summary

Name	OSType
Type	Data Typee
Targets	Desktop, Console, Web
Platforms	All
Related	<a href="#">Structure</a>

## Notes



Although this type is available on iOS, it cannot currently be used for anything as you cannot convert it to/from Text.

## Ptr (Data Type)

A pointer to a chunk of memory. You can pass a MemoryBlock object using this data type and it will be treated as a pointer to the memory contained within the MemoryBlock.

Ptr is 4 bytes for 32-bit apps and 8 bytes for 64-bit apps.

### Item Summary

Name	Ptr
Type	Data Type
Inherits	n/a
Implements	n/a
Properties	<a href="#">Boolean</a> <a href="#">Byte</a> <a href="#">CFStringRef</a> <a href="#">Class</a> <a href="#">Color</a> <a href="#">CString</a> <a href="#">Currency</a> <a href="#">Double</a> <a href="#">Int16</a> <a href="#">Int32</a> <a href="#">Int64</a> <a href="#">Int8</a> <a href="#">Integer</a> <a href="#">Object</a> <a href="#">Ptr</a> <a href="#">Short</a> <a href="#">Single</a> <a href="#">String</a> <a href="#">Text</a> <a href="#">UInt16</a> <a href="#">UInt32</a> <a href="#">UInt64</a> <a href="#">UInt8</a> <a href="#">UInteger</a> <a href="#">Variant</a> <a href="#">WindowPtr</a> <a href="#">WString</a>
Targets	All
Platforms	All
Related	<a href="#">Declare</a>

## Notes

You can compare one Ptr to another Ptr or to Nil.

You can convert the value referenced to by the pointer to a specific data type using the properties listed below.

## Properties

`Boolean(offset As Integer = 0) As Boolean`

Converts the value referenced to by the pointer to a Boolean.

---

`Byte(offset As Integer = 0) As Byte`

---

`CFStringRef(offset As Integer = 0) As CFStringRef`

---

`Class(offset As Integer = 0) As Class`

---

`Color(offset As Integer = 0) As Color`

---

`CString(offset As Integer = 0) As CString`

---

`Currency(offset As Integer = 0) As Currency`

---

`Double(offset As Integer = 0) As Double`

---

`Int16(offset As Integer = 0) As Int16`

---

`Int32(offset As Integer = 0) As Int32`

---

`Int64(offset As Integer = 0) As Int64`

---

`Int8(offset As Integer = 0) As Int8`

---

---

Integer(offset As Integer = 0) As Integer

---

Object(offset As Integer = 0) As Object

---

Ptr(offset As Integer = 0) As Ptr

---

Short(offset As Integer = 0) As Ptr

---

Single(offset As Integer = 0) As Single

---

String(offset As Integer = 0) As String

---

Text(offset As Integer = 0) As Text

---

UInt16(offset As Integer = 0) As UInt16

---

UInt32(offset As Integer = 0) As UInt32

---

UInt64(offset As Integer = 0) As UInt64

---

UInt8(offset As Integer = 0) As UInt8

---

UInteger(offset As Integer = 0) As UInteger

---

Variant(offset As Integer = 0) As Variant

---

WindowPtr(offset As Integer = 0) As WindowPtr

---

WString(offset As Integer = 0) As WString

# Single

A Single-precision floating-point number. The default value of a Single is 0.0.

## Data Type Summary

Name	Single
Type	Data Type
Methods	<a href="#">ToText</a>
Shared Methods	<a href="#">FromText</a> <a href="#">Parse</a>
Targets	All
Platforms	All
Related	<a href="#">Currency</a> <a href="#">Double</a>

## Notes

Single is an IEEE single-precision, floating-point value. This means it is speedy but has some limitations in the type of values it can contain. For more information, refer to the wikipedia page about [floating point](#).

In most situations you should use Currency when dealing with monetary values.

## Methods

**ToText**(Extends value As Single, Optional locale As [Locale](#), format As Text = "") As Text

Converts a Double value to a Text value using the optional locale and format.

### Notes

Refer to [Unicode Number Format Patterns](#) for a list of formats.

### Sample Code

Convert single values to text:

```
Dim s As Single = 123.45
```

```
Dim t As Text
```

```
t = s.ToText
```

```
t = s.ToText(Xojo.Core.Locale.Current)
```

```
Dim n As Single = 1239.4567
Dim t As Text = n.ToText(Xojo.Core.Locale.Current, "#,###.##") ' t = 1,239.46
```

## Shared Methods

### FromText(theText As Text, Optional locale As Locale) As Single

Converts a Text value that containing a number that can be represented as a single to a Single.

#### Sample Code

Convert text values to single values:

```
Dim userValue As Text
userValue = "123.45"
```

```
Dim s As Single
s = Single.FromText(userValue)
```

---

### Parse(theText As Text, Optional locale As Locale) As Single

Converts theText to a Single value.

#### Notes

Numbers are converted only if they are found at the beginning of the text. Any numbers that follow a non-numeric value are ignored. Empty text returns 0.

#### Sample Code

```
Dim s As Single
s = Single.Parse("123ABC")
' s = 123
```

## Structure (Data Type)

A structure is a compound value type. It consists of a series of fields that are grouped together as a single block. You control the size and order of the fields. A structure can provide a convenient alternative to a [MemoryBlock](#).

You typically only use structures when you have very specific memory or performance requirements or when you need to interface with an outside API that requires a structure. In most cases, you will want to use Classes with appropriate properties for your data management.

### Data Type Summary

Name	Structure
Type	Data Type
Inherits	n/a
Implements	n/a
Methods	<a href="#">ByteValue</a>
Targets	All
Platforms	All
Related	<a href="#">MemoryBlock</a> <a href="#">Auto</a>

### Notes

Once you have defined a structure, you can use it in almost any context where you would use any other data type. Use the dot syntax to access the fields of the structure.

### Methods

`ByteValue(littleEndian As Boolean, Assigns value() As Byte)`

`ByteValue(littleEndian As Boolean) As Byte()`

Gets or sets the structure data using a byte array.

# WString (Data Type)

A null-terminated UTF-16 String. This is typically used on Microsoft Windows.

## Item Summary

Name	WString
Type	Data Type
Inherits	n/a
Implements	n/a
Targets	All
Platforms	All
Related	<a href="#">Declare MemoryBlock</a>

## Notes

You can assign a standard String to it and it will be converted to UTF-16 and terminated automatically.

WString implicitly converts to String when assigned to a String variable.

# Literals

These syntax elements are used to define literal values.

<b>""</b>	A Text or String literal.
<b>&amp;b</b>	A binary literal.
<b>&amp;c</b>	A color literal.
<b>&amp;h</b>	A hexadecimal literal.
<b>&amp;o</b>	An octal literal.
<b>&amp;u</b>	An unicode literal.

## Syntax Elements

""

Represents a Text or String literal.

### Notes

To denote empty text, use two double quotes with no spaces.

An empty text value is not equivalent to Nil. Nil represents a non-existing object.

To include a quote within the text, enter two consecutive double quotes.

The ampersand character (&) is often interpreted as a keyboard shortcut when used for user interface labels, such as for buttons, menus, label controls. If you want to display an & in these situation, simply use two of them together, such as "&&".

### Sample Code

```
Dim embeddedDoubleQuote As Text = "He said ""yes"", believe it or not!"
```

```
Dim emptyText As Text = ""
```

```
Dim name As Text = "Bob Roberts"
```

## &b

Represents binary literals.

### Notes

Binary literals are stored in Integer data types as integer values.

## Sample Code

```
Dim value As Integer
value = &b101 ' value = 5
```

## &cRRGGBBAA

Represents a Color literal.

### Notes

Specify each portion of the color as hexadecimal.

RR	Red component of color as hexadecimal. The range is 00 to FF.
GG	Green component of color as hexadecimal. The range is 00 to FF.
BB	Blue component of color as hexadecimal. The range is 00 to FF.
AA	Optional Alpha channel (transparency) in the color. The range is 00 (opaque) to FF (fully transparent).

Using this literal is equivalent to using the Color.RGB method, except the RGB method uses decimal values rather than hexadecimal.

## Sample Code

```
Dim red As Color = &cFF000000
Dim blue As Color = &c0000FF10 ' mostly transparent</code>
```

## &h

Represents hexadecimal literals.

### Notes

Hexadecimal literals are stored in Integer data types as integer values.

## Sample Code

```
Dim hex As Integer
hex = &hff
```

## &o

Represents octal literals.

## Notes

Octal literals are stored in Integer data types as integer values.

## Sample Code

```
Dim oct As Integer  
oct = &o755
```

---

## &u

Used to create a Unicode codepoint (hex value) as Text (or String) constants.

## Notes

To create a text value using an integer Unicode codepoint, use `Text.FromUnicodeCodepoint`.

Also refer to:

- [Unicode Codepoint Chart](#)

## Sample Code

```
Const kTab = &u9 ' the tab character
```

# Operators

This section describes the operators in the Xojo Programming Language.

<u>Addition (+)</u>	<u>AddressOf</u>	<u>And</u>	<u>CType</u>
<u>Division (/)</u>	<u>Division, Integer (\)</u>	<u>Equals (=)</u>	<u>Exponentiation (^)</u>
<u>If</u>	<u>Greater Than (&gt;, &gt;=)</u>	<u>Is</u>	<u>IsA</u>
<u>Less Than (&lt;, &lt;=)</u>	<u>Mod</u>	<u>Multiplication (*)</u>	<u>Negation</u>
<u>New</u>	<u>Not</u>	<u>Not Equals (&lt;&gt;)</u>	<u>Or</u>
<u>Subtraction (-)</u>	<u>Xor</u>	<u>WeakAddressOf</u>	

## Operator Precedence

When there are multiple operators in an expressions, they are evaluated in order of precedence. For example, consider this mathematical expression:

```
5 + 2 * 3
```

which has both multiplication and addition operators. Operator precedence dictates that multiplication is done before addition, so the above expression evaluates to 11.

You can always use parentheses to force a particular evaluation order since parentheses are evaluated first. For example, if you want to do the "5+2" before the multiplication, you would write the expression like this:

```
(5 + 2) * 3
```

This table lists operators in order from highest precedence to lowest:

<b>Operator</b>	<b>Description</b>
.	Dot operator for accessing class members.
<u>AddressOf</u> , <u>WeakAddressOf</u>	
<u>IsA</u>	
<u>Exponentiation (^)</u>	
<u>Negation (-)</u>	
<u>Not</u>	
<u>Multiplication (*)</u> , <u>Division (/)</u> , <u>Integer Division (\)</u> , <u>Mod</u>	
<u>Subtraction (-)</u> , <u>Addition (+)</u>	

Operator	Description
<u>Equals (=), Greater Than (&gt;), Less Than (&lt;), Greater Than Equals (&gt;=), Less Than Equals (&lt;=), Not Equals (&lt;&gt;)</u>	
<u>And</u>	
<u>Or, Xor</u>	
<u>Pair (:)</u>	

If there is more than one of the same operator in an expression, the precedence goes from left to right in the table.

All operators are left-associative except for Pair (:) and Exponentiation (^). Left association means that an expression such as "True Or False Or True" evaluates as if it were written like "(True Or False) Or True". Conversely, right association means that an expression such as "2^3^2" evaluates as "2^(3^2)".

# Addition (+) (Operator)

Adds two numbers or concatenates two Text values.

## Operator Summary

Name	Multiplication
Type	Operator
Token	+
Targets	All
Platforms	All
Related	<a href="#">Multiplication</a> , <a href="#">Subtraction (-)</a>

## Syntax

```
result = expression1 + expression2
```

## Parameters

<i>result</i>	The sum of <i>expression1</i> and <i>expression2</i> .
<i>expression1</i>	Any numeric expression or Text value.
<i>expression2</i>	Any numeric expression or Text value.

## Notes

You can use the `Operator_Add` method to define this operator for classes.

## Sample Code

```
Dim sum As Integer  
sum = 1 + 2 + 3 // sum = 6
```

```
Dim name As String  
name = "Bob " + "Roberts" // name = "Bob Roberts"
```

# AddressOf (Operator)

Gets the address of a method (as a delegate). Can be used with [AddHandler](#), [Declares](#) and other advanced commands.

## Operator Summary

Name	AddressOf
Type	Operator
Implements	n/a
Targets	All
Platforms	All
Related	<a href="#">AddHandler</a> , Delegate

## Syntax

```
delegate = AddressOf methodName
```

## Parameters

<i>methodName</i>	The name of the method.
<i>delegate</i>	The delegate that references <i>methodName</i> .

# And (Operator)

Performs a logical And comparison of two boolean expressions or a bitwise comparison of two integers.

## Operator Summary

Name	And
Type	Operator
Targets	All
Platforms	All
Related	Or

## Syntax

```
result = expression1 And expression2
```

```
result = integer1 And integer2
```

## Parameters

<i>result</i>	The Boolean result if <i>expression1</i> and <i>expression2</i> are Boolean values. The Integer result if <i>integer1</i> and <i>integer2</i> are Integer values.
<i>expression1</i> <i>expression2</i>	Any valid boolean expression.
<i>integer1</i> <i>integer2</i>	Any valid Integer.

## Notes

### Boolean Comparison

The most common use of And is to compare to boolean expressions. Refer to the truth table below to see the results from a comparison of boolean expressions

Expression1	Expression2	And	Or	Xor
True	True	True	True	False
True	False	False	True	True
False	True	False	True	True
False	False	False	False	False

## Bitwise Comparison

If you pass two integers, And returns an integer that is the result of comparing each bit of the two integers and assigning 1 to the bit position in the integer returned if both bits in the same position in the integers passed are 1. Otherwise, 0 is assigned to the bit position.

The following table gives the results for bitwise operators.

Integer1	Integer2	And	Or	Xor
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

The Operator\_Add method can be used to define the And operator for classes.

## Use Within If Statements

When used within an If..Then statement, once the first expression results to False, none of the subsequent expressions are evaluated. In this example, the IsValid method is not called because the first expression is False:

```
Dim b As Boolean = False
If b And IsValid("value") Then // IsValid does not get called
    ...
End If
```

## Sample Code

```
Dim a As Boolean
Dim b As Boolean
Dim c As Boolean // defaults to False
Dim d As Boolean // defaults to False

a = True
b = True
d = a And b // Returns True
d = a And c // Returns False
```

# CType (Operator)

Explicitly converts a value of one data type to another data type (only for data types that can already be implicitly converted).

## Operator Summary

Name	CType
Type	Operator
Targets	All
Platforms	All
Related	<a href="#">Double.FromText</a> <a href="#">Integer.FromText</a> <a href="#">Double.ToText</a> <a href="#">Integer.ToText</a>

## Syntax

```
result = CType(value, dataType)
```

## Parameters

<i>result</i>	The pass <i>value</i> converted to the specified <i>dataType</i> .
<i>value</i>	The value to convert to the specified <i>dataType</i> . This must be a value that can be implicitly converted to <i>dataType</i> .
<i>dataType</i>	A built-in data type. This must be a datatype to which <i>value</i> can be implicitly converted.

## Notes

The CType operator is the explicit version of type conversion. Implicit conversion is available via the assignment (=) operator.

This means you can only use CType with types that you can implicitly convert between. For example, you can use CType with Integer and Double because you can implicitly assign a Double to an Integer. However, you cannot use CType with a Double and a Text because you cannot implicitly convert a Double to a Text. You have to instead do that explicitly using [Double.FromText](#).

CType does not necessarily preserve the value across the types. It converts the passed value source type to the destination data type. In that regard, it is identical to implicit conversion. If you are using CType to convert a real number data type (e.g., [Single](#), [Double](#), [Currency](#)) to an [Integer](#) data type, it truncates the value rather than rounds it. Sign extension is preserved when converting from a signed data type to another signed data type. Conversion

between signed and unsigned types (either way) preserves the bits but not the value.

## Sample Code

```
Dim s As Single
Dim i As Integer

i = 5
s = i // implicit conversion
s = CType(i, Single) // explicit conversion
```

```
Dim d As Double
Dim i As Integer

d = 566.75
i = CType(d, Integer) // returns 566
i = d // returns 567
```

# Division (/) (Operator)

Calculates the floating-point division between two numbers.

## Operator Summary

Name	Division
Type	Operator
Token	/
Targets	All
Platforms	All
Related	<a href="#">Integer Division</a> , <a href="#">Mod</a> , <a href="#">Multiplication</a>

## Syntax

```
result = expression1 / expression2
```

### Parameters

<i>result</i>	The floating-point division of <i>expression1</i> and <i>expression2</i> .
<i>expression1</i>	Any numeric expression.
<i>expression2</i>	Any numeric expression.

## Notes

Use this operator when you require division that considers the decimal portion of the expressions.

Regardless of the datatypes of *expression1* and *expression2*, both operands are coerced to [Doubles](#) and the result is a Double. If you divide by zero, the result is positive or negative infinity, except if the numerator is also zero. In that case, the result is NaN. For that reason, you should always check to see whether the denominator is zero before dividing.

Use the `\` operator for integer division or the [Mod](#) operator to get the remainder of the division.

You can use the `Operator_Multiple` method to define this operator for classes.

## Sample Code

```
Dim x As Double
x = 50.56 / 30.34 // x = 1.6664469
```

# Integer Division (\) (Operator)

Calculates the integer division between two numbers.

## Operator Summary

Name	Integer Division
Type	Operator
Token	\
Targets	All
Platforms	All
Related	<a href="#">Division</a> , <a href="#">Mod</a> , <a href="#">Multiplication</a>

## Syntax

```
result = expression1 \ expression2
```

## Parameters

<i>result</i>	The integer division of <i>expression1</i> and <i>expression2</i> .
<i>expression1</i>	Any numeric expression.
<i>expression2</i>	Any numeric expression.

## Notes

Use this operator when you require division that does not consider the decimal portion of the expressions. The result may differ from systems that round the expressions prior to the division.

If *expression1* and *expression2* are not integers, they are coerced to Int32s. The result is an Integer. If you divide by zero, the result is undefined even if the numerator is also zero. The only expected behavior is the application will not crash. For that reason, you should always check to see whether the denominator is zero before dividing.

When converting a floating point number to an integer, there is always the chance of data loss. The same is true when a floating-point value holds a sentinel such as infinity or NaN. Integers do not reserve space for sentinel values, so that information is lost. Converting a floating-point number that represents a sentinel to an Integer yields undefined results.

Use the `/` operator for [floating point division](#) and the `Mod` operator to obtain the remainder of the division.

## Sample Code

```
Dim x As Integer
x = 50.56 \ 30.34 // x = 1
x = 49.56 \ 25.34 // x = 1
```

# Equals (Operator)

Determines whether one expression is equal to another or to assign a value to an identifier. Text comparisons are case-insensitive.

## Operator Summary

Name	Equals
Type	Operator
Token	=
Targets	All
Platforms	All
Related	

## Syntax

```
result = expression1 = expression2
```

### Parameters

<i>result</i>	True if <i>expression1</i> is equal to <i>expression2</i> .
<i>expression1</i> <i>expression2</i>	Any expression, but the data types must match.

```
result = value
```

### Parameters

<i>result</i>	An identifier (variable, property, etc.)
<i>value</i>	Any valid Integer.

## Notes

The data types of *expression1* and *expression2* must match. You can make comparisons between objects of any data type and between objects. If you compare objects, their references are compared, not their contents. For example, when you compare two `FolderItems`, this operator determines whether they have the same reference, not whether they point to the same file.

Use the [Text.Compare](#) method to do case-sensitive string comparisons.

## Sample Code

```
Dim t1 As Text = "Hello"  
Dim t2 As Text = "There"  
  
If t1 = t2 Then  
    // t1 is equal to t2  
Else  
    // t1 is not equal to t2  
End If
```

# Not Equals (Operator)

Determines whether one expression is not equal to another. Text comparisons are case-insensitive.

## Operator Summary

Name	Not Equals
Type	Operator
Token	<>
Targets	All
Platforms	All
Related	<a href="#">Equals (=)</a>

## Syntax

```
result = expression1 <> expression2
```

### Parameters

<i>result</i>	True if <i>expression1</i> is not equal to <i>expression2</i> .
<i>expression1</i> <i>expression2</i>	Any expression, but the data types must match.

## Notes

The data types of *expression1* and *expression2* must match. You can make comparisons between objects of any data type and between objects. If you compare objects, their references are compared, not their contents. For example, when you compare two `FolderItems`, this operator determines whether they have the same reference, not whether they point to the same file.

You can use Not Equals (<>) with objects to test if they are Nil.

Use the [Text.Compare](#) method to do case-sensitive string comparisons.

Use the `Operator_Compare` method to do comparisons for classes.

## Sample Code

```
Dim t1 As Text = "Hello"
Dim t2 As Text = "There"
```

```
If t1 <> t2 Then
    // t1 is not equal to t2
Else
    // t1 is equal to t2
End If
```

```
Dim d As New Dictionary
If d <> Nil Then
    d.Value("Test") = "Hello"
End If
```

# Exponentiation (^) (Operator)

Calculates exponentiation.

## Operator Summary

Name	Exponentiation
Type	Operator
Token	^
Targets	All
Platforms	All
Related	

## Syntax

```
result = expression1 ^ expression2
```

## Parameters

<i>result</i>	The result of raising expression1 to the power of expression2.
<i>expression1</i>	Any numeric expression.
<i>expression2</i>	Any numeric expression.

## Notes

You can use the `Operator_Power` method to define this operator for classes.

## Sample Code

```
Dim i As Double  
i = 2 ^ 2 // i = 4  
i = (2 + 3) ^ (4 - 2) // i = 25  
i = 3.75 ^ 3.5 // i = 102.12
```

# If (Operator)

A trinary operator that evaluates an expression and returns results when it is True or False.

## Operator Summary

Name	If
Type	Operator
Targets	All
Platforms	All
Related	<a href="#">If keyword</a>

## Syntax

```
result = If(expression, resultWhenTrue, resultWhenFalse)
```

### Parameters

<i>result</i>	<i>resultWhenTrue</i> if <i>expression</i> is True, <i>resultWhenFalse</i> when <i>expression</i> is False.
<i>resultWhenTrue</i>	The <i>result</i> when <i>expression</i> is True.
<i>resultWhenFalse</i>	The <i>result</i> when <i>expression</i> is False.

## Notes

The return type is the common type between the *resultWhenTrue* and *resultWhenFalse*. For example, if an Int8 and an Int32 are used as result values, the result type will actually be Int32. Having no common type results in a Type Mismatch compile error.

This operator uses the same evaluation rules as a normal If..Then statement. Only the result that is needed gets evaluated.

## Sample Code

```
Dim myInteger As Integer = 41
Dim result As Text
result = If(myInteger > 40, "Big number", "Small number")
```

## Greater Than (>, >=) (Operator)

Determines whether one alphabetic or other expression is larger (or equal to and larger) than another. Text comparisons are case-insensitive.

### Operator Summary

Name	Greater Than
Type	Operator
Tokens	>, >=
Targets	All
Platforms	All
Related	<a href="#">Equals (=)</a>

### Syntax

```
result = expression1 > expression2
```

#### Parameters

<i>result</i>	True if <i>expression1</i> is greater than <i>expression2</i> .
<i>expression1</i> <i>expression2</i>	Any expression, but the data types must match.

```
result = expression1 >= expression2
```

#### Parameters

<i>result</i>	True if <i>expression1</i> is greater than or equal to <i>expression2</i> .
<i>expression1</i> <i>expression2</i>	Any expression, but the data types must match.

### Notes

Text is "greater than" another Text if it is last when the two strings are sorted alphabetically. Use [Text.Compare](#) to do a case-sensitive comparisons.

Use the `Operator_Compare` method to define comparisons for classes.

### Sample Code

```
Dim num1 As Integer = 100
```

```
Dim num2 As Integer = 250

If num1 > num2 Then
    // This block is skipped
Else
    // This block is executed
End If

num2 = 100
If num1 >= num2 Then
    // This block is executed
Else
    // This block is skipped
End If

Dim t1 As Text = "Hello"
Dim t2 As Text = "World"

If t1 > t2 Then
    // This block is skipped
Else
    // This block is executed
End If
```

# Is (Operator)

Compares two object references to determine whether they both refer to the same object.

## Operator Summary

Name	Is
Type	Operator
Targets	All
Platforms	All
Related	<a href="#">IsA</a>

## Syntax

```
result = object1 Is object2
```

### Parameters

<i>result</i>	A Boolean value that is True when <i>object1</i> refers to <i>object2</i> .
<i>object1</i>	Any object reference.
<i>object2</i>	Any object reference.

## Notes

The Is operator returns True if object1 and object2 actually refer to the same object. It checks identity, not contents, so it is not affected by the presence of a comparison operator.

## Sample Code

```
Dim d1 As New Dictionary  
Dim d2 As Dictionary
```

```
d2 = d1
```

```
If d1 Is d2 Then  
    // d1 and d2 are the same reference  
Else  
    // d1 and d2 are not the same reference  
End If
```

# IsA (Operator)

Compares a class instance (object) with a class type or interface.

## Operator Summary

Name	IsA
Type	Operator
Targets	All
Platforms	All
Related	

## Syntax

```
result = object IsA class
```

### Parameters

<i>result</i>	A Boolean value that is True when <i>object</i> is an instance of <i>class</i> , a subclass of <i>class</i> or implements the <i>class</i> interface.
<i>object</i>	Any object reference.
<i>class</i>	Any class or interface name.

## Notes

If *object* is of type *class*, then IsA returns True; if not, it returns False. If *object* is Nil, then IsA returns False. The IsA operator returns True for the object's own class as well as the super class from which that class was derived, all the way to the Object class. IsA will report that all classes ultimately derive from Object.

Remember that object must always be a class instance, not just a variable with the class of the type. For example, this code always returns False because

```
Dim d As Dictionary

// Returns False because d is not an instance
If d IsA Dictionary Then
    ...
Else
    ...
End If</code>
```

Instead you create an instance like this:

```
Dim d As New Dictionary

// Returns True because d an instance of Dictionary
If d IsA Dictionary Then
    ...
Else
    ...
End If
```

### Casting

The IsA operator is often used test an object's type before casting it. This code loops through all the controls on a Window and when it finds a TextField, it sets its Text property to "Found You!".

```
For i As Integer = 0 To Self.ControlCount-1
    // Check if the control is a TextField
    If Self.Control(i) IsA TextField Then
        // Cast the control to a TextField in order to access the Text property
        TextField(Self.Control(i)).Text = "Found You!"
    End If
Next
```

### Interfaces

The IsA operator can also be used with class interfaces. It reports True is the object implements the class interface.

## Sample Code

```
// Count all the Labels on a Window
Dim labelCount As Integer
For i As Integer = 0 To Self.ControlCount-1
    If Self.Control(i) IsA Label Then labelCount = labelCount + 1
Next
```

## Less Than (<, <=) (Operator)

Determines whether one alphabetic or other expression is larger (or equal to and larger) than another. Text comparisons are case-insensitive.

### Operator Summary

Name	Less Than
Type	Operator
Tokens	<, <=
Targets	All
Platforms	All
Related	<a href="#">Equals (=)</a> , <a href="#">Greater Than (&gt;, &gt;=)</a>

### Syntax

```
result = expression1 < expression2
```

#### Parameters

<i>result</i>	True if <i>expression1</i> is less than <i>expression2</i> .
<i>expression1</i> <i>expression2</i>	Any expression, but the data types must match.

```
result = expression1 <= expression2
```

#### Parameters

<i>result</i>	True if <i>expression1</i> is less than or equal to <i>expression2</i> .
<i>expression1</i> <i>expression2</i>	Any expression, but the data types must match.

### Notes

Text is "less than" another Text if it is first when the two strings are sorted alphabetically. Use [Text.Compare](#) to do a case-sensitive comparisons.

Use the `Operator_Compare` method to define comparisons for classes.

### Sample Code

```
Dim num1 As Integer = 100
```

```
Dim num2 As Integer = 250

If num1 < num2 Then
    // This block is executed
Else
    // This block is skipped
End If

num2 = 100
If num1 <= num2 Then
    // This block is executed
Else
    // This block is skipped
End If

Dim t1 As Text = "Hello"
Dim t2 As Text = "World"

If t1 < t2 Then
    // This block is executed
Else
    // This block is skipped
End If
```

# Mod (Operator)

Calculates the remainder of the division of two numbers.

## Operator Summary

Name	Mod
Type	Operator
Targets	All
Platforms	All
Related	<a href="#">Division (/)</a> , <a href="#">Integer Division (\)</a>

## Syntax

```
result = number1 Mod number2
```

### Parameters

<i>result</i>	The remainder (as an Integer) of <i>number1</i> divided by <i>number2</i> .
<i>number1</i>	Any number.
<i>number2</i>	Any not null (non-zero) number.

## Notes

The Mod operator divides *number1* by *number2* and returns the remainder as the result.

### Type Rules

<i>number1</i> or <i>number2</i> are floating point	both are converted to Int32
<i>number1</i> or <i>number</i> are Int64	both are converted to Int64
<i>number2</i> is 0 (zero)	undefined result



When converting a floating-point number to an Integer, there is always the possibility of data loss. The same is true when a floating-point value holds a sentinel such as infinity (INF) or not a number (NaN). Integers do not reserve space for sentinel values, so that information is lost. Converting a floating-point number that represents a sentinel to an Integer yields undefined results.

## Sample Code

```
Dim r As Integer
r = 5 Mod 2 // returns 1
r = 5 Mod 1.99999 // returns 0
```

```
r = -10 Mod 3 // Returns -1
r = -10 Mod -3 // Returns -1
r = 10 Mod 3 // Returns 1
```

```
r = 10 Mod 3 // Returns 1
r = 2 Mod 4 // Returns 2
r = 9.3 Mod 2.75 // Returns 1
r = 4.5 Mod 1 // Returns 0
r = 25 Mod 5 // Returns 0
```

Determine if an Integer is odd or even:

```
If myInteger Mod 2 = 0 Then
    // myInteger is even
Else
    // myInteger is odd
End If
```

# Multiplication (\*) (Operator)

Multiplies two numbers.

## Operator Summary

Name	Multiplication
Type	Operator
Token	*
Targets	All
Platforms	All
Related	<a href="#">Division (/)</a>

## Syntax

```
result = expression1 * expression2
```

## Parameters

<i>result</i>	The product of <i>expression1</i> and <i>expression2</i> .
<i>expression1</i>	Any numeric expression.
<i>expression2</i>	Any numeric expression.

## Notes

You can use the `Operator_Multiple` method to define this operator for classes.

## Sample Code

```
Dim x As Integer  
x = 50 * 30 // x = 1500
```

```
Dim y As Integer = 42  
x = y * 10 // x = 420
```

# New (Operator)

Creates instances of classes (instantiation).

## Operator Summary

Name	New
Type	Operator
Targets	All
Platforms	All
Related	

## Syntax

```
objectInstance = New className
```

```
control = New controlSetName
```

The [Dim](#) and [Static](#) statements can also use New to declare a variable and assign it a value in one step.

### Parameters

<i>objectInstance</i>	A variable of type <i>className</i> .
<i>className</i>	Any class available to your project.

## Notes

If the class has a Constructor defined, it is called when you instantiate it using New.

### Dynamically Adding Desktop Controls at Run-Time

In desktop projects, you can dynamically add new controls to a Window or ContainerControl if the control is part of a Control Set by using New. For example, if TextField1 is a TextField on a Window and assigned to a Control Set, you can add another TextField to the Window like this:

```
Dim tf As TextField1
tf = New TextField1
```

To dynamically add web controls at run-time, you should put the web controls on a WebContainer and then add the WebContainer at run-time.

To dynamically add iOS controls at run-time, create new instances of the control and add it to the View using [View.AddControl](#).

## Sample Code

```
// Create a Dictionary  
Dim d As New Dictionary  
d.Value("Name") = "Red Sox"
```

# Not (Operator)

Performs a logical negation of a boolean expression or a bitwise Not operation on an integer expression.

## Operator Summary

Name	Not
Type	Operator
Targets	All
Platforms	All
Related	

## Syntax

```
result = Not expression
.....
result = Not integerExpression
.....
```

## Parameters

<i>result</i>	The Boolean result if <i>expression</i> is a Boolean value. The Integer result if <i>integer</i> is an Integer value.
<i>expression</i>	Any valid boolean expression.
<i>integerExpression</i>	Any valid Integer expression.

## Notes

### Boolean Comparison

The most common use of Not is to negate boolean expressions. Refer to the truth table below:

Expression	Not
True	False
False	True

### Bitwise Comparison

If you pass an integer, Not returns an integer that is the result of inverting the bit values of the integer expression.

The following table gives the results for bitwise operators.

---

<b>IntegerExpression</b>	<b>Not</b>
0	1
1	0

The Operator\_Not method can be used to define the Not operator for classes.

## Sample Code

```
Dim a As Boolean
Dim b As Boolean
Dim c As Boolean // defaults to False
```

```
a = True
```

```
b = Not a // b = False
b = Not c // b = True
```

# Operator Overloading

You can define the various operators in the Xojo languages for your own classes and subclasses by overloading these specific methods.

## Summary

Name	Operator Overloading
Methods	<a href="#">Operator_Add</a> , <a href="#">Operator_And</a> , <a href="#">Operator_Compare</a> , <a href="#">Operator_Convert</a> , <a href="#">Operator_Divide</a> , <a href="#">Operator_IntegerDivide</a> , <a href="#">Operator_Lookup</a> , <a href="#">Operator_Modulo</a> , <a href="#">Operator_Multiply</a> , <a href="#">Operator_Negate</a> , <a href="#">Operator_Not</a> , <a href="#">Operator_Or</a> , <a href="#">Operator_Power</a> , <a href="#">Operator_Redim</a> , <a href="#">Operator_Subscript</a> , <a href="#">Operator_Subtract</a> , <a href="#">Operator_XOr</a>
Targets	All
Platforms	All
Related	

## Notes

Several of the example below use a Vector class, which is defined as follows:

```
Class Vector
  Property x As Integer
  Property y As Integer
End Class
```

## Methods

**Operator\_Add(rightSide As Type1) As Type2**

**Operator\_AddRight(leftSide As Type1) As Type2**

Allows the class to support the Addition (+) operator.

### Notes

The Self instance is the left operand. Use `Operator_AddRight` to have the Self instance be the right operand.

### Sample Code

```
// Vector addition
Function Operator_Add(rightSide As Vector) As Vector
  Dim newVector As New Vector
  newVector.x = Self.x + rightSide.x
  newVector.y = Self.y + rightSide.y
```

```
Return newVector  
End Function
```

---

## Operator\_And(rightSide As Type1) As Type2

## Operator\_AndRight(leftSide As Type1) As Type2

### Notes

The Self instance is the left operand. Use Operator\_AndRight to have the Self instance be the right operand.

### Sample Code

```
// Class MyClass has a property Test As Boolean  
Function Operator_And(rightSide As MyClass) As Boolean  
Return Self.Test And rightSide.Test  
End Function
```

---

## Operator\_Compare(rightSide As Type1) As Integer

Allows the class to support the comparison operators.

### Notes

Works with Equals (=), Greater Than (>, >=), Less Than (<, <=) and Not Equal (<>).

### Returns

0 means Self is equal to *rightSide*.

< 0 means Self is less than *rightSide*.

> 0 means Self is greater than *rightSide*.

### Sample Code

```
// Compares two Vectors  
Function Operator_Compare(rightSide As Vector) As Integer  
Dim a, b As Integer  
a = Self.x ^ 2 + Self.y ^ 2  
b = rightSide.x ^ 2 + rightSide.y ^ 2  
  
If a > b Then Return 1  
If a = b Then Return 0  
If a < b Then Return -1  
End Function
```

---

## Operator\_Convert(rightSide As Type1)

## Operator\_Convert As Type2

Allows the class to be converted from one type to another or allows another type to be converted to the class.

### Notes

When you use the convert-to form of the operator (the one with a parameter), the compiler needs to create a new object implicitly so that you can copy data into it. When you use a convert-to operator, the compiler needs to create a new instance of the item it located the `Operator_Convert` method on, and then it calls that method to complete the conversion. When it creates the new instance of the item, it does not call any constructors for it. The `Operator_Convert` method takes the place of the constructor.

When you use a convert-from operator (the one without a parameter), the compiler doesn't have to do any extra work. Whenever an implicit (or explicit) conversion needs to take place, the compiler looks for the convert-from operator, and if it finds one, it calls it. The function itself already does all of the conversion work.

Constructors are not called when doing a convert-to operation. That's simply something to be aware of. If you're doing required work in your constructor, you'll need to perform that work yourself in the conversion. This generally won't need to happen, since most constructors simply initialize things to sensible values, and conversion operations would do exactly the same thing. But if you are doing some sort of Declare work, it may still be necessary. For instance, let's say that you were doing some sort of COM work on Windows. Your `Operator_Convert` method had better call `CoInitialize` just to be on the safe side.

### Sample Code

```
// Person is a class with a property FullName As Text
Dim p1 As New Person
p1.FullName = "Bob Roberts"

// You can define an Operator_Convert method on Person to allow
// the name to be assigned directly:
Sub Operator_Convert(name As Text)
    Self.FullName = name
End Sub

// Now you can write code like this, </code>which converts a Text value to an instance
// of Person:
Dim p2 As Person
p2 = "Bob Roberts" // p2.FullName = "Bob Roberts"

// You can also define an Operator_Convert method on Person to convert Person to a Text
Function Operator_Convert As Text
    Return Self.FullName
End If
```

```
// This allows you to write code like this, which converts a Person to a Text value for use by a Label:  
Label1.Text = p2
```

---

## Operator\_Divide

### Operator\_DivideRight

Allows the class to support the division (/) operator.

#### Sample Code

```
// Divide the sum of two vectors  
Function Operator_Divide(rightSide As Vector) As Double  
    Dim a, b As Integer  
    a = Self.x + Self.y  
    b = rightSide.x + rightSide.y  
    Return a / b  
End Function
```

---

## Operator\_IntegerDivide

### Operator\_IntegerDivideRight

Allows the class to support the integer division (\) operator.

#### Sample Code

```
// Integer Divide the sum of two vectors  
Function Operator_Divide(rightSide As Vector) As Double  
    Dim a, b As Integer  
    a = Self.x + Self.y  
    b = rightSide.x + rightSide.y  
    Return a \ b  
End Function
```

---

## Operator\_Lookup(name As String)

### Operator\_Lookup(name As String) As Type

Allows the class to handle dot notation references that are not class members.

#### Notes

If you implement `Operator_Lookup` on a class, you will no longer get compilation errors if you, for example, inadvertently misspell a property name. Instead, the `Operator_Lookup` method is called at runtime and is passed the

name of your incorrectly spelled property.

### Sample Code

```
// Gets values entered as dot notation from a dictionary
// MyClass.AnyName returns what is in
// ValueDict.Value("AnyName")
```

```
Function Operator_Lookup(name As String) As String
    If ValueDict.HasKey(name) Then
        Return ValueDict.Value(name)
    Else
        Return ""
    End If
End Function
```

---

## Operator\_Modulo

### Operator\_ModuloRight

Allows the class to handle the Mod operator.

### Sample Code

```
// The remainder of the division of the sum of two vectors
Function Operator_Modulo(rightSide As Vector) As Integer
    Dim a, b As Integer
    a = Self.x + Self.y
    b = rightSide.x + rightSide.y
    Return a Mod b
End If
```

---

## Operator\_Multiply

### Operator\_MultiplyRight

Allows the class to handle the multiplication (\*) operator.

### Sample Code

```
// Multiplies the sum of two vectors
Function Operator_Multiply(rightSide As Vector) As Integer
    Dim a, b As Integer
    a = Self.x + Self.y
    b = rightSide.x + rightSide.y
    Return a * b
```

End If

---

## Operator\_Negate

Allows the class to handle the negation (-) operator.

Sample Code

```
// Negates a vector as the negative of its square length
Function Operator_Negate As Integer
    Dim a As Integer
    a = Self.X ^ 2 + Self.y ^ 2
    Return -a
End If
```

---

## Operator\_Not

Allows the class to handle the Not operator.

Sample Code

```
// Class MyClass has a property Test As Boolean
Function Operator_Not As Boolean
    Return Not Self.Test
End Function
```

---

## Operator\_Or

### Operator\_OrRight

Allows the class to handle the Or operator.

Sample Code

```
// Class MyClass has a property Test As Boolean
Function Operator_Or(rightSide As MyClass) As Boolean
    Return Self.Test Or rightSide.Test
End Function
```

---

## Operator\_Power

### Operator\_PowerRight

Allows the class to handle the exponentiation (^) operator.

## Sample Code

```
// Raises the sum of the vector to the sum of the power vector
Function Operator_Power(rightSide As Vector) As Integer
    Dim a, b As Integer
    a = Self.x + Self.y
    b = rightSide.x + rightSide.y
    Return a ^ b
End Function
```

---

## Operator\_Redim

Allows the class to support the Redim statement, allowing you to use Redim on objects that are not arrays.

## Sample Code

```
Class OneBasedArray
    Sub Constructor(bounds As Integer = 0)
        Redim mArray(bounds - 1)
    End Sub

    Function Count() As Integer
        Return mArray.UBound + 1
    End Function

    Sub Operator_Redim(newSize As Integer)
        Redim mArray(newSize - 1)
    End Sub

    Function Operator_Subscript(index As Integer) As Auto
        Return mArray(index - 1)
    End Function

    Sub Operator_Subscript(index As Integer, Assigns a As Auto)
        mArray(index - 1) = a
    End Sub

    Private mArray() As Auto
End Class
```

---

## Operator\_Subscript

Allows the array subscript operator to be used by a class when it is not an array. Refer to [Operator\\_Redim](#) for sample code.

## Operator\_Subtract

### Operator\_SubtractRight

Allows the class to handle the subtraction (-) operator.

#### Sample Code

```
// Subtracts two vectors
Function Operator_Subtract(rightSide As Vector) As Vector
    Dim newVector As New Vector
    newVector.x = Self.x - rightSide.x
    newVector.y = Self.y - rightSide.y
    Return newVector
End Function
```

---

## Operator\_XOr

### Operator\_XOrRight

Allows the class to handle the Xor operator.

#### Sample Code

```
// Class MyClass has a property Test As Boolean
Function Operator_XOr(rightSide As MyClass) As Boolean
    Return Self.Test Xor rightSide.Test
End Function
```

# Or (Operator)

Performs a logical Or comparison of two boolean expressions or a bitwise comparison of two integers.

## Operator Summary

Name	Or
Type	Operator
Targets	All
Platforms	All
Related	<a href="#">And</a>

## Syntax

```
result = expression1 Or expression2
```

```
result = integer1 Or integer2
```

## Parameters

<i>result</i>	The Boolean result if <i>expression1</i> and <i>expression2</i> are Boolean values. The Integer result if <i>integer1</i> and <i>integer2</i> are Integer values.
<i>expression1</i> <i>expression2</i>	Any valid boolean expression.
<i>integer1</i> <i>integer2</i>	Any valid Integer.

## Notes

### Boolean Comparison

The most common use of Or is to compare to boolean expressions. Refer to the truth table below to see the results from a comparison of boolean expressions

Expression1	Expression2	And	Or	Xor
True	True	True	True	False
True	False	False	True	True
False	True	False	True	True
False	False	False	False	False

## Bitwise Comparison

If you pass two integers, And returns an integer that is the result of comparing each bit of the two integers and assigning 1 to the bit position in the integer returned if both bits in the same position in the integers passed are 1. Otherwise, 0 is assigned to the bit position.

The following table gives the results for bitwise operators.

Integer1	Integer2	And	Or	Xor
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

The Operator\_Or method can be used to define the Or operator for classes.

## Sample Code

```
Dim a As Boolean
Dim b As Boolean
Dim c As Boolean // defaults to False
Dim d As Boolean

a = True
b = False
d = a Or b // Returns True
d = b Or c // Returns False
```

# Subtraction (-) (Operator)

Subtracts two numbers or negates the sign of a number.

## Operator Summary

Name	Subtraction
Type	Operator
Token	-
Targets	All
Platforms	All
Related	<a href="#">Addition (+)</a>

## Syntax

```
result = expression1 - expression2
```

```
result = -expression
```

## Parameters

<i>result</i>	The difference between <i>expression1</i> and <i>expression2</i> .
<i>expression1</i>	Any numeric expression.
<i>expression2</i>	Any numeric expression.
<i>expression</i>	Any numeric expression.

## Notes

You can use the `Operator_Subtract` method to define the subtraction operation for classes.

You can use the `Operator_Negate` method to define the negation operation for classes.

## Sample Code

```
Dim diff As Integer
diff = 50 - 30 // diff = 20
```

```
Dim diff As Integer = 50
diff = -diff // diff = -50
```

```
Dim value As Integer = 100  
diff = value - 60 // diff = 40
```

# Xor (Operator)

Performs a logical Xor comparison of two boolean expressions or a bitwise Xor comparison of two integers.

## Operator Summary

Name	Xor
Type	Operator
Targets	All
Platforms	All
Related	<a href="#">Or</a>

## Syntax

```
result = expression1 Xor expression2
```

```
result = integer1 Xor integer2
```

## Parameters

<i>result</i>	The Boolean result if <i>expression1</i> and <i>expression2</i> are Boolean values. The Integer result if <i>integer1</i> and <i>integer2</i> are Integer values.
<i>expression1</i> <i>expression2</i>	Any valid boolean expression.
<i>integer1</i> <i>integer2</i>	Any valid Integer.

## Notes

### Boolean Comparison

Refer to the truth table below to see the results from a comparison of boolean expressions

Expression1	Expression2	And	Or	Xor
True	True	True	True	False
True	False	False	True	True
False	True	False	True	True
False	False	False	False	False

## Bitwise Comparison

If you pass two integers, And returns an integer that is the result of comparing each bit of the two integers and assigning 1 to the bit position in the integer returned if both bits in the same position in the integers passed are 1. Otherwise, 0 is assigned to the bit position.

The following table gives the results for bitwise operators.

<b>Integer1</b>	<b>Integer2</b>	<b>And</b>	<b>Or</b>	<b>Xor</b>
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

The Operator\_XOr method can be used to define the Xor operator for classes.

## Sample Code

```
Dim i As Integer  
i = 5 Xor 3 // i = 6
```

# WeakAddressOf (Operator)

Gets the address of a method (as a delegate). Can be used with [AddHandler](#), [Declares](#) and other advanced commands. This uses a [WeakRef](#) internally.

## Operator Summary

Name	WeakAddressOf
Type	Operator
Targets	All
Platforms	All
Related	<a href="#">AddressOf</a> , <a href="#">AddHandler</a> , <a href="#">Delegate</a>

## Syntax

```
delegate = WeakAddressOf methodName
```

## Parameters

<i>methodName</i>	The name of the method.
<i>delegate</i>	The delegate that references <i>methodName</i> .

# Pragma Directives

Pragma directives are used to alter standard behavior of the compiler and are specified using the #Pragma command.

Directive	Syntax	Description
BackgroundTasks	<code>#Pragma BackgroundTasks True   False;</code> <code>#Pragma BackgroundTasks On   Off;</code>	Controls auto-yielding for background threads. When set to False/Off, then the compiler does not auto-yield to other threads during loop boundaries. This can speed up processor-intensive operations. This can be toggled within a method. Do not set BackgroundTasks to False in web applications as this will prevent other sessions from running until the background tasks are resumed. This could cause sessions to disconnect or other undesired behavior. <code>#Pragma BackgroundTasks Off;</code>
BoundsChecking	<code>#Pragma BoundsChecking True   False;</code> <code>#Pragma BoundsChecking On   Off;</code>	Controls whether bounds checking is done for array index values. This can be toggled within a method. Setting BoundsChecking False/Off will result in a slight speed improvement, but it is not recommended because it may cause your app to crash. <code>#Pragma BoundsChecking Off;</code>
BreakOnExceptions	<code>#Pragma BreakOnExceptions True   False;</code> <code>#Pragma BreakOnExceptions On   Off;</code>	Controls Project->Break On Exceptions setting in the IDE. When False, the Debugger does not automatically appear when an exception occurs. When True, the Debugger automatically appears when an exception occurs (regardless of whether the exception is handled). You can toggle this setting anywhere within the method. At the end of the method, the setting reverts to how Project->Break On Exceptions is set. <code>#Pragma BreakOnExceptions Off;</code>
DisableBackgroundTasks	<code>#Pragma DisableBackgroundTasks;</code>	The same as setting <code>BackgroundTasks</code> to False/Off.
DisableBoundsChecking	<code>#Pragma DisableBoundsChecking;</code>	The same as setting <code>BoundsChecking</code> to False/Off.
Error	<code>#Pragma Error <i>textMessage</i>;</code>	Used to manually generate compile errors. This can be useful as a reminder of code that you must update. By specifying the Pragma, you will get a compiler error with a message. <code>#Pragma Error "Fix this code!";</code>

Directive	Syntax	Description
NilObjectChecking	<code>#Pragma NilObjectChecking True   False #Pragma NilObjectChecking On   Off</code>	Controls whether objects are checked for Nil before accessing their members. This can be toggled within a method. Setting NilObjectChecking False/Off will result in a slight speed improvement, but it is not recommended because unhandled exceptions can cause your app to crash. <code>#Pragma NilObjectChecking Off</code>
StackOverflowChecking	<code>#Pragma StackOverflowChecking True   False #Pragma StackOverflowChecking On   Off</code>	Controls whether to check for stack overflows. This can be toggled within a method. Setting StackOverflowChecking False/Off will result in a slight speed improvement, but it is not recommended because stack overflows can crash your app. <code>#Pragma StackOverflowChecking Off</code>
Unused	<code>#Pragma Unused <i>variableName</i></code>	Indicates a variable that is not being used to prevent it from appearing in a compiler warning. <code>#Pragma Unused row</code>
Warning	<code>#Pragma Warning <i>textMessage</i></code>	Used to manually display a warning that will only appear when you Analyze Warnings for the project. This is useful to remind you of a code fix. <code>#Pragma Warning "Fix this code!"</code>
X86CallingConvention	<code>#Pragma X86CallingConvention StdCall   cdecl</code>	Specifies the calling convention that a method is compiled with on X86 CPUs. This allows you to write callback functions on Windows, which typically require the StdCall calling convention. <code>#Pragma X86CallingConvention StdCall</code>

## Sample Code

Normally you want Project->Break On Exceptions to be enabled. The BreakOnExceptions Pragma can be a handy way to prevent the debugger from appearing in code where you are expecting (and handle) an exception:

```
#Pragma BreakOnExceptions Off
Try
    TextOutputStream.Create(myFile)
Catch e As IOException
    // Could not create file, probably permissions
    ErrorMessage = "Unable to create file."
End Try
#Pragma BreakOnExceptions On
```

# Reserved Words

These words are reserved by the Xojo Programming Language and may not be used as identifiers for your own variables, objects, event definitions, methods, etc.

In addition, do not use the underscore ("\_") as the first character of an identifier.

## Reserved Words

#Bad  
#Else  
#Elseif  
#If  
#Pragma  
#Tag  
AddHandler  
AddressOf  
Aggregates  
And  
Array  
As  
Assigns  
Attributes  
Break  
ByRef  
ByVal  
Call  
Case  
Catch  
Class  
Const  
Continue  
CType  
Declare  
Delegate  
Dim  
Do  
DownTo  
Each  
Else  
Elseif  
End  
Enum  
Event  
Exception  
Exit  
Extends  
False  
Finally  
For  
Function  
Global  
GoTo  
Handles  
If  
Implements  
In  
Inherits  
Inline68K  
Interface  
Is  
IsA  
Lib  
Loop  
Me  
Mod  
Module  
Namespace  
New  
Next  
Nil  
Not  
Object  
Of  
Optional  
Or  
ParamArray  
Private  
Property  
Protected  
Public  
Raise  
RaiseEvent  
Rect  
ReDim  
Rem  
RemoveHandler  
Return  
Select  
Self  
Shared  
Soft  
Static  
Step  
Structure  
Sub  
Super  
Then  
To  
True  
Try  
Until  
Using  
WeakAddressOf  
Wend  
While  
With  
Xor

# Constructor

A constructor is a special (and optional) method of a class that is called each time an instance of a class is created.

## Syntax

```
Constructor(parameter list)
```

## Notes

When you create a new object, you may want to perform some sort of initialization on the object. The constructor is a mechanism for doing this. A class's constructor is the method that will be called automatically when an instance of the class is created using the [New](#) keyword.

You write a constructor for a custom class by creating a new method for the class and naming it "Constructor". The drop-down list for the Method name field suggests this name (and the names of all other methods that can be overridden, if you are creating a subclass).

When you create a constructor for any subclass, the Code Editor automatically inserts code that calls the constructor for its super class using the [Super](#) keyword. If there is more than one constructor, it inserts calls to all of them. This is because the subclass's constructor overrides its super class's constructor but the new subclass may not initialize itself correctly without a call to the super class's constructor. You can edit the inserted calls in the event that this assumption is incorrect.

## Sample Code

Creates a new point, supply the point coordinates in the Constructor:

```
Dim p As New Xojo.Core.Point(100, 50)
```

---

# Destructor

A destructor is a special (and optional) method of a class that is called just before it is released from memory, which is typically when the class goes out of scope.

## Syntax

Destructor

## Notes

The Destructor method is called automatically, so you would never call it yourself. It takes no parameters and does not return a value.

Destructors are typically used to do any "clean up" that is not done for you automatically. If it exists, a class's destructor is called automatically when an instance of the class is deleted or goes out of scope.

To create the Destructor for a custom class, add a method to the class and name it "Destructor".

Destructors are called when the last reference to an object is removed, even if execution is in a destructor for another object. Note that this means you can cause a stack overflow if your destructor triggers other destructors in a deep recursion. However, such overflow will not happen as long as properties of the object are being cleaned up automatically. So, it is generally preferable to not set properties to Nil in your destructor, but instead let them be cleaned up for you.

Unlike a [Constructor](#) or other methods, the Destructor in a subclass does not override its superclass Destructor. Normally you would not call Super.Destructor in your subclass' Destructor since this will cause the superclass' Destructor to fire twice.

# Enumeration

An enumeration (or enum) is a group of named values (called elements). By default, the elements are numbered consecutively, starting with zero, but you can assign any Integer value.

Although Enumerations only accept Integer constants, the enumeration values cannot be treated as Integers in your code because an Enumeration is a separate type. Likewise, Integer values cannot be assigned to an enumeration. Should you need to get the Integer value of an enumeration, you need to cast it to an integer.

Enumerations are added to classes (and modules) using Insert -> Enumeration. From the Enum Editor, you can add the named values.

In your code, you refer to an enum by its name and refer to its elements by using dot notation. For example:

```
EnumName.EnumElement
```

So why should you use an Enum instead of an Integer constant?

An Integer constant is a great way to refer to a constant value using an name. But since it is equivalent to an Integer, you cannot restrict its usage. With an enum, you can enforce that only the specific set of values are allowed.

## Sample Code

Consider a global Enumeration named **SecurityLevel** that has four elements: **Unauthorized**, **Minimal**, **Maximum**, and **Forced**. Their Integer values range from 0 to 3.

To assign a level of Minimal to a variable of type SecurityLevel:

```
Dim level As SecurityLevel  
level = SecurityLevel.Minimal
```

The level variable can only be assigned Enum elements. You cannot just assign its Integer value:

```
level = 1 ' Compile error
```

You can, however, get the Integer value of an element by casting it to an Integer:

```
Dim levelInt As Int32  
levelInt = Int32(SecurityLevel.Minimal) ' levelInt = 1
```

# Xojo

The Xojo namespace contains all parts of the Xojo framework.

## Namespace Summary

Name	Xojo
Type	Namespace
Namespaces	<a href="#">Core</a> <a href="#">Crypto</a> <a href="#">Data</a> <a href="#">Introspection</a> <a href="#">IO</a> <a href="#">Math</a> <a href="#">Net</a> <a href="#">System</a> <a href="#">Threading</a>
Enumerations	<a href="#">iOSKeyboardTypes</a> <a href="#">iOSLineBreakMode</a> <a href="#">iOSTextAlignment</a>
Methods	<a href="#">IsText</a>
Project Types	All
Platforms	All
Related	<a href="#">Using</a>

## Notes

 iOS projects include the entire Xojo namespace by default (Using Xojo.\*), making it simple to refer to all the classes.

## Enumerations

### iOSKeyboardTypes

Specifies the type of keyboard to use with [iOSTextFields](#) and [iOSTextAreas](#).

Default	Use the default keyboard for the current input method.
ASCIICapable	Use a keyboard that displays standard ASCII characters.
NumbersAndPunctuation	Use the numbers and punctuation keyboard.

URL	Use a keyboard optimized for URL entry. This type features “.”, “/”, and “.com” prominently.
NumberPad	Use a numeric keypad designed for PIN entry. This type features the numbers 0 through 9 prominently. This keyboard type does not support auto-capitalization.
PhonePad	Use a keypad designed for entering telephone numbers. This type features the numbers 0 through 9 and the “*” and “#” characters prominently. This keyboard type does not support auto-capitalization.
NamePhonePad	Use a keypad designed for entering a person’s name or phone number. This keyboard type does not support auto-capitalization.
EmailAddress	Use a keyboard optimized for specifying email addresses. This type features the “@”, “.” and space characters prominently.
DecimalPad	Use a keyboard with numbers and a decimal point.
Twitter	Use a keyboard optimized for twitter text entry, with easy access to the @ and # characters.
WebSearch	Use a keyboard optimized for web search terms and URL entry. This type features the space and “.” characters prominently.

## iOSLineBreakMode

Specifies how [iOStextLabel](#) text contents break or wrap when the text is too long to fit.

BreakByWordWrapping	Text breaks by word.
BreakByCharWrapping	Text breaks on a single character, possibly breaking in the middle of a word.
BreakByClipping	The text is clipped at the end, perhaps in mid-character.
BreakByTruncatingHead	When the entire text cannot fit, text at the beginning is truncated and replaced with ellipses.
BreakByTruncatingTail	When the entire text cannot fit, text at the end is truncated and replaced with ellipses.
BreakByTruncatingMiddle	When the entire text cannot fit, text in the middle is truncated and replaced with ellipses.

## iOSTextAlignment

Specifies text alignment for iOS controls such as [iOStextLabel](#), [iOStextField](#), [iOStextArea](#) and [iOStextGraphics](#).

Left	Align text along the left edge.
Center	Align text equally along both sides of the center line.
Right	Align text along the right edge.

---

Justified	Fully justify the text so that the last line in a paragraph is natural aligned.
Natural	Use the default alignment associated with the current script.

## Methods

### IsText(val As Auto) As Boolean

Determines if the supplied Auto value contains a Text.

# Xojo.Core

The Xojo.Core namespace contains core classes you can use in all your projects.

## Namespace Summary

Name	Xojo.Core
Type	Namespace
Classes	<a href="#">Date</a> <a href="#">DateInterval</a> <a href="#">Dictionary</a> <a href="#">DictionaryEntry</a> <a href="#">Locale</a> <a href="#">MemoryBlock</a> <a href="#">MutableMemoryBlock</a> <a href="#">Point</a> <a href="#">Rect</a> <a href="#">Size</a> <a href="#">StackFrame</a> <a href="#">TextEncoding</a> <a href="#">Timer</a> <a href="#">TimeZone</a> <a href="#">WeakRef</a>
Interfaces	<a href="#">Iterable</a> <a href="#">Iterator</a>
Exceptions	<a href="#">BadDataException</a> <a href="#">ErrorException</a> <a href="#">InvalidArgumentException</a> <a href="#">IteratorException</a> <a href="#">LogicException</a> <a href="#">UnsupportedOperatorException</a>
Project Types	All
Platforms	All
Related	<a href="#">Using</a>

## Notes

Normally you have to include the full namespace to access the classes in Xojo.Core. The [Using](#) command allows you to just refer to the classes by name. For example, at the top of a method you can put this code:

```
Using Xojo.Core
```

Within the method you are then able to access the classes in Xojo.Core just by their simple names. So instead of Xojo.Core.Dictionary, you can write:

---

## Dim d As Dictionary



iOS projects include the entire Xojo namespace by default (Using Xojo), making it simple to refer to all the classes.

# Date

Used to store date/time values. Once a date is created, it cannot be modified, but you can create new dates by applying a [DateInterval](#) to an existing date.

## Class Summary

Name	Xojo.Core.Date
Type	Class
Enumerations	<a href="#">FormatStyles</a>
Properties	<a href="#">Day</a> <a href="#">DayOfWeek</a> <a href="#">DayOfYear</a> <a href="#">Hour</a> <a href="#">Minute</a> <a href="#">Month</a> <a href="#">Nanosecond</a> <a href="#">Second</a> <a href="#">SecondsFrom1970</a> <a href="#">TimeZone</a> <a href="#">Year</a>
Methods	<a href="#">ToText</a>
Shared Methods	<a href="#">FromText</a> <a href="#">Now</a>
Operators	<a href="#">Add</a> <a href="#">Subtract</a>
Project Types	All
Platforms	All
Related	<a href="#">DateInterval</a> <a href="#">TimeZone</a>
Example Projects	Examples/iOS/Framework/DateExample

## Notes



You cannot use **Dim d As New Date** to get the current date. Instead, use **Dim d As Date = Date.Now** or more simply, just **Date.Now**.

## Constructors

### Constructor(secondsFrom1970 As Double, timezone As TimeZone)

Creates a date using number of seconds from the first instant of 1 January 1970, GMT. Use a negative value to refer to specify a date before this date.

#### Sample Code

Convert a date in the current time zone to GMT:

```
Dim myDate As Xojo.Core.Date = Xojo.Core.Date.Now // Get date for current time zone
```

```
Dim gmt As New Xojo.Core.TimeZone(0) // Get GMT time zone
Dim gmtDate As New Xojo.Core.Date(myDate.SecondsFrom1970, gmt)
```

### Constructor(year As Integer, month As Integer, day As Integer, hour As Integer = 0, minute As Integer = 0, seconds As Integer = 0, nanoseconds As Integer = 0, timezone As TimeZone)

Creates a new date using the specified values. Use this method to create a specific date.

#### Sample Code

Create a new date for Aug 1, 2015:

```
Dim d As New Xojo.Core.Date(2015, 8, 1, Xojo.Core.TimeZone.Current) ' Aug 1, 2015</code>
```

Create a new date for Sep 1, 2015 at 3:20pm (and 30 seconds):

```
Dim d As New Xojo.Core.Date(2015, 8, 1, 15, 20, 30, 0, Xojo.Core.TimeZone.Current)
```

Convert a Classic Date to Xojo.Core.Date:

```
Dim d As New Date(2015, 10, 15)
Dim xcd As New Xojo.Core.Date(d.Year, d.Month, d.Day, Xojo.Core.TimeZone.Current)
```

## Enumerations

### FormatStyles

These are the various styles that can be used to convert a Date to Text with the ToText method.

Enum Value	Description
Short	The short date or time format: 12/1/14 12:23 PM
Medium	The medium date or time format: Dec 1, 2014 12:20:06 PM
Long	The long date or time format: December 1, 2014 12:23:53 PM EST
Full	The full date or time format: Monday, December 1, 2014 12:24:17 PM Eastern Standard Time
None	Prevents the date or time value from displaying.

## Notes

The samples shown above are the typical default settings. The specific formats are defined in the locale settings for your OS:

- OS X: Language & Region System Preference
- Windows: Time & Language setting
- Linux varies

## Properties

### Day As Integer (read-only)

The day of the month as a number.

#### Sample Code

```
Using Xojo.Core
Dim d As Date = Date.Now
Dim day As Integer = d.Day
```

### DayOfWeek As Integer (read-only)

The day of the week, with 1 = Sunday, 2 = Monday, 3 = Tuesday, 4 = Wednesday, 5 = Thursday, 6 = Friday, 7 = Saturday.

#### Sample Code

```
Dim day As Integer = Xojo.Core.Date.Now.DayOfWeek
```

## DayOfYear As integer (read-only)

The number of days into the year, 1-based. January 1st is 1.

### Sample Code

```
Dim yearDay As Integer = Xojo.Core.Date.Now.DayOfYear</code>
```

---

## Hour As Integer (read-only)

The hour number for the time portion of the date, using a 24-hour clock. For example, 20 is the value for 8 pm.

### Sample Code

```
Using Xojo.Core
Dim d As Date = Date.Now
Dim hour As Integer = d.Hour
```

---

## Minute As Integer (read-only)

The minute number of the time portion of the date.

### Sample Code

```
Using Xojo.Core
Dim d As Date = Date.Now
Dim minute As Integer = d.Minute
```

---

## Month As Integer (read-only)

The month number of the date.

### Sample Code

```
Dim d As Xojo.Core.Date = Xojo.Core.Date.Now
Dim month As Integer = d.Month
```

---

## Nanosecond As Integer (read-only)

The nanoseconds of the time portion of the date. A nanosecond is one billionth of a second.

## Sample Code

```
Dim d As Xojo.Core.Date = Xojo.Core.Date.Now
Dim nanosecond As Integer = d.Nanosecond
```

---

## Second As Integer (read-only)

The second number of the time portion of the date.

## Sample Code

```
Dim d As Xojo.Core.Date = Xojo.Core.Date.Now
Dim second As Integer = d.Second
```

---

## SecondsFrom1970 As Double (read-only)

The number of seconds since the first instant of 1 January 1970, GMT.

## Notes

You can save this value (in a file or database, for example) and use a Constructor to create a new Date instance for it.

## Sample Code

```
Using Xojo.Core
Dim d1 As Date = Date.Now
Dim seconds As Double = d1.SecondsFrom1970
Dim d2 As New Date(seconds, TimeZone.Current)
// Both d1 and d2 have the same date value
```

---

## TimeZone As TimeZone (read-only)

The time zone.

## Sample Code

```
Using Xojo.Core
Dim d As Date = Date.Now
Dim tz As TimeZone
tz = d.TimeZone
```

To change a date to a new time zone, for example GMT:

```
// myDate is an existing date
```

```
Using Xojo.Core
Dim GMTZone As New TimeZone("GMT")
Dim GMTDate As New Date(myDate.SecondsFrom1970, GMTZone)
```

---

## Year As Integer (read-only)

The year portion of the date.

### Sample Code

```
Dim d As Xojo.Core.Date = Xojo.Core.Date.Now
Dim year As Integer = d.Year
```

## Methods

**ToText**(loc As Locale = Nil, dateStyle As FormatStyles = FormatStyles.Medium, timeStyle As FormatStyles = FormatStyles.Medium) As Text

Converts the Date (along with the time component) to a text format using the optional locale and formats.

### Notes

The default non-locale savvy version (*loc* is Nil) of ToText returns a date in the format **YYYY-MM-DD HH:MM:SS**.

Example date output would be: 2014-10-31 09:38:24

### Sample Code

Display the current date and time:

```
Dim d As Xojo.Core.Date = Xojo.Core.Date.Now
Label1.Text = d.ToText</code>
```

To get just the date portion:

```
Using Xojo.Core
Dim d As Date = Date.Now
Dim t As Text = d.ToText(Locale.Current, Date.FormatStyles.Short,
Date.FormatStyles.None)
```

## Shared Methods

**FromText**(input As Text, loc As Locale = Nil) As Date

Converts a textual date to an actual date using the optional locale.

### Notes

The resulting date is in the current time zone (Xojo.Core.TimeZone.Current).

When no locale is specified, you can choose to include a time portion in 24-hour format.

### Sample Code

```
Using Xojo.Core
Dim dateValue As Text = "2015-08-01 11:00"
Dim myDate As Date = Date.FromText(dateValue)

dateValue = "2015-06-01"
Dim myDate As Date = Date.FromText(dateValue)

dateValue = Date.Now.ToText
myDate = Date.FromText(dateValue)
```

---

## Now As Date

The current date and time.

### Sample Code

Get the current date and time:

```
Dim d As Xojo.Core.Date = Xojo.Core.Date.Now
```

## Operators

### + (Add)

You can add a Date and a [DateInterval](#) to get a new Date.

### Sample Code

Get the date two months from today:

```
Using Xojo.Core
Dim twoMonths As New DateInterval
twoMonths.Months = 2 // 2 month interval

// Get date two months from today
Dim future As Date = Date.Now + twoMonths
```

---

### - (Subtract)

You can subtract a Date and a DateInterval to get a new Date.

You can subtract a Date from a Date and get a DateInterval

## Example

Get the date two months before today:

```
Using Xojo.Core
```

```
Dim twoMonths As New DateInterval  
twoMonths.Months = 2 // 2 month interval
```

```
// Get date two months before today  
Dim past As Date = Date.Now - twoMonths
```

Calculate the interval until January 1, 2030:

```
Using Xojo.Core
```

```
Dim d2 As New Date(2030, 1, 1, TimeZone.Current)  
Dim interval As DateInterval  
interval = d2 - Date.Now
```

# DateInterval

A DateInterval is used to modify a date object. You specify how much you want to offset the date by and then add (or subtract) the interval to the date to get a new date object.

## Class Summary

Name	Xojo.Core.DateInterval
Type	Class
Properties	<a href="#">Days</a> <a href="#">Hours</a> <a href="#">Minutes</a> <a href="#">Months</a> <a href="#">NanoSeconds</a> <a href="#">Seconds</a> <a href="#">Years</a>
Project Types	All
Platforms	All
Related	<a href="#">Date</a>

## Constructors

Constructor(years As Integer = 0, months As Integer = 0, days As Integer = 0, hours As Integer = 0, minutes As Integer = 0, seconds As Integer = 0, nanoseconds As Integer = 0)

Specifies an interval using the various properties.

## Example

Creates a date that is two months from today and two months before today:

```
Dim d As Date = Date.Now

Dim di As New DateInterval
di.Months = 2 // 2 months

// Get date two months from today
Dim d2 As Date = d + di

// Get date two months before today
Dim d3 As Date = d - di
```

---

## Properties

### Days As Integer

The number of days in the interval.

---

### Hours As Integer

The number of hours in the interval.

---

### Minutes As Integer

The number of minutes in the interval.

---

### Months As Integer

The number of months in the interval.

---

### NanoSeconds As Integer

The number of nanoseconds in the interval.

---

### Seconds As Integer

The number of seconds in the interval.

---

### Years As Integer

The number of years in the interval.

---

# Dictionary

The Dictionary class is an unordered, mutable, key-value store that is loosely typed. It implements the [Iterable](#) interface, allowing efficient and easy iteration over all key-value pairs.

## Class Summary

Name	Xojo.Core.Dictionary
Type	Class
Interfaces	<a href="#">Iterable</a>
Events	<a href="#">CompareKeys</a>
Properties	<a href="#">Count</a>
Methods	<a href="#">Clone</a> <a href="#">GetIterator</a> <a href="#">HasKey</a> <a href="#">Lookup</a> <a href="#">RemoveAll</a> <a href="#">Remove</a> <a href="#">Value</a>
Project Types	All
Platforms	All
Related	<a href="#">DictionaryEntry</a> <a href="#">KeyNotFoundException</a>

## Events

### CompareKeys(*lhs* As Auto, *rhs* As Auto) As Integer

Implement this event handler if you would like the Dictionary to support case-sensitive keys.

#### Parameters

<i>lhs</i>	The left-hand side of the comparison.
<i>rhs</i>	The right-hand side of the comparison.

#### Returns

Return a positive integer if *lhs* is greater than *rhs*.

Return 0 if *lhs* is the same as *rhs*.

Return a negative integer if *lhs* is less than *rhs*.

## Example

This code does a case-sensitive comparison of the text value of the specified keys:

```
Dim lhsText As Text = lhs
Dim rhsText As Text = rhs
Return lhsText.Compare(rhsText, Text.CompareCaseSensitive)
```

# Properties

## Count As Integer

The number of entries in the Dictionary (read-only).

# Methods

## Clone As Dictionary

Performs a shallow clone of the Dictionary, resulting in a new Dictionary that can be manipulated independently of the first. A shallow clone means that if a Dictionary Value or Key refers to a class instance, its contents are not also cloned.

### Return Value

Returns the clone of the Dictionary.

## Example

Clone a dictionary and then change the original:

```
Dim d1 As New Dictionary
d1.Value("Test") = "Hello, World!"

Dim d2 As Dictionary
d2 = d1.Clone

d1.Value("Test") = "Changed!"

// d2.Value("Test") is still "Hello, World!"
```

---

## HasKey(key As Auto) As Boolean

Determines where or not the Dictionary contains a value for the specified *key*.

### Parameters

<i>key</i>	An Auto that identifies the key to look for.
------------	--

## Return Value

True if the key exists in the Dictionary, False if it does not.

## Example

Check if "Test" is used as a key value:

```
If Not d1.HasKey("Test") Then
    d1.Value("Test") = "Initial value"
End If
```

---

## GetEnumerator As Iterator

Creates a new iterator for the Dictionary which will yield [DictionaryEntry](#) objects for its values. Part of the [Iterable](#) interface.

## Notes

You will not typically access this method directly. Instead use the ability to iterate over the Dictionary using a `For..Each` loop.

## Example

Iterate over the Dictionary entries using For Each:

Using `Xojo.Core`

```
Dim d As New Dictionary
d.Value("One") = "Testing"
d.Value("Two") = "Iterator"
```

```
For Each entry As DictionaryEntry In d
    TextArea1.Text = TextArea1.Text + " " + entry.Key + " " + entry.Value
Next
```

---

## Lookup(key As Auto, defaultValue As Auto) As Auto

Returns the value associated with the specified key. If there is no such entry, the *defaultValue* parameter is returned and no exception is raised.

## Parameters

<i>key</i>	An Auto that identifies the key to look for.
<i>defaultValue</i>	An Auto value that is returned when the key is not found in the Dictionary.

## Return Value

Returns the value for *key* (if *key* is found). Returns the *defaultValue* if *key* is not found.

## Example

if the User ID is not found in the Dictionary, return "UnknownUser":

```
Dim userID As Integer = 123
Dim user As Text
user = d1.Lookup(userID, "UnknownUser")
```

---

## RemoveAll

Removes all entries from the Dictionary. This invalidates all iterators that were created from the Dictionary.

## Example

Remove all entries from a Dictionary:

```
d1.RemoveAll
```

---

## Remove(key As Auto)

Removes a single entry with the specified *key* from the Dictionary, invalidating all iterators that were created from the Dictionary. If there is no entry in the Dictionary for the key, a `KeyNotFoundException` is raised.

## Parameters

<i>key</i>	An Auto that identifies the key to look for.
------------	--

## Exceptions

<a href="#">KeyNotFoundException</a>	When <i>key</i> is not in the Dictionary.
--------------------------------------	---

## Example

Remove the entry for the value "Test" in the Dictionary:

```
d1.Remove("Test")
```

---

## Value(key As Auto, Assigns newValue As Auto)

Adds a new key-value pair to the Dictionary. If there is already an entry with the specified key, its value is altered and no exception is raised.

## Parameters

<i>key</i>	An Auto specifying the key to which <i>newValue</i> is assigned.
<i>newValue</i>	An Auto that is the value to assign to <i>key</i> .

## Example

As this is an extension method, you use it with the assignment operator.

```
Dim d As New Xojo.Core.Dictionary
d.Value(123) = "Bob Roberts"
```

## Value(key As Auto) As Auto

Returns the value associated with the specified key. If there is no such entry, a `KeyNotFoundException` is raised.

## Parameters

<i>key</i>	An Auto that identifies the key to look for.
------------	--

## Exceptions

<a href="#">KeyNotFoundException</a>	When <i>key</i> is not in the Dictionary.
--------------------------------------	---

## Example

```
Dim name As Text
name = d.Value(123)
```

```
Dim id As Integer = d.Value("Age")
Label1.Text = id.ToText</code>
```

If the Dictionary contains an array, you can iterate through by first assigning the value to an Auto array and then using the correct type in a `For...Each` loop. For example, this code gets an array of Dictionaries:

```
Dim myAutoArray() As Auto = myDictionary.Value("KeyForArrayOfDictionaries")
```

```
Dim value As Auto
For Each dict as Dictionary In myAutoArray
    value = dict.Value("Test")
Next
```

# DictionaryEntry

The DictionaryEntry class represents a single key-value pair in a [Dictionary](#). It is immutable and is not tied to the Dictionary it was created from, so subsequent updates to the Dictionary will not change the DictionaryEntry property values.

## Class Summary

Name	<a href="#">Xojo.Core.DictionaryEntry</a>
Type	Class
Properties	<a href="#">Key</a> <a href="#">Value</a>
Project Types	All
Platforms	All
Related	<a href="#">Dictionary</a>

## Properties

### Key As Auto

The entry's key (read-only).

#### Sample Code

```
Dim entries() As Xojo.Core.DictionaryEntry
For Each entry As Xojo.Core.DictionaryEntry In entries
  TextArea1.Text = TextArea1.Text + " " + entry.Key + " " + entry.Value
Next
```

### Value As Auto

The entry's value (read-only).

#### Sample Code

```
Dim entries() As Xojo.Core.DictionaryEntry
For Each entry As Xojo.Core.DictionaryEntry In entries
  TextArea1.Text = TextArea1.Text + " " + entry.Key + " " + entry.Value
Next
```

# Iterable

The Iterable interface allows objects to denote themselves as being able to be used with "For Each" loops and provides a way to create Iterators, which do the actual work.

## Interface Summary

Name	Xojo.Core.Iterable
Type	Interface
Methods	<a href="#">GetEnumerator</a>
Project Types	All
Platforms	All
Related	<a href="#">Dictionary</a> <a href="#">FolderItem</a> <a href="#">Iterator</a>

## Methods

### GetEnumerator As [Iterator](#)

Returns an Iterator that can be used with "For Each" loops.

#### Sample Code

Classes that implement Iterable, such as [Dictionary](#), can be used with For..Each loops:

```
Dim entries() As Xojo.Core.DictionaryEntry
For Each entry As Xojo.Core.DictionaryEntry In d
  TextArea1.Text = TextArea1.Text + " " + entry.Key + " " + entry.Value
Next
```

# Iterator

Iterators perform the actual task of yielding values from the Iterable object.

## Interface Summary

Name	Xojo.Core.Iterator
Type	Interface
Methods	<a href="#">MoveNext</a> <a href="#">Value</a>
Project Types	All
Platforms	All
Related	<a href="#">DictionaryEntry</a> <a href="#">Iterable</a> <a href="#">IteratorException</a>

## Notes

Initially the Iterator is positioned before the first item and invoking `MoveNext` is required before accessing the Iterator's current value. Failure to do so is undefined behavior and typically results in the implementor raising an exception.

The `MoveNext` method advances the Iterator one item. If the Iterator has ran out of values to yield, `MoveNext` returns `False` and continues returning `False` until the Iterator is destructed. Access to the iterator's current value at that point is undefined behavior and implementors typically raise an [IteratorException](#).

Iterators may be invalidated if the iterable object is was created from is mutated. Invoking `MoveNext` or `Value` on an invalidated iterator results in undefined behavior and implementors should raise an [IteratorException](#).

While the iterator is valid, the `Value` method must remain the same across multiple calls to `Value` until `MoveNext` has been invoked.

## Methods

### MoveNext As Boolean

Moves the iterator to the next item.

#### Return Value

Returns `False` when there are no more items, otherwise returns `True`.

## Value As Auto

The value of the current item in the iterator. The Value is invalid when MoveNext returns False. It is also invalid before the first call to MoveNext.

### Return Value

Returns the value of the current item in the iterator.

# Locale

Used to represent a locale, which describes linguistic, cultural and other locale-specific information. For example, a locale might specify "English as spoken in the United States, using the metric system".

## Class Summary

Name	Xojo.Core.Locale
Type	Class
Properties	<a href="#">Identifier</a> <a href="#">CurrencySymbol</a> <a href="#">GroupingSeparator</a> <a href="#">DecimalSeparator</a>
Shared Properties	<a href="#">Current</a> <a href="#">Raw</a>
Project Types	All
Platforms	All
Related	<a href="#">Text</a>

## Constructors

### Constructor(localeIdentifier As Text)

Creates a locale with the given *localeIdentifier*.

#### Notes

As an example, "en-US" is used as a localeIdentifier for English in the United States.

Links to look up codes:

- The two-letter [ISO 639-1 standard](#)
- [ISO Online Browsing Platform](#)
- [ISO Language Code Table](#)

#### Exceptions

RuntimeException	Invalid <i>localeIdentifier</i> .
------------------	-----------------------------------

#### Example

Create an English US locale:

```
Dim locale As New Xojo.Core.Locale("en-US")
```

## Properties

### Identifier As Text (read-only)

This is the locale's identifier. The identifier may not exactly match the identifier passed in via the constructor due to conversions (e.g. 'en\_US' could become 'en-US').

#### Example

Get the locale's identifier:

```
Dim locale As New Xojo.Core.Locale("en-US")
```

```
Dim id As Text  
id = locale.Identifier
```

---

### CurrencySymbol As Text (read-only)

The locale's currency symbol.

#### Example

Get the locale's currency symbol:

```
Dim locale As New Xojo.Core.Locale("en-US")
```

```
Dim symbol As Text  
symbol = locale.CurrencySymbol
```

---

### GroupingSeparator As Text (read-only)

The locale's separator for grouping a number. In "en-US", this is the 'comma' placed between every three integer digits.

#### Example

Get the locale's grouping separator:

```
Dim locale As New Xojo.Core.Locale("en-US")
```

```
Dim grouping As Text  
grouping = locale.GroupingSeparator
```

---

---

## DecimalSeparator As Text (read-only)

The locale's separator between the integer and decimal portions of a number. In "en-US", this is the 'period'.

### Example

Get the locale's decimal separator:

```
Dim locale As New Xojo.Core.Locale("en-US")
```

```
Dim decimal As Text  
decimal = locale.DecimalSeparator
```

## Shared Properties

### Current As Locale (read-only)

The user's current locale. For web apps, this is the session's current locale, which is based off of the information sent by the browser and will return Nil if there is no session or the information is unavailable.

### Example

Get the current locale:

```
Dim currentLocale As Xojo.Core.Locale  
currentLocale = Xojo.Core.Locale.Current
```

---

### Raw As Locale (read-only)

A machine-independent locale that has a well-defined format (i.e. the [POSIX locale](#)) and is not affected by the user's settings.

### Example

Get the raw locale:

```
Dim currentLocale As Xojo.Core.Locale  
currentLocale = Xojo.Core.Locale.Raw
```

# MemoryBlock

A read-only class for managing blocks of memory. Since the contents of a MemoryBlock cannot be altered, if you need to alter a MemoryBlock, use [MutableMemoryBlock](#).

## Class Summary

Name	Xojo.Core.MemoryBlock
Type	Class
Properties	<a href="#">LittleEndian</a> <a href="#">Size</a> <a href="#">Data</a>
Methods	<a href="#">Clone</a> <a href="#">IndexOf</a> <a href="#">Left</a> <a href="#">Mid</a> <a href="#">Right</a> <a href="#">BooleanValue</a> <a href="#">CurrencyValue</a> <a href="#">CStringValue</a> <a href="#">DoubleValue</a> <a href="#">Int8Value</a> <a href="#">Int16Value</a> <a href="#">Int32Value</a> <a href="#">Int64Value</a> <a href="#">PtrValue</a> <a href="#">SingleValue</a> <a href="#">UInt8Value</a> <a href="#">UInt16Value</a> <a href="#">UInt32Value</a> <a href="#">UInt64Value</a> <a href="#">WStringValue</a>
Project Types	All
Platforms	All
Related	<a href="#">BinaryStream</a> <a href="#">Byte</a> <a href="#">UInteger</a> <a href="#">Ptr</a> <a href="#">MutableMemoryBlock</a> <a href="#">TextEncoding</a>

## Notes

To convert Text to a MemoryBlock, use [TextEncoding.ConvertTextToData](#). To convert a MemoryBlock to Text, use [TextEncoding.ConvertDataToText](#).

Compared to [MutableMemoryBlock](#), `MemoryBlock` is more efficient in terms of memory use savings, but doesn't really have any difference in terms of access. Being able to pass around immutable chunks of data allows the framework to directly reference internal data and avoid having to copy it on the off chance it might get modified.

Converting to a Classic `MemoryBlock`

When using both the Classic and New Framework in a single (desktop, web or console) app, you may find that you need to convert a `Xojo.Core.MemoryBlock` to a Classic [MemoryBlock](#) for use with other methods. You can do so by copying the data from the `Xojo.Core.MemoryBlock` to the Classic `MemoryBlock`:

```
' newMB is a Xojo.Core.MemoryBlock
Dim temp As MemoryBlock = newMb.Data
Dim mb As New MemoryBlock(newMb.Size)
mb.StringValue(0, mb.Size) = temp.StringValue(0, mb.Size)
temp = Nil ' optional as it will also be cleared when it goes out of scope
```

## Constructors

### Constructor(size As UInteger)

Creates a `MemoryBlock` with the desired size in bytes.

#### Parameters

<i>size</i>	The size (in bytes) of the <code>MemoryBlock</code> to create. Size is either a 32-bit or 64-bit unsigned integer depending on the OS.
-------------	--

#### Exceptions

<code>OutOfMemoryException</code>	If there is insufficient memory to allocate <i>size</i> bytes <b>and</b> there is sufficient memory to recover from this error.
-----------------------------------	---

#### Sample Code

```
Dim mb As New Xojo.Core.MemoryBlock(1024) // Reserve 1K bytes
```

### Constructor(p As Ptr, size As UInteger)

Creates a `MemoryBlock` from an existing chunk of memory of a specific size.

#### Parameters

<i>p</i>	A <code>Ptr</code> that points to a chunk of memory that was allocated using some other method such as a <code>Declare to an OS API call</code> .
<i>size</i>	The size (in bytes) of the allocated memory as either a 32-bit or 64-bit unsigned integer depending on the OS. This size is used for bounds checking on the various getter and setter functions.

## Notes

The pointer *p* serves as the MemoryBlock's backing. Mutating the MemoryBlock will also mutate the contents of the chunk of memory. The MemoryBlock does not take ownership of the memory, so the application is responsible for freeing it after the MemoryBlock has been destructed. If it is freed before then, all MemoryBlock behavior is undefined and will likely crash.

## Exceptions

NilObjectException	If <i>p</i> is Nil.
RuntimeException	If <i>size</i> is 0.
RuntimeException	If it can be determined that <i>p</i> is neither readable or writeable.

## Constructor(*p* As Ptr)

Creates a MemoryBlock from an existing chunk of memory.

### Parameters

<i>p</i>	A Ptr that points to a chunk of memory that was allocated using some other method such as a Declare to an OS API call.
----------	--

## Constructor(bytes() As Byte)

Creates a MemoryBlock from an existing Byte array.

### Parameters

<i>bytes()</i>	A Byte array that is used to populate the MemoryBlock.
----------------	--

## Notes

The contents of the Byte array are copied into the MemoryBlock. Mutating the MemoryBlock has no effect on the original array.

## Exceptions

NilObjectException	If <i>bytes()</i> is Nil.
OutOfMemoryException	If there is insufficient memory to copy the byte array <b>and</b> there is sufficient memory to recover from this error.

## Sample Code

```
Dim bytes() As Byte
bytes.Append(&hff)
bytes.Append(&hff)
```

```
bytes.Append(&hff)

Dim b As New Xojo.Core.MemoryBlock(bytes)
```

---

## Constructor(other As MemoryBlock)

Creates a new MemoryBlock with the data from an existing MemoryBlock.

### Sample Code

```
Dim f As FolderItem = SpecialFolder.GetResource("MyImage.JPG")

Dim b As BinaryStream
b = BinaryStream.Open(f, BinaryStream.LockModes.Read)

Dim mb As New MemoryBlock(b.Read(b.Length))

b.Close

Dim image As iOSImage
image = Image.FromData(mb)

ImageView1.Image = image</code>
```

## Properties

### LittleEndian As Boolean

When True, indicates that the MemoryBlock data is in LittleEndian format. The default is True.

---

### Size As UInteger (read-only)

The size (in bytes) of the MemoryBlock.

#### Notes

If the size is unknown, this returns &hFFFFFF (or &hFFFFFFFFFFFFFFFF on 64-bit builds).

---

### Data As Ptr (read-only)

The memory data in the MemoryBlock.

#### Notes

This data is invalidated when the MemoryBlock is changed.

## Methods

### Clone As MemoryBlock

Makes a copy of the MemoryBlock. If its size is unknown, it copies the content.

### IndexOf(offset As UInteger, other As MemoryBlock) As UInteger

Returns the position of the *other* MemoryBlock starting at the *offset*. Returns -1 if *other* was not found.

Parameters

<i>offset</i>	The starting byte position.
<i>other</i>	The MemoryBlock to find.

Returns

A UInteger of the byte position where *other* starts.

### Left(bytes As UInteger) As MemoryBlock

Gets the specified number of bytes at the beginning of the MemoryBlock.

Parameters

<i>bytes</i>	The number of bytes to get.
--------------	-----------------------------

Exceptions

RuntimeException	If the MemoryBlock does not have a known size.
OutOfBoundsException	If the bytes exceeds the size of the MemoryBlock.

### Mid(offset As UInteger) As MemoryBlock

### Mid(offset As UInteger, length As UInteger) As MemoryBlock

Gets the specified number of bytes at the offset position within the MemoryBlock.

Parameters

<i>offset</i>	The starting position within the MemoryBlock.
<i>length</i>	The number of bytes to get.

Exceptions

OutOfBoundsException	If the offset or offset + length exceeds the size of the MemoryBlock.
OutOfBoundsException	If the length of bytes exceeds the MemoryBlock or Byte array.

## Right(bytes As UInteger) As MemoryBlock

Gets the bytes from the end of the MemoryBlock.

### Parameters

<i>bytes</i>	The number of bytes to get.
--------------	-----------------------------

### Exceptions

OutOfBoundsException	If the number of bytes exceeds the size of the MemoryBlock.
RuntimeException	If the MemoryBlock does not have a known size.

### Sample Code

Get the last two bytes of a MemoryBlock:

```
Dim bytes() As Byte
bytes.Append(&h0a)
bytes.Append(&h0b)
bytes.Append(&h0c)
```

```
Dim data As New Xojo.Core.MemoryBlock(bytes)
Dim mb As Xojo.Core.MemoryBlock = data.Right(2)
```

## BooleanValue(offset As UInteger) As Boolean

Gets a Boolean value in the MemoryBlock at the specified *offset* (in bytes). Any non-zero value is considered True.

## CurrencyValue(offset As UInteger) As Currency

Gets an 8-byte Currency value in the MemoryBlock at the specified *offset* (in bytes).

## CStringValue(offset As UInteger) As CString

Gets a CString value in the MemoryBlock at the specified *offset* (in bytes).

### Notes

A CString is a sequence of non-zero bytes followed by a terminating Chr(0) to mark the end of the CString.

---

## DoubleValue(offset As UInteger) As Double

Gets a Double value in the MemoryBlock at the specified *offset* (in bytes).

---

## Int8Value(offset As UInteger) As Int8

Gets an Int8 value in the MemoryBlock at the specified *offset* (in bytes).

---

## Int16Value(offset As UInteger) As Int16

Gets an Int16 value in the MemoryBlock at the specified *offset* (in bytes).

---

## Int32Value(offset As UInteger) As Int32

Gets or sets an Int32 value in the MemoryBlock at the specified *offset* (in bytes).

---

## Int64Value(offset As UInteger) As Int64

Gets an Int64 value in the MemoryBlock at the specified *offset* (in bytes).

---

## PtrValue(offset As UInteger) As Ptr

Gets a Ptr value in the MemoryBlock at the specified *offset* (in bytes).

---

## SingleValue(offset As UInteger) As Single

Gets a Single value in the MemoryBlock at the specified *offset* (in bytes).

---

## UInt8Value(offset As UInteger) As UInt8

Gets an UInt8 value in the MemoryBlock at the specified *offset* (in bytes).

---

## UInt16Value(offset As UInteger) As UInt16

Gets an UInt16 value in the MemoryBlock at the specified *offset* (in bytes).

---

## UInt32Value(offset As UInteger) As UInt32

Gets an UInt32 value in the MemoryBlock at the specified *offset* (in bytes).

---

## UInt64Value(offset As UInteger) As UInt64

Gets an UInt64 value in the MemoryBlock at the specified *offset* (in bytes).

---

## WStringValue(offset As UInteger) As WString

Gets a WString value in the MemoryBlock at the specified *offset* (in bytes).

## Operators

### = (Comparison)

MemoryBlocks can be compared with each other using the "=" operator.

#### Notes

MemoryBlocks are considered equal for these conditions:

- The MemoryBlocks are the same object
- Both MemoryBlocks' data point to the same location
- Both MemoryBlocks' size and content are the same

There is no concept of one MemoryBlock being greater than or less than another MemoryBlock.

# MutableMemoryBlock

A mutable class for managing and modifying blocks of memory.

## Class Summary

Name	Xojo.Core.MutableMemoryBlock
Type	Class
Super	<a href="#">MemoryBlock</a>
Properties	<a href="#">LittleEndian</a> <a href="#">Size</a> <a href="#">Data</a>
Methods	<a href="#">Append</a> <a href="#">Insert</a> <a href="#">Left</a> <a href="#">Mid</a> <a href="#">Remove</a> <a href="#">Right</a> <a href="#">BooleanValue</a> <a href="#">CurrencyValue</a> <a href="#">CStringValue</a> <a href="#">DoubleValue</a> <a href="#">Int8Value</a> <a href="#">Int16Value</a> <a href="#">Int32Value</a> <a href="#">Int64Value</a> <a href="#">PtrValue</a> <a href="#">SingleValue</a> <a href="#">UInt8Value</a> <a href="#">UInt16Value</a> <a href="#">UInt32Value</a> <a href="#">UInt64Value</a> <a href="#">WStringValue</a>
Project Types	All
Platforms	All
Related	<a href="#">Byte</a> <a href="#">UInteger</a> <a href="#">MemoryBlock</a> <a href="#">Ptr</a>

## Constructors

## Constructor(size As UInteger)

Creates a MemoryBlock with the desired size in bytes.

### Parameters

<i>size</i>	The size (in bytes) of the MemoryBlock to create. Size is either a 32-bit or 64-bit unsigned integer depending on the OS.
-------------	---

### Exceptions

OutOfMemoryException	If there is insufficient memory to allocate <i>size</i> bytes <b>and</b> there is sufficient memory to recover from this error.
----------------------	---

## Constructor(p As Ptr, size As UInteger)

Creates a MemoryBlock from an existing chunk of memory of a specific size.

### Parameters

<i>p</i>	A Ptr that points to a chunk of memory that was allocated using some other method such as a Declare to an OS API call.
<i>size</i>	The size (in bytes) of the allocated memory as either a 32-bit or 64-bit unsigned integer depending on the OS. This size is used for bounds checking on the various getter and setter functions.

### Notes

The pointer *p* serves as the MemoryBlock's backing. Mutating the MemoryBlock will also mutate the contents of the chunk of memory. The MemoryBlock does not take ownership of the memory, so the application is responsible for freeing it after the MemoryBlock has been destructed. If it is freed before then, all MemoryBlock behavior is undefined and will likely crash.

### Exceptions

NilObjectException	If <i>p</i> is Nil.
RuntimeException	If <i>size</i> is 0.
RuntimeException	If it can be determined that <i>p</i> is neither readable or writeable.

## Constructor(p As Ptr)

Creates a MemoryBlock from an existing chunk of memory.

### Parameters

<i>p</i>	A Ptr that points to a chunk of memory that was allocated using some other method such as a Declare to an OS API call.
----------	--

## Constructor(bytes() As Byte)

Creates a MemoryBlock from an existing Byte array.

### Parameters

<i>bytes()</i>	A Byte array that is used to populate the MemoryBlock.
----------------	--

### Notes

The contents of the Byte array are copied into the MemoryBlock. Mutating the MemoryBlock has no effect on the original array.

### Exceptions

NilObjectException	If bytes() is Nil.
OutOfMemoryException	If there is insufficient memory to copy the byte array <b>and</b> there is sufficient memory to recover from this error.

## Constructor(other As MemoryBlock)

Creates a MutableMemoryBlock from a [MemoryBlock](#).

### Parameters

<i>other</i>	The MemoryBlock to recreate as a MutableMemoryBlock.
--------------	--

## Methods

### Append(other As MemoryBlock)

### Append(other() As Byte)

Appends *other* data (which consists of either a MemoryBlock or a Byte array) to the end of the MemoryBlock and adjusts its size. This invalidates the Data property.

### Parameters

<i>other</i>	You can supply the data to append as either another MemoryBlock or a Byte array.
--------------	--

### Exceptions

NilObjectException	If <i>other</i> is Nil.
RuntimeException	If the source MemoryBlock has an unknown size or if the <i>other</i> MemoryBlock has an unknown size.

## Insert(offset As UInteger, other As MemoryBlock)

### Insert(offset As UInteger, other() As Byte)

Inserts other data (which consists of either a MemoryBlock or a Byte array) into the MemoryBlock at the specified offset (byte position). This invalidates the Data property.

#### Parameters

<i>other</i>	You can supply the data to append as either another MemoryBlock or a Byte array.
--------------	--

#### Exceptions

NilObjectException	If <i>other</i> is Nil.
RuntimeException	If the source MemoryBlock has an unknown size or if the <i>other</i> MemoryBlock has an unknown size.
OutOfBoundsException	If <i>offset</i> is greater than the size of the MemoryBlock.

## Remove(offset As UInteger, length As UInteger)

Removes *length* bytes from the MemoryBlock starting at *offset*, adjusting its size. This potentially invalidates the Data property.

#### Parameters

<i>offset</i>	The starting offset (in bytes) in the MemoryBlock from which to remove bytes.
<i>length</i>	The number of bytes to remove.

#### Exceptions

RuntimeException	If the MemoryBlock has an unkown size.
OutOfBoundsException	If the offset (or offset + length) is greater than the size of the MemoryBlock.

## Left(bytes As UInteger) As MemoryBlock

### Left(bytes As UInteger, Assigns value As MemoryBlock)

Gets or Replaces the specified number of bytes at the beginning of the MemoryBlock.

#### Parameters

<i>bytes</i>	The number of bytes to get or replace.
<i>value</i>	When replacing data, specifies the MemoryBlock or the Byte array to use.

## Exceptions

<code>NilObjectException</code>	If trying to assign a Nil MemoryBlock or Byte array.
<code>RuntimeException</code>	If the MemoryBlock does not have a known size.
<code>OutOfBoundsException</code>	If the bytes exceeds the size of the MemoryBlock.

## Mid(offset As UInteger, length As UInteger, Assigns value As MemoryBlock)

## Mid(offset As UInteger, length As UInteger, Assigns value() As Byte)

Gets or Replaces the specified number of bytes at the offset position within the MemoryBlock.

## Parameters

<i>offset</i>	The starting position within the MemoryBlock.
<i>length</i>	The number of bytes to get or replace.

## Exceptions

<code>OutOfBoundsException</code>	If the offset or offset + length exceeds the size of the MemoryBlock.
<code>NilObjectException</code>	If trying to assign a Nil MemoryBlock or Byte array.
<code>OutOfBoundsException</code>	If the length of bytes exceeds the MemoryBlock or Byte array.

## Right(bytes As UInteger) As MemoryBlock

## Right(bytes As UInteger, Assigns value As MemoryBlock)

Gets or Replaces the bytes from the end of the MemoryBlock.

## Parameters

<i>bytes</i>	The number of bytes to get or replace.
<i>value</i>	When replacing data, specifies the MemoryBlock or the Byte array to use.

## Exceptions

<code>OutOfBoundsException</code>	If the number of bytes exceeds the size of the MemoryBlock.
<code>RuntimeException</code>	If the MemoryBlock does not have a known size.
<code>NilObjectException</code>	If trying to assign a Nil MemoryBlock or Byte array

---

## BooleanValue(offset As UInteger, Assigns value As Boolean) As Boolean

Sets a Boolean value in the MemoryBlock at the specified *offset* (in bytes). Any non-zero value is considered True.

---

## CStringValue(offset As UInteger, Assigns value As CString)

Sets a CString value in the MemoryBlock at the specified *offset* (in bytes).

---

## CurrencyValue(offset As UInteger, Assigns value As Currency)

Sets an 8-byte Currency value in the MemoryBlock at the specified *offset* (in bytes).

---

## DoubleValue(offset As UInteger, Assigns value As Double)

Sets a Double value in the MemoryBlock at the specified *offset* (in bytes).

---

## Int8Value(offset As UInteger) As Int8

Gets or sets an Int8 value in the MemoryBlock at the specified *offset* (in bytes).

---

## Int16Value(offset As UInteger, Assigns value As Int16)

Sets an Int16 value in the MemoryBlock at the specified *offset* (in bytes).

---

## Int32Value(offset As UInteger, Assigns value As Int32)

Sets an Int32 value in the MemoryBlock at the specified *offset* (in bytes).

---

## Int64Value(offset As UInteger, Assigns value As Int64)

Sets an Int64 value in the MemoryBlock at the specified *offset* (in bytes).

---

## PtrValue(offset As UInteger, Assigns value As Ptr)

Sets A Ptr value in the MemoryBlock at the specified *offset* (in bytes).

---

## SingleValue(offset As UInteger, Assigns value As Single)

Sets a Single value in the MemoryBlock at the specified *offset* (in bytes).

---

---

## UInt8Value(offset As Integer, Assigns value As UInt8)

Sets an UInt8 value in the MemoryBlock at the specified *offset* (in bytes).

---

## UInt16Value(offset As Integer, Assigns value As UInt16)

Sets an UInt16 value in the MemoryBlock at the specified *offset* (in bytes).

---

## UInt32Value(offset As Integer, Assigns value As UInt32)

Sets an UInt32 value in the MemoryBlock at the specified *offset* (in bytes).

---

## UInt64Value(offset As Integer, Assigns value As UInt64)

Sets an UInt64 value in the MemoryBlock at the specified *offset* (in bytes).

---

## WStringValue(offset As Integer, Assigns value As WString)

Sets a WString value in the MemoryBlock at the specified *offset* (in bytes).

---

# Point

Represents a point in 2D space. Useful for things like the center of a rectangle or the location of the mouse or touch event. Points are immutable and cannot be changed.

## Class Summary

Name	Xojo.Core.Point
Type	Class
Properties	<a href="#">X</a> , <a href="#">Y</a>
Methods	<a href="#">DistanceTo</a> <a href="#">Translate</a>
Shared Methods	<a href="#">Zero</a>
Project Types	All
Platforms	All
Related	<a href="#">Rect</a> <a href="#">Size</a>

## Constructors

### Constructor(X As Double, Y As Double)

Initializes the Point with an (x, y) of the parameters passed in.

## Properties

### X As Double (read-only)

The point's X coordinate.

### Y As Double (read-only)

The point's X coordinate.

## Methods

## DistanceTo(Other As Point) As Double

Calculates the distance from one point to another.

### Parameters

<i>other</i>	The destination point.
--------------	------------------------

### Return Value

Returns a double that measures the distance between the two points.

### Example

Calculate the distance (length of the line) between two points:

```
Dim p1 As New Xojo.Core.Point(10, 20)
Dim p2 As New Xojo.Core.Point(200, 5)
Dim dist As Double = p1.DistanceTo(p2)
```

---

## Translate(deltaX As Double, deltaY As Double) As Point

Creates a new point by adding 'deltaX' to Self.X and 'deltaY' to Self.Y.

### Parameters

<i>deltaX</i>	The amount to add to the original X position.
<i>deltaY</i>	The amount to add to the original Y position.

### Return Value

Returns a new Point with the updated coordinates.

### Example

Create a new point that is 100 greater than the old point's X coordinate:

```
Dim origPoint As New Point(50, 50)
Dim newPoint As Point = origPoint.Translate(100, 0)
```

## Shared Methods

### Zero As Xojo.Core.Point

Returns a Point with an (x, y) of (0, 0). This is a convenience for creating a new Point with those coordinates and allows the implementation to have a singleton value (as Point objects are immutable).

## Example

Calculate the distance from (0, 0):

```
Dim origPoint As New Point(123, 456)

Dim distance As Double = Xojo.Core.Point.Zero.DistanceTo(origPoint)
```

# Operators

## Add (+)

You can use the plus operator (+) to add the X and Y coordinates of the points together and get a new Point:

### Example

Adds two points together to get a new point:

```
Dim p1 As New Point(50, 50)
Dim p2 As New Point(25, 40)

Dim p3 As Point = p1 + p2 // p3 is (75, 90)
```

## Compare (>, <, =, >=, <=, <>)

You can use comparison operators to determine if one point is greater than another. A point is compared using its X value first and then the Y value if both X values are the same.

### Example

```
Dim p1 As New Point(50, 50)
Dim p2 As New Point(25, 60)

If p1 > p2 Then
    // True, because 50 > 25
End If
```

## Negate (-)

Use the negate operator (-) to create a new point with the X and Y coordinates negated.

### Example

Negate an existing point to get a new point:

```
Dim p1 As New Point(50, 50)
Dim p2 As Point

p2 = -p1 // p2 is (-50, -50)
```

---

## Subtract (-)

Use the subtraction operator (-) to subtract the coordinates of two points to get a new Point.

### Example

Subtract two points to get a new point:

```
Dim p1 As New Point(50, 50)
```

```
Dim p2 As New Point(10, 10)
```

```
Dim p3 As Point
```

```
p3 = p1 - p2 // p3 is (40, 40)
```

# Rect

A way to represent an axis-aligned rectangular area. Rect can be used for window size/position, control size/position and drawing of controls.

## Class Summary

Name	Xojo.Core.Rect
Type	Class
Properties	<a href="#">Bottom</a> <a href="#">Center</a> <a href="#">Height</a> <a href="#">Left</a> <a href="#">Origin</a> <a href="#">Right</a> <a href="#">Size</a> <a href="#">Top</a> <a href="#">Width</a>
Methods	<a href="#">Contains</a> <a href="#">Inset</a> <a href="#">Intersection</a> <a href="#">Offset</a> <a href="#">Union</a>
Project Types	All
Platforms	All
Related	<a href="#">Point</a> <a href="#">Size</a>

## Constructors

### Constructor(left As Double, top As Double, width As double, Height As Double)

Creates a new Rect with the specified values.

#### Parameters

<i>left</i>	A Double specifying the leftmost position of the Rect.
<i>top</i>	A Double specifying the topmost position of the Rect.
<i>width</i>	A Double specifying the width of the Rect.
<i>height</i>	A Double specifying the height of the Rect.

## Example

Create a new Rect:

```
Dim myRect As New Rect(10, 10, 100, 50)
```

---

## Constructor(origin As Point, size As xoyo.Core.Size)

Creates a new Rect starting at the *origin* Point for the specified *size*.

### Parameters

<i>origin</i>	A <a href="#">Point</a> specifying the top-left coordinate of the Rect.
<i>size</i>	A <a href="#">Size</a> specifying the size of the Rect.

## Example

Creates a new Rect:

```
Dim p As New Point(10, 10)
Dim s As New Size(100, 50)

Dim myRect As New Rect(p, s)
```

## Properties

### Bottom As Double (read-only)

The bottom position of the Rect.

---

### Center As [Point](#) (read-only)

The center point of the Rect.

---

### Height As Double (read-only)

The height of the Rect.

---

### Left As Double (read-only)

The left position of the Rect.

---

---

## Origin As Point (read-only)

The top-left point of the Rect.

---

## Right As Double (read-only)

The right position of the Rect.

---

## Size As Size (read-only)

The size of the Rect.

---

## Top As Double (read-only)

The top position of the Rect.

---

## Width As Double (read-only)

The width of the Rect.

---

## Methods

### Contains(P As Point) As Boolean

Determines if the point is contained within the rect.

---

### Contains(Other As Rect) As Boolean

Determines if the other rect is contained with the rect.

---

### Inset(deltaX As Double, deltaY As Double) As Rect

Returns a new rect with the same center, but each side inset by the given amount. Specify a negative value to grow the rectangle.

---

### Intersection(Other As Rect) As Rect

Calculates the area of intersection with the other rect.

---

---

## Offset(DeltaX As Integer, DeltaY As Integer) As Rect

A new rect offset by the specified values.

---

## Offset(delta as Point) As Rect

A new rect offset by the specified delta point.

---

## ScaledFromCenter(factor As Double) As Rect

Scales the rect around its center by the given factor.

---

## Union(Other As Rect) As Rect

A new rect that contains both rects.

## Operators

### Comparison

Compares first by Size, then by Origin

# Size

Used to represent 2D sizes, commonly used for window/view size, control size, etc.

## Class Summary

Name	Xojo.Core.Size
Type	Class
Properties	<a href="#">Height</a> <a href="#">Width</a>
Shared Methods	<a href="#">Zero</a>
Project Types	All
Platforms	All
Related	<a href="#">Point</a> <a href="#">Rect</a>

## Constructors

### Constructor(width As Double, height As Double)

Creates a new Size using the specified values.

## Properties

### Height As Double (read-only)

The height.

---

### Width As Double (read-only)

The width.

## Shared Methods

### Between(p1 As Point, p2 As Point) As Size

Calculates the Size between two points.

---

## Zero As Size

A Size with width and height of 0.

## Operators

Operator\_Add(other As Size) As Size

Operator\_Compare(Target As Size) As Integer

Compare first by width, then by height.

Operator\_Divide(divisor As Double) As Size

Operator\_Multiply(factor As Double) As Size

Operator\_Negate() As Size

Operator\_Subtract(other As Size) As Size

# StackFrame

Represents a single frame on the stack.

## Class Summary

Name	Xojo.Core.StackFrame
Type	Class
Properties	<a href="#">Address</a> <a href="#">Name</a>
Project Types	All
Platforms	All
Related	<a href="#">RuntimeException</a>

## Properties

### Address As Ptr (read-only)

The frame's instruction pointer.

---

### Name As Text (read-only)

The locally resolved function name corresponding to the address. Since this is looked up at runtime, it will be incorrect if the image containing the address lacks symbol information or has had symbols stripped.

# TextEncoding

An encoding represents a way of converting text to and from raw bytes. Encodings are created from IANA character set names or the shared properties for the very common encodings. Use this class to convert Text to specific encodings and to convert Text to and from [MemoryBlocks](#).

## Class Summary

Name	Xojo.Core.TextEncoding
Type	Class
Properties	<a href="#">IANAName</a>
Methods	<a href="#">ConvertTextToData</a> <a href="#">ConvertDataToText</a>
Shared Properties	<a href="#">ASCII</a> <a href="#">UTF8</a> <a href="#">UTF16</a> <a href="#">UTF16LittleEndian</a> <a href="#">UTF16BigEndian</a> <a href="#">UTF32</a> <a href="#">UTF32LittleEndian</a> <a href="#">UTF32BigEndian</a> <a href="#">Windows1252</a>
Shared Methods	<a href="#">FromIANAName</a>
Project Types	All
Platforms	All
Related	<a href="#">Locale</a> <a href="#">MemoryBlock</a> <a href="#">Text</a> <a href="#">Text and Encodings</a>

## Properties

### IANAName As Text (read-only)

The encoding's name as specified in [IANA's Character Sets](#) document.

## Methods

## ConvertTextToData(value As Text, allowLossy As Boolean = False) As MemoryBlock

Converts a text value to bytes in a MemoryBlock.

### Parameters

<i>value</i>	The Text value to convert.
<i>allowLossy</i>	When True, any characters that cannot be represented are replaced with a question mark.

### Return Value

Returns a MemoryBlock containing the converted data.

### Notes

If the *value* cannot be represented accurately in this encoding, an exception is raised. The *allowLossy* parameter can be used to override this behavior. If it is True, any characters that cannot be represented are replaced with a question mark. For example, Emoji is not representable in the ASCII encoding.

If the encoding is UTF-16 or UTF-32, the resulting data will begin with a byte order mark. If the byte order mark is not desired, the encoding should specify an explicit big endian or little endian (e.g. using UTF32LittleEndian).

### Exceptions

RuntimeException	If <i>value</i> cannot be represented accurately in the encoding and <i>allowLossy</i> is False.
------------------	--

### Example

Convert text to UTF-8 data:

```
Dim t As Text = "Jåbberwøcky"
```

```
Dim utf8Data As Xojo.Core.MemoryBlock
utf8Data = Xojo.Core.TextEncoding.UTF8.ConvertTextToData(t)
```

## ConvertDataToText(data As MemoryBlock, allowLossy As Boolean = False) As Text

Converts a chunk of data in a MemoryBlock to Text.

### Parameters

<i>data</i>	A <u>MemoryBlock</u> containing the data to convert.
<i>allowLossy</i>	When True, any characters that cannot be represented are replaced with the Unicode replacement character (U+FFFD).

### Return Value

Returns a Text value of the converted data.

## Notes

Converts a chunk of data to Text. If the data is not valid for this encoding (e.g. overlong UTF-8 sequences), an exception is raised. The `allowLossy` parameter can be used to override this behavior. If it is True, any invalid input is replaced with the Unicode replacement character (U+FFFD).

## Exceptions

<code>NilObjectException</code>	If <i>data</i> is Nil.
<code>RuntimeException</code>	If <i>data</i> is not valid for the encoding and <i>allowLossy</i> is False.
<code>RuntimeException</code>	If <i>data</i> has an unknown size.

## Example

This example takes data that came in on a `MemoryBlock` from an `HTTPSocket.PageReceived` event and converts it to UTF8 Text:

```
Dim jsonData As Text = TextEncoding.UTF8.ConvertDataToText(content)
```

## Shared Properties

### ASCII As TextEncoding (read-only)

Returns the ASCII text encoding.

---

### UTF8 As TextEncoding (read-only)

Returns the UTF-8 text encoding.

---

### UTF16 As TextEncoding (read-only)

Returns the UTF-16 text encoding. This is an alias for either `UTF16LittleEndian` or `UTF16BigEndian` depending on the endianness of the target.

---

### UTF16LittleEndian As TextEncoding (read-only)

Returns the UTF-16 text encoding with each code unit stored as little endian.

---

### UTF16BigEndian As TextEncoding (read-only)

Returns the UTF-16 text encoding with each code unit stored as big endian.

## UTF32 As TextEncoding (read-only)

Returns the UTF-32 text encoding. This is an alias for either UTF32LittleEndian or UTF32BigEndian depending on the endianness of the target.

## UTF32LittleEndian As TextEncoding (read-only)

Returns the UTF-32 text encoding with each code unit stored as little endian.

## UTF32BigEndian As TextEncoding (read-only)

Returns the UTF-32 text encoding with each code unit stored as big endian.

## Windows1252 As TextEncoding (read-only)

Returns the Windows 1252 (ISO Latin-1) text encoding.

## Shared Methods

### FromIANAName(name As Text) As TextEncoding

Returns an encoding given its IANA *name* as specified in [IANA's Character Sets](#) document.

#### Parameters

<i>name</i>	The IANA name to find.
-------------	------------------------

#### Return Value

Returns the appropriate TextEncoding.

#### Exceptions

RuntimeException	If <i>name</i> is empty.
RuntimeException	If <i>name</i> is invalid or unsupported.

#### Example

Gets the US ASCII encoding from its name:

```
Dim encoding As Xojo.Core.TextEncoding
encoding = Xojo.Core.TextEncoding.FromIANAName("US-ASCII")
```

# Timer

Provides a way to run code at specified intervals.

## Class Summary

Name	Xojo.Core.Timer
Type	Class
Enumerations	<a href="#">Modes</a>
Events	<a href="#">Action</a>
Properties	<a href="#">Mode</a> <a href="#">Period</a> <a href="#">Tolerance</a>
Shared Methods	<a href="#">CallLater</a> <a href="#">CancelCall</a>
Project Types	All
Platforms	All
Related	<a href="#">AddHandler</a> , <a href="#">RemoveHandler</a>

## Enumerations

### Modes

Specifies when the timer gets called.

Off	Disables the Timer. The action event handler is no longer called.
Single	Calls the Action event handler once after the Period is reached, then turns the Timer to Off.
Multiple	Call the Action event handler each time the Period is reached.

## Events

### Action

Called when the Timer interval is reached, based on the Mode and Period properties.

### Notes

Timer code always runs in the main thread, shared with UI updates. A long-running process called by a Timer can make the UI unresponsive. In these situations, use a Thread instead.

## Sample Code

This code updates a ProgressBar:

```
ProgressBar1.Value = ProgressBar1.Value + 1

// Stop Timer when ProgressBar reaches maximum
If ProgressBar1.Value > ProgressBar1.MaxValue Then
    Me.Mode = Xojo.Core.Timer.Modes.Off
End If
```

## Properties

### Mode As Modes

Specifies if the timer calls the Action event handler once, multiple times or not at all using the Modes enumeration.

#### Example

Set the Mode to Multiple:

```
Timer1.Mode = Xojo.Core.Timer.Modes.Multiple
```

---

### Period As Integer

Specifies the number of milliseconds between calls to the Action event handler.

#### Notes

A period of 0 means to start the timer immediately on the next event loop and is useful for creating a Timer that runs as soon as possible.

#### Sample Code

Set the period to 1 second:

```
Timer1.Period = 1000</code>
```

---

### Tolerance As Integer

A hint to the system as to how precise (in milliseconds) you need the timer to be, although you may get less tolerance than this depending on what the operating system supports. A value of 0 (the default) means to use the standard for the platform. A value of 100 asks for 100 millisecond tolerance.

## Shared Methods

### CallLater(afterMsec As Integer, method As Xojo.Core.Timer.CallNoParams)

Used to call a method (without parameters) once after the specified delay in milliseconds.

#### Parameters

<i>afterMsec</i>	The amount of milliseconds to wait before calling the method.
<i>method</i>	A delegate to the method to call.

#### Sample Code

Suppose you want to display some help text for a few seconds and then hide it. You can do this by creating a method to clear a Label (ClearLabel):

```
Sub ClearLabel
  MyLabel.Text = ""
End Sub
```

In the initial method, you set the Label help text and then use CallLater to set it to clear it after 2 seconds:

```
MyLabel.Text = "Help text goes here"
Xojo.Core.Timer.CallLater(2000, AddressOf ClearLabel)
```

### CallLater(afterMsec As Integer, method As Xojo.Core.Timer.CallWithArg, argument As Auto)

Used to call a method (with a parameter) once after the specified delay in milliseconds.

#### Parameters

<i>afterMsec</i>	The amount of milliseconds to wait before calling the method.
<i>method</i>	A delegate to the method to call.
<i>argument</i>	A Dynamic value that serves as a parameter to the delegate method.

#### Sample Code

Suppose you want to display some help text for a few seconds and then replace it with different text. You can do this by creating a method that takes the text to display as a parameter (SetLabel):

```
Sub SetLabel(helpText As Auto)
  MyLabel.Text = helpText
End Sub</code>
```

In the initial method, you set up the Label help text and use CallLater to change it after 2 seconds:

```
MyLabel.Text = "First help text goes here"  
Xojo.Core.Timer.CallLater(2000, AddressOf SetLabel, "Second help text goes here")
```

---

## CancelCall(method As Xojo.Core.Timer.CallNoParams)

Used to cancel a previously scheduled CallLater that has not yet been run.

### Parameters

method	A delegate to the method whose callback is to be cancelled.
--------	---

### Sample Code

Cancel the ClearLabel callback:

```
Xojo.Core.Timer.CancelCall(AddressOf ClearLabel)
```

---

## CancelCall(method As Xojo.Core.Timer.CallWithArg)

Used to cancel a previously scheduled CallLater that has not yet been run.

### Parameters

method	A delegate to the method whose callback is to be cancelled.
--------	---

### Sample Code

Cancel the SetLabel callback:

```
Xojo.Core.Timer.CancelCall(AddressOf SetLabel)
```

# TimeZone

Used to identify a time zone in relationship to a Date.

## Class Summary

Name	Xojo.Core.TimeZone
Type	Class
Properties	<a href="#">Abbreviation</a> <a href="#">SecondsFromGMT</a>
Shared Method	<a href="#">Current</a>
Project Types	All
Platforms	All
Related	<a href="#">Date</a>

## Constructors

### Constructor(gmtOffsetInSeconds As Integer)

Creates a time zone using the specified GMT offset.

### Constructor(name As Text)

Creates a time zone using the specified name. If the name is invalid, then an `InvalidArgumentException` is raised.

## Properties

### Abbreviation As Text (read-only)

The abbreviation for the time zone.

### SecondsFromGMT As Integer (read-only)

The GMT offset in seconds.

## Shared Methods

### Current As TimeZone

The current time zone.

# WeakRef

The WeakRef class is used to store references to an object without maintaining a strong reference that would force the target object to remain in memory instead of getting removed by reference counting.

## Class Summary

Name	Xojo.Core.WeakRef
Type	Class
Properties	<a href="#">Value</a>
Shared Methods	<a href="#">Create</a>
Project Types	All
Platforms	All
Related	

## Properties

### Value As Object (read-only)

The object that is wrapped by the WeakRef object. If the object has been destructed, this returns Nil.

## Shared Methods

### Create(value As Object) As WeakRef

Creates a WeakRef that wraps the given object.

#### Parameters

<i>value</i>	The object to wrap as a WeakRef.
--------------	----------------------------------

#### Exceptions

NilObjectException	If <i>value</i> is Nil.
--------------------	-------------------------

#### Example

Create a WeakRef of an object instance:

```
Dim ref As WeakRef
ref = WeakRef.Create(myObject)
```

Later you can test if there are references remaining by using the Value property:

```
If ref.Value <> Nil Then
  myObject2 = MyObject(ref.Value)
Else
  // no more references
End If
```

# BadDataException

Raised when the data passed into a function is invalid.

## Class Summary

Name	Xojo.Core.BadDataException
Type	Class
Super	<a href="#">RuntimeException</a>
Properties	<a href="#">ErrorNumber</a> <a href="#">Message</a> <a href="#">Reason</a>
Methods	<a href="#">CallStack</a> <a href="#">Stack</a>
Project Types	All
Platforms	All
Related	

## Notes

This typically only applies to the actual data, such as the bytes passed into [TextEncoding.ConvertDataToText](#).

# ErrorException

Represents an exception that is not necessarily caused by programmer error but rather something from the OS. A good example is trying to write to a read-only file. Checking the writability in advance introduces a 'time of check to time of use' race condition, so the correct thing actually is to just open it.

## Class Summary

Name	Xojo.Core.ErrorException
Type	Class
Super	<a href="#">RuntimeException</a>
Properties	<a href="#">ErrorDomain</a> <a href="#">ErrorNumber</a> <a href="#">Handle</a> <a href="#">LocalizedReason</a> <a href="#">Message</a> <a href="#">Reason</a> <a href="#">UnderlyingError</a>
Methods	<a href="#">CallStack</a> <a href="#">Stack</a>
Project Types	All
Platforms	All
Related	

## Notes

## Constants

POSIXErrorDomain	POSIX	The error domain for error codes coming from POSIX functions in the form of errno values (OS X and Linux).
WindowsErrorDomain	Windows	The error domain for error codes coming from Windows functions in the form of HRESULT values.
XojoErrorDomain	Xojo	The error domain for the framework's cross platform error codes.

## Properties

### ErrorDomain As Text (read-only)

The error domain for the framework's cross platform error codes, specified using the constants POSIXErrorDomain,

WindowsErrorDomain and XojoErrorDomain.

---

### Handle As Ptr (read-only)

If this exception was created from an NSError on iOS or OS X, this provides access to the NSError.

---

### LocalizedReason As Text (read-only)

When available, a localized and user-presentable error message. If no suitable localized message is known, this will be empty.

---

### UnderlyingError As RuntimeException (read-only)

The lower level error, if any that triggered this error.

# InvalidArgumentException

Raised when an invalid argument is passed to a function.

## Class Summary

Name	Xojo.Core.InvalidArgumentException
Type	Class
Super	<a href="#">RuntimeException</a>
Properties	<a href="#">ErrorNumber</a> <a href="#">Message</a> <a href="#">Reason</a>
Methods	<a href="#">CallStack</a> <a href="#">Stack</a>
Project Types	All
Platforms	All
Related	

# IteratorException

Raised when an invalid operation is performed on an iterator. For example, invoking Value before calling MoveNext.

## Class Summary

Name	Xojo.Core.IteratorException
Type	Class
Super	<a href="#">RuntimeException</a>
Properties	<a href="#">ErrorNumber</a> <a href="#">Message</a> <a href="#">Reason</a>
Methods	<a href="#">CallStack</a> <a href="#">Stack</a>
Project Types	All
Platforms	All
Related	<a href="#">Iterator</a>

# LogicException

A runtime exception.

## Class Summary

Name	Xojo.Core.LogicException
Type	Class
Super	<a href="#">RuntimeException</a>
Properties	<a href="#">ErrorNumber</a> <a href="#">Message</a> <a href="#">Reason</a>
Methods	<a href="#">CallStack</a> <a href="#">Stack</a>
Project Types	All
Platforms	All
Related	

# UnsupportedOperationException (Class)

An unsupported operation occurred.

## Class Summary

Name	Xojo.Core.UnsupportedOperationException
Type	Class
Inherits	<a href="#">LogicException</a>
Implements	n/a
Properties	<a href="#">ErrorNumber</a> <a href="#">Message</a> <a href="#">Reason</a>
Methods	<a href="#">CallStack</a> <a href="#">Stack</a>
Project Types	All
Platforms	All
Related	

# Xojo.Crypto

This namespace contains hashing methods for use with cryptography.

## Namespace Summary

Name	Xojo.Crypto
Type	Namespace
Enumerations	<a href="#">HashAlgorithms</a>
Methods	<a href="#">BERDecodePrivateKey</a> <a href="#">BERDecodePublicKey</a> <a href="#">DEREncodePrivateKey</a> <a href="#">DEREncodePublicKey</a> <a href="#">GenerateRandomBytes</a> <a href="#">Hash</a> <a href="#">HMAC</a> <a href="#">MD5</a> <a href="#">PBKDF2</a> <a href="#">RSADecrypt</a> <a href="#">RSAEncrypt</a> <a href="#">RSAGenerateKeyPair</a> <a href="#">RSASign</a> <a href="#">RSAVerifyKey</a> <a href="#">RSAVerifySignature</a> <a href="#">SHA1</a> <a href="#">SHA256</a> <a href="#">SHA512</a>
Classes	<a href="#">CryptoException</a>
Project Types	All
Platforms	All
Related	<a href="#">MemoryBlock</a> <a href="#">Using</a>
Example Projects	Examples/iOS/Framework/CryptoExample

## Notes

Uses [Crypto++ Library 5.6.2](#).

## Enumerations

## HashAlgorithms

Used by the [HMAC](#), [Hash](#) and [PBKDF2](#) methods to specify the type of algorithm to use.

MD5	<a href="#">MD5</a> message-digest algorithm.
SHA1	<a href="#">SHA-1</a> produces a 160-bit (20-byte) hash value.
SHA256	<a href="#">SHA-2</a> with 256 bit digest.
SHA512	<a href="#">SHA-2</a> with 512 bit digest.

## Methods

### BERDecodePrivateKey(privateKey As MemoryBlock) As MemoryBlock

Decodes a private key for interoperability with the BER encoding for use by other libraries.

#### Notes

For more information, refer to [BER and DER Encoding](#).

---

### BERDecodePublicKey(publicKey As MemoryBlock) As MemoryBlock

Decodes a public key for interoperability with the BER encoding for use by other libraries.

#### Notes

For more information, refer to [BER and DER Encoding](#).

---

### DEREncodePrivateKey(privateKey As MemoryBlock) As MemoryBlock

Encodes a private key for interoperability with the DER encoding for use by other libraries.

#### Notes

For more information, refer to [BER and DER Encoding](#).

---

### DEREncodePublicKey(publicKey As MemoryBlock) As MemoryBlock

Encodes a public key for interoperability with the DER encoding for use by other libraries.

#### Notes

For more information, refer to [BER and DER Encoding](#).

---

---

## GenerateRandomBytes(byteCount As UInteger) As MemoryBlock

Generates a random block of data of the specified *byteCount*.

---

## Hash(data As MemoryBlock, algorithm As HashAlgorithms) As MemoryBlock

Creates a hash value for the data using the specified *algorithm*.

### Sample Code

```
Using Xojo.Core
Using Xojo.Crypto
Dim hash As MemoryBlock
hash = Hash("YourPasswordSentence", Crypto.HashAlgorithms.SHA512)
```

---

## HMAC(key As MemoryBlock, data As MemoryBlock, algorithm As HashAlgorithms) As MemoryBlock

Creates the hash-based message authentication code using the *data*, the supplied *key* value and the supplied *algorithm*.

### Notes

The key value is applied to the data before generating the hash. Refer to [HMAC on wikipedia](#).

---

## MD5(data As MemoryBlock) As MemoryBlock

Generates the MD5 message-digest value of the *data*.

---

## PBKDF2(salt As MemoryBlock, data As MemoryBlock, iterations As UInt32, desiredHashLength As UInteger, hashAlgorithm As HashAlgorithms) As MemoryBlock

Returns the PBKDF2 hash value of the *data*, first applying the *salt* value and using the specified *hashAlgorithm*. The *iterations* parameter is the number of loops that the hash algorithm does. The *desiredHashLength* parameter lets you specify the number of bytes that you want the resulting hash to be. 16 or 32 bytes are commonly used.

### Notes

PBKDF2 is a "slow", i.e. deliberately processing intensive, algorithm for generating hash values. Slow is relative, for generating a single hash value it is plenty fast. The benefit of a slow algorithm is that it is impractical for hackers to generate hash tables using it because it would take too long to generate the thousands of hashes for commonly used values.

Use a higher value for iterations' to further slow the hash creation.

Refer to [PBKDF2](#) on Wikipedia.

### Sample Code

```
Using Xojo.Core
Using Xojo.Crypto

Dim salt As Text = "SaltValue"
Dim saltMB As MemoryBlock
saltMB = Xojo.Core.TextEncoding.UTF8.ConvertTextToData(salt)

Dim password As Text = "YourPasswordSentence"
Dim passwordMB As MemoryBlock
passwordMB = Xojo.Core.TextEncoding.UTF8.ConvertTextToData(password)

Dim hash As MemoryBlock
hash = PBKDF2(saltMB, passwordMB, 100, 32, HashAlgorithms.SHA512)
```

## RSADecrypt(data As MemoryBlock, privateKey As MemoryBlock) As MemoryBlock

Decrypts *data* using the specified *privateKey*.

### Exceptions

NilObjectException	If data or privateKey are Nil.
RuntimeException	If data or privateKey have unknown size.

## RSAEncrypt(data As MemoryBlock, publicKey As MemoryBlock) As MemoryBlock

Encrypts the *data* using the specified *publicKey*.

### Exceptions

CryptoException	If you attempt to encrypt using a private key.
-----------------	--

## RSAGenerateKeyPair(bits As UInteger, ByRef privateKey As MemoryBlock, ByRef publicKey As MemoryBlock) As Boolean

Generates a private and public key pair that is hex encoded.

### Returns

True if the keys were successfully generated, False if they were not.

## Notes

You will typically use a *bits* value of 1024 or 2048.

---

## RSASign(data As MemoryBlock, privateKey As MemoryBlock) As MemoryBlock

Signs the data block using the specified *privateKey*.

### Example

Sign a message:

```
Using Xojo.Core
Using Xojo.Crypto
□
Dim privateKey As MemoryBlock
Dim publicKey As MemoryBlock

If RSAGenerateKeyPair( 1024, privateKey, publicKey ) Then
  // 1024-bit private and public keys were generated

  Dim msg As MemoryBlock = TextEncoding.UTF8.ConvertTextToData("this is a test")

  Dim signature As MemoryBlock = RSASign( msg, privateKey )
  If signature <> Nil Then
    // msg was successfully signed
  End If
End If
```

---

## RSASign(key As MemoryBlock) As Boolean

Attempts to validate the specified *key*.

### Returns

True if the signature is valid, False if it is not.

---

## RSASign(signature As MemoryBlock, data As MemoryBlock, publicKey As MemoryBlock) As Boolean

Verifies the data using the specified signature and key.

### Returns

True if the signature is valid, False if it is not.

---

## SHA1(data As MemoryBlock) As MemoryBlock

Generates the SHA1 hash value for *data*.

---

## SHA256(data As MemoryBlock) As MemoryBlock

Generates SHA256 hash value for *data*.

### Sample Code

```
Dim t As Text = "TestData"

// Convert text to a MemoryBlock
Dim textData As Xojo.Core.MemoryBlock
textData = Xojo.Core.TextEncoding.UTF8.ConvertTextToData(t)

// Create SHA256
Dim sha1 As Xojo.Core.MemoryBlock
sha1 = Xojo.Crypto.SHA256(textData)

// Convert SHA256 to Text
// Note that this contains invalid UTF8 characters so really
// cannot be displayed
Dim sha1Text As Text
sha1Text = Xojo.Core.TextEncoding.UTF8.ConvertDataToText(sha1, True)
```

## SHA512(data As MemoryBlock) As MemoryBlock

Generates SHA512 hash value for *data*.

---

# CryptoException

This exception is raised by the various Crypto methods when an error occurs.

## Class Summary

Name	Xojo.Crypto.CryptoException
Type	Class
Super	<a href="#">RuntimeException</a>
Project Types	All
Platforms	All
Related	

# Xojo.Data

This namespace contains methods and classes used for handling data.

## Namespace Summary

Name	Xojo.Data
Type	Namespace
Methods	<a href="#">GenerateJSON</a> <a href="#">ParseJSON</a>
Classes	<a href="#">InvalidJSONException</a>
Project Types	All
Platforms	All
Related	<a href="#">Auto</a> <a href="#">Dictionary</a> <a href="#">FolderItem</a> <a href="#">Using</a> <a href="#">JSON Video</a>

## Methods

### GenerateJSON(value As [Auto](#)) As Text

Generates JSON text from the supplied value, which is typically a Dictionary, an array of Dictionaries or an array of primitive data types (Integer, Text, etc.).

#### Notes

The order of data in the Dictionary is not guaranteed to match the order in the generated JSON.

You can actually supply a single value that is not a Dictionary or an array and get it back as a single valid JSON object (it will encode quotes in text, for example), but those are not valid JSON by themselves.

[Currency](#) is not a valid type that can be converted to JSON.

#### Exceptions

<a href="#">InvalidArgumentException</a>	If one of the types in <i>value</i> cannot be converted to a <a href="#">JSON data type</a> .
--	---

#### Sample Code

Convert a simple array of Text to JSON data:

```
Dim values() As Text = Array("Red Sox", "Yankees", "Orioles", "Blue Jays", "Rays")
```

```
Dim json As Text = Xojo.Data.GenerateJSON(values)
' ["Red Sox","Yankees","Orioles","Blue Jays","Rays"]
```

Convert a Dictionary of values to JSON data:

```
Dim d As New Xojo.Core.Dictionary
d.Value("Team") = "Red Sox"
d.Value("City") = "Boston"
```

```
Dim json As Text
json = Xojo.Data.GenerateJSON(d)
' {"City":"Boston","Team":"Red Sox"}
```

If you want to store an array of information, create an array of Dictionaries and use that to generate the JSON:

```
Dim dictArray() As Xojo.Core.Dictionary
Dim d As Xojo.Core.Dictionary
```

```
d = New Xojo.Core.Dictionary
d.Value("Team") = "Red Sox"
d.Value("City") = "Boston"
dictArray.Append(d)
```

```
d = New Xojo.Core.Dictionary
d.Value("Team") = "Yankees"
d.Value("City") = "New York"
dictArray.Append(d)
```

```
Dim json As Text
json = Xojo.Data.GenerateJSON(dictArray)
' [{"City":"Boston","Team":"Red Sox"},{"City":"New York","Team":"Yankees"}]
```

## ParseJSON(json As Text) As Auto

Parses JSON text and returns it as an Auto, which will typically contain a Dictionary, an array of Dictionaries or an array of primitive data types (Integer, Text, etc.) depending on the JSON text. If the returned value contains a Dictionary, it is case-sensitive.

### Notes

The order of information in the JSON text is not guaranteed to match the order of the information in the resulting Dictionary. Order is retained for arrays.

### Exceptions

<b>InvalidJSONException</b>	If the supplied <i>json</i> Text is not valid JSON data.
-----------------------------	--

## Sample Code

Convert JSON data (in array form) back to an array:

```
' json contains actual JSON data:
' ["Red Sox","Yankees","Orioles","Blue Jays","Rays"]
Dim names() As Auto
names = Xojo.Data.ParseJSON(json)
```

```
Dim teamNames() As Text
For Each name As Text In names
    teamNames.Append(name)
Next
```

Convert JSON data to a Dictionary:

```
' jsonText contains actual JSON data:
' {"City":"Boston","Team":"Red Sox"}
Dim dict As Dictionary
dict = Xojo.Data.ParseJSON(jsonText)
```

To load a JSON data array into an array of Dictionaries:

```
' jsonData contains the actual JSON data:
' [{"City":"Boston","Team":"Red Sox"},{"City":"New York","Team":"Yankees"}]
Dim dictionaries() As Auto
dictionaries = Xojo.Data.ParseJSON(jsonData)
```

```
' Now loop through the array
Dim dict As Xojo.Core.Dictionary
Dim value As Text
For Each d As Xojo.Core.Dictionary In dictionaries
    value = d.Value("Team")
Next
```

To get a value that is within an object in the JSON data, you assign the object to a Dictionary and then reference the value it contains:

```
' jsonData contains the actual JSON data:
' {"team":"Red Sox","topplayer":{"name":"David Ortiz", "position":"DH", "uniform
number":34}}
```

```
Dim d As Xojo.Core.Dictionary
d = Xojo.Data.ParseJSON(jsonData)
Dim topPlayer As Xojo.Core.Dictionary = d.Value("topplayer")
Dim playerName As Text = topPlayer.Value("name")
```

# InvalidJSONException (Class)

Raised when the JSON is invalid.

## Class Summary

Name	Xojo.Core.InvalidJSONException
Type	Class
Super	<a href="#">RuntimeException</a>
Properties	<a href="#">CallStack</a> , <a href="#">Reason</a>
Project Types	All
Platforms	All
Related	<a href="#">GenerateJSON</a> , <a href="#">ParseJSON</a>

# Xojo.Introspection

This namespace contains methods and classes used for introspection.

## Namespace Summary

Name	Xojo.Introspection
Type	Namespace
Classes	<a href="#">AttributeInfo</a> <a href="#">ConstructorInfo</a> <a href="#">MemberInfo</a> <a href="#">MethodInfo</a> <a href="#">ParameterInfo</a> <a href="#">PropertyInfo</a> <a href="#">TypeInfo</a>
Methods	<a href="#">GetType</a>
Project Types	All
Platforms	All
Related	<a href="#">Using</a>
Example Projects	Examples/iOS/Framework/IntrospectionExample

## Methods

### `GetType(obj As Auto) As TypeInfo`

Gets a `TypeInfo` object that describes the supplied object. Before you can do anything with introspection, you need to use this method to get a `TypeInfo` object.

#### Notes

You can supply any data type, including simple types (Integer, Text, etc.), object instances, arrays and structures.

Each `TypeInfo` instance that is returned is unique and immutable, so two objects of the same class will always return the same `TypeInfo` instance.

#### Example

Gets the `TypeInfo` for an object and displays its name:

```
Using Xojo.Introspection
Dim info As TypeInfo = GetType(Self)
Label1.Text = "Object name: " + info.Name
```

# AttributeInfo

Provides information on the attributes of a class instance, method or property using introspection.

## Class Summary

Name	Xojo.Introspection.AttributeInfo
Type	Class
Properties	<a href="#">Name</a> <a href="#">Value</a>
Project Types	All
Platforms	All
Related	<a href="#">ConstructorInfo</a> <a href="#">MethodInfo</a> <a href="#">PropertyInfo</a> <a href="#">TypeInfo</a>

## Notes

Attributes are added to project items, methods, properties, constants, etc. by using the "Advanced" tab of the Inspector.

## Properties

### Name As Text

The name of the attribute.

### Example

This code gets the names of all the attributes on Class1:

```
Using Xojo.Introspection
```

```
Dim obj As New Class1
Dim info As TypeInfo = GetType(obj)
Dim attrs() As AttributeInfo = info.GetAttributes
```

```
Dim attributeNames() As Text
For Each a As AttributeInfo In attrs
  attributeNames.Append(a.Name)
Next
```

## Value As Auto

The optional value of the attribute.

# ConstructorInfo

Provides information on the constructors of a class instance using introspection.

## Class Summary

Name	Xojo.Introspection.ConstructorInfo
Type	Class
Super	<a href="#">MemberInfo</a>
Properties	<a href="#">IsGlobal</a> <a href="#">IsPrivate</a> <a href="#">IsProtected</a> <a href="#">IsPublic</a> <a href="#">Name</a>
Methods	<a href="#">GetAttributes</a> <a href="#">Invoke</a> <a href="#">Parameters</a>
Project Types	All
Platforms	All
Related	

## Methods

### Invoke(Optional params() As Auto) As Auto

Creates a new instance and invokes the target constructor on the class instance, passing in the specified parameters.

#### Parameters

params()	An array of parameters to pass to the constructor.
----------	--

#### Returns

An Auto containing the data type of the object.

#### Exceptions

<a href="#">OutOfBoundsException</a>	If the number of parameters do not match.
<a href="#">IllegalCastException</a>	If the types of parameters do not match.

# MemberInfo

The superclass for many of the Introspection classes.

## Class Summary

Name	Xojo.Introspection.MemberInfo
Type	Class
Properties	<a href="#">IsPrivate</a> <a href="#">IsProtected</a> <a href="#">IsPublic</a> <a href="#">Name</a>
Methods	<a href="#">GetAttributes</a> <a href="#">Parameters</a>
Project Types	All
Platforms	All
Related	<a href="#">ConstructorInfo</a> <a href="#">MethodInfo</a> <a href="#">PropertyInfo</a>

## Properties

### IsPrivate As Boolean

Checks if the item has private scope.

---

### IsProtected As Boolean

Checks if the item has protected scope.

---

### IsPublic As Boolean

Checks if the item has public scope.

---

### Name As Text

The name of the item.

### Sample Code

Gets the names of methods on a class:

Using Xoj.Introspection

```
Dim obj As New Class1
Dim info As TypeInfo = GetType(obj)
Dim methods() As MethodInfo = info.Methods
```

```
Dim methodNames() As Text
For Each m As MethodInfo In methods
    methodNames.Append(m.Name)
Next</code>
```

## Methods

### GetAttributes As AttributeInfo()

Gets the attributes for the associated item.

Returns

Return an array of the attributes.

#### Sample Code

The following code gets the attributes for the window containing the code:

```
Dim myAttributes() As Introspection.AttributeInfo
myAttributes = GetType(Self).GetAttributes
```

---

### Parameters As ParameterInfo()

Gets the parameters for the member.

# MethodInfo

Provides information on the methods of a class instance using introspection.

## Class Summary

Name	Xojo.Introspection.MethodInfo
Type	Class
Super	<a href="#">MemberInfo</a>
Properties	<a href="#">IsGlobal</a> <a href="#">IsPrivate</a> <a href="#">IsProtected</a> <a href="#">IsPublic</a> <a href="#">IsShared</a> <a href="#">Name</a> <a href="#">ReturnType</a>
Methods	<a href="#">GetAttributes</a> <a href="#">Invoke</a> <a href="#">Parameters</a>
Project Types	All
Platforms	All
Related	<a href="#">ParameterInfo</a>

## Properties

### IsShared As Boolean

Checks if this is a shared method.

### ReturnType As [TypeInfo](#)

The return type of the method.

## Methods

### Invoke(target As Object, Optional params() As Auto) As [Auto](#)

Calls the method, given its base object and an instance of the class that created it.

## Invoke(target As Object, Optional params() As Auto)

Calls the function, given its base object and an instance of the class that created it.

### Parameters

<i>target</i>	The base object. Nil may be used to call a shared method.
<i>params</i>	An array of parameters to pass to the method.

### Returns

For a function call, returns an Auto containing the return value of the function.

### Notes

You can use Nil as the *target* object if calling a shared method.

### Exceptions

OutOfBoundsException	If the number of parameters do not match.
IllegalCastException	If the types of parameters do not match.

# ParameterInfo

Provides information on method parameters using introspection.

## Class Summary

Name	Xojo.Introspection.ParameterInfo
Type	Class
Properties	<a href="#">IsByRef</a> <a href="#">ParameterType</a>
Project Types	All
Platforms	All
Related	<a href="#">MethodInfo</a>

## Notes

You get an array of ParameterInfo objects by calling [MemberInfo.Parameters](#).

## Properties

### IsByRef As Boolean

Checks if the parameter is passed ByRef instead of ByVal.

---

### ParameterType As [TypeInfo](#)

Type information for the parameter.

# PropertyInfo

Provides information on the methods of a class instance using introspection.

## Class Summary

Name	Xojo.Introspection.PropertyInfo
Type	Class
Super	<a href="#">MemberInfo</a>
Properties	<a href="#">CanRead</a> <a href="#">CanWrite</a> <a href="#">IsComputed</a> <a href="#">IsShared</a>
Methods	<a href="#">Value</a>
Project Types	All
Platforms	All
Related	

## Properties

### CanRead As Boolean

The property value can be read, indicating a simple value property or a computed property with a Get accessor.

---

### CanWrite As Boolean

The property value can be written, indicating a simple value property or a computed property with a Set accessor.

---

### IsComputed As Boolean

Checks if the property is a computed property or a value property.

---

### IsShared As Boolean

Checks if this is a shared property.

## Methods

Value(obj As Object) As Auto

Value(obj As Object, Assigns value As Auto)

Gets or sets the value of the property.

### Parameters

<i>obj</i>	The object that contains the property value to access.
<i>value</i>	Used to assign a value.

# TypeInfo

Contains type information for objects, arrays, simple data types and structures.

## Class Summary

Name	Xojo.Introspection.TypeInfo
Type	Class
Super	<a href="#">MemberInfo</a>
Properties	<a href="#">BaseType</a> <a href="#">FullName</a> <a href="#">HasElementType</a> <a href="#">IsArray</a> <a href="#">IsClass</a> <a href="#">IsEnum</a> <a href="#">IsInterface</a> <a href="#">IsPointer</a> <a href="#">IsPrimitive</a> <a href="#">IsPrivate</a> <a href="#">IsProtected</a> <a href="#">IsPublic</a> <a href="#">IsValueType</a> <a href="#">Name</a>
Methods	<a href="#">ArrayElementType</a> <a href="#">ArrayRank</a> <a href="#">GetAttributes</a> <a href="#">Constructors</a> <a href="#">Methods</a> <a href="#">Properties</a> <a href="#">IsSubClassOf</a>
Project Types	All
Platforms	All
Related	

## Properties

### BaseType As TypeInfo

Gets parent (or Super) of the current object. Allows you to navigate the type hierarchy from child to parent.

## FullName As Text

The fully qualified name used as a unique identifier. This can differ from the simple name whenever the type lives in a namespace.

---

## HasElementType As Boolean

Compound data types have their own attributes as well as an element type.

---

## IsArray As Boolean

Indicates the item is an array.

---

## IsClass As Boolean

Indicates the item is a class.

---

## IsEnum As Boolean

Indicates the item is an enumeration.

---

## IsInterface As Boolean

Indicates the item is a class interface.

---

## IsPointer As Boolean

Indicates the item is a Ptr or another external pointer type.

---

## IsPrimitive As Boolean

Indicates the item is a primitive type, which are: Boolean, String, Color, Integer, Double, Single, Currency and other numeric types.

---

## IsValueType As Boolean

Indicates the item is a value type, which means it is not a reference. With value types, assignments copy the value itself rather than the reference to the value. Primitive, pointer, structure and enum types are all value types.

---

---

## Methods

### ArrayElementType As TypeInfo

Get the type information for the array.

Returns

The TypeInfo for the element type.

Exceptions

RuntimeException	When the item is not an array.
------------------	--------------------------------

---

### ArrayRank As Integer

The number of dimensions of an array. For arrays only (isArray = True).

Returns

An integer indicating the number of dimensions of the array.

Notes

Exceptions

RuntimeException	If the item is not an array.
------------------	------------------------------

---

### GetAttributes As AttributeInfo()

Gets the attributes for the item, which must be a class instance.

Returns

An array of AttributeInfo objects.

---

### Constructors As ConstructorInfo()

Gets the constructors for the item, which must be a class instance.

Returns

An array of ConstructorInfo objects.

---

### Methods As MethodInfo()

Gets the methods for the item, which must be a class instance.

**Returns**

An array of MethodInfo objects.

---

**Properties As PropertyInfo()**

Gets the properties for the item, which must be a class instance.

**Returns**

An array of PropertyInfo objects.

---

**IsSubClassOf(c As TypeInfo) As Boolean**

Checks if this item a subclass of the specified type *c*.

**Returns**

True if the item is a subclass of type *c*, False if it is not.

# IO

Contains classes used for file input and output.

## Namespace Summary

Name	Xojo.IO
Type	Namespace
Enumerations	<a href="#">HandleTypes</a>
Classes	<a href="#">BinaryStream</a> <a href="#">FolderItem</a> <a href="#">IOException</a> <a href="#">SpecialFolder</a> <a href="#">TextInputStream</a> <a href="#">TextOutputStream</a>
Project Types	All
Platforms	All
Related	<a href="#">Using</a>

## Enumerations

### HandleTypes

The type of OS handle for the stream.

FileNumber	
FilePointer	
Win32	

# BinaryStream

The BinaryStream class is an input/output mechanism for reading and writing arbitrary binary data. It can be used with MemoryBlocks and FolderItems.

## Class Summary

Name	Xojo.IO.BinaryStream
Type	Class
Constructors	<a href="#">Constructor(MemoryBlock)</a>
Enumerations	<a href="#">LockModes</a>
Properties	<a href="#">IsClosed</a> <a href="#">Length</a> <a href="#">LittleEndian</a> <a href="#">Position</a>
Methods	<a href="#">Close</a> <a href="#">EOF</a> <a href="#">Flush</a> <a href="#">Handle</a> <a href="#">Read</a> <a href="#">ReadBoolean</a> <a href="#">ReadCurrency</a> <a href="#">ReadDouble</a> <a href="#">ReadInt8</a> <a href="#">ReadInt16</a> <a href="#">ReadInt32</a> <a href="#">ReadInt64</a> <a href="#">ReadSingle</a> <a href="#">ReadText</a> <a href="#">ReadUInt8</a> <a href="#">ReadUInt16</a> <a href="#">ReadUInt32</a> <a href="#">ReadUInt64</a> <a href="#">Write</a> <a href="#">WriteBoolean</a> <a href="#">WriteCurrency</a> <a href="#">WriteDouble</a> <a href="#">WriteInt8</a> <a href="#">WriteInt16</a> <a href="#">WriteInt32</a> <a href="#">WriteInt64</a> <a href="#">WriteSingle</a> <a href="#">WriteText</a> <a href="#">WriteUInt8</a> <a href="#">WriteUInt16</a> <a href="#">WriteUInt32</a> <a href="#">WriteUInt64</a>

Shared Methods	<a href="#">Create</a> <a href="#">FromHandle</a> <a href="#">Open</a>
Project Types	All
Platforms	All
Related	<a href="#">FolderItem</a> <a href="#">MemoryBlock</a>

## Notes

The Read/Write methods raise an exception when reading or writing past the end of the stream.

File IO prevents thread switching.

## Sample Code

```
// Load an image that is copied to an iOS app using a Copy Files Build Step
Dim f As FolderItem = SpecialFolder.GetResource("MyImage.JPG")
```

```
Dim b As BinaryStream
b = BinaryStream.Open(f, BinaryStream.LockModes.Read)
```

```
Dim mb As New MemoryBlock(b.Read(b.Length))
```

```
b.Close
```

```
Dim image As UIImage
image = Image.FromData(mb)
```

```
ImageView1.Image = image
```

## Constructors

### Constructor(mb As MemoryBlock)

Creates a BinaryStream from a MemoryBlock.

#### Sample Code

```
// MyImage is an UIImage in the project
Dim binaryData As New BinaryStream(MyImage.ToData("public.PNG"))</code>
```

## Enumerations

## LockModes

The type of locking modes for the binary stream.

Read	Read access.
Write	Write access.
ReadWrite	Read and write access.
Exclusive	Exclusive lock; nothing else can access the file.

## Properties

### IsClosed As Boolean (read-only)

Determines if the binary stream has been closed.

---

### Length As UInt64

Gets or sets the length of the file in bytes. If you set the Length property to a value smaller than its current value, it truncates the file. If the BinaryStream is backed by a MemoryBlock, it returns [MemoryBlock.Size](#). Raises an exception if setting the Length of a BinaryStream backed by a MemoryBlock with unknown size.

---

### LittleEndian As Boolean

Gets or sets the byte order when reading or writing to a BinaryStream. A value of True sets the byte order to little endian.

---

### Position As UInt64

Gets or sets the current position within the BinaryStream. The first position is numbered zero. To move the position to the end of the stream, set the Position to Length.

---

## Methods

### Close

Closes an existing BinaryStream. Raises an exception if the stream is already closed.

---

### EOF As Boolean

Returns True if the BinaryStream position = the file length. If the stream is closed this function always returns True.

---

---

## Flush

Immediately sends the contents of internal write buffers to disk or to the output stream.

---

## Handle(type As HandleTypes) As Ptr

Given a handle type, gets a handle to the current stream . File Descriptor, File Pointer, or a Windows32 OS handle.

---

## Read(count As Integer) As MemoryBlock

Reads the specified number of bytes. Raises an exception if the file is not readable.

---

## ReadBoolean As Boolean

Reads a Boolean from the stream. Raises an exception if the file is not readable.

---

## ReadCurrency As Currency

Reads a Currency from the stream. Raises an exception if the file is not readable.

---

## ReadDouble As Double

Reads a Boolean from the stream. Raises an exception if the file is not readable.

---

## ReadInt16 As Int16

Reads an Int16 from the stream. Raises an exception if the file is not readable.

---

## ReadInt32 As Int32

Reads an Int32 from the stream. Raises an exception if the file is not readable.

---

## ReadInt64 As Int64

Reads an Int64 from the stream. Raises an exception if the file is not readable.

---

## ReadInt8 As Int8

Reads an Int8 from the stream. Raises an exception if the file is not readable.

---

---

## ReadSingle As Single

Reads a Single from the stream. Raises an exception if the file is not readable.

---

## ReadText(count As Integer, encoding As TextEncoding) As Text

Reads the next *count* bytes from the stream. Raises an exception if the file is not readable or if encoding is Nil.

---

## ReadUInt16 As UInt16

Reads a UInt16 from the stream. Raises an exception if the file is not readable.

---

## ReadUInt32 As UInt32

Reads a UInt32 from the stream. Raises an exception if the file is not readable.

---

## ReadUInt64 As UInt64

Reads a UInt64 from the stream. Raises an exception if the file is not readable.

---

## ReadUInt8 As UInt8

Reads a UInt8 from the stream. Raises an exception if the file is not readable.

---

## Write(block As MemoryBlock)

Writes the bytes in the memoryblock. Raises an exception if the stream is not writable or the MemoryBlock has an unknown size.

---

## WriteBoolean(value As Boolean)

Writes a Boolean to the stream. Raises an exception if the stream is not writable.

---

## WriteCurrency(value As Currency)

Writes a Currency to the stream. Raises an exception if the stream is not writable.

---

---

## WriteDouble(value As Double)

Writes a Double to the stream. Raises an exception if the stream is not writable.

---

## WriteInt16(value As Int16)

Writes an Int16 to the stream. Raises an exception if the stream is not writable.

---

## WriteInt32(value As Int32)

Writes an Int32 to the stream. Raises an exception if the stream is not writable.

---

## WriteInt64(value As Int64)

Writes an Int64 to the stream. Raises an exception if the stream is not writable.

---

## WriteInt8(value As Int8)

Writes an Int8 to the stream. Raises an exception if the stream is not writable.

---

## WriteSingle(value As Single)

Writes a Single to the stream. Raises an exception if the stream is not writable.

---

## WriteText(value As Text, encoding As TextEncoding)

Writes the bytes in the Text. Raises an exception if the stream is not writable, or encoding is Nil.

---

## WriteUInt16(value As UInt16)

Writes a UInt16 to the stream. Raises an exception if the stream is not writable.

---

## WriteUInt32(value As UInt32)

Writes a UInt32 to the stream. Raises an exception if the stream is not writable.

---

## WriteUInt64(value As UInt64)

Writes a UInt64 to the stream. Raises an exception if the stream is not writable.

---

## WriteUInt8(value As UInt8)

Writes a UInt8 to the stream. Raises an exception if the stream is not writable.

## Shared Methods

### Create(file As FolderItem) As BinaryStream

Creates a new BinaryStream, bound to the passed File. Raises a RuntimeException if File is Nil, does not exist, is not accessible or another error occurs.

---

### FromHandle(handle As Ptr, type As HandleTypes) As BinaryStream

Creates a new BinaryStream from an OS handle.

---

### Open(file As FolderItem, mode As LockModes) As BinaryStream

Opens the passed FolderItem as a binary stream with the specified lock mode. If an error occurs, an IOException is raised.

#### Sample Code

```
Using Xojo.IO
```

```
Dim f As FolderItem = SpecialFolder.Documents.Child("MyFile.data")
```

```
Dim bStream As BinaryStream
```

```
bStream = BinaryStream.Open(f, LockModes.Read)
```

```
Dim data As MemoryBlock
```

```
data = bStream.Read(bStream.Length)
```

```
bStream.Close
```

# FolderItem

The FolderItem class is used to represent files, applications, folders and other items in the file system.

## Class Summary

Name	Xojo.IO.FolderItem
Type	Class
Interfaces	<a href="#">Iterable</a>
Properties	<a href="#">Count</a> <a href="#">CreationDate</a> <a href="#">DisplayName</a> <a href="#">Exists</a> <a href="#">IsAlias</a> <a href="#">IsExtensionVisible</a> <a href="#">IsFolder</a> <a href="#">IsReadable</a> <a href="#">IsVisible</a> <a href="#">IsWriteable</a> <a href="#">Length</a> <a href="#">ModificationDate</a> <a href="#">Name</a> <a href="#">Parent</a> <a href="#">Path</a> <a href="#">URLPath</a>
Methods	<a href="#">Child</a> <a href="#">Children</a> <a href="#">CopyTo</a> <a href="#">CreateAsFolder</a> <a href="#">Delete</a> <a href="#">MoveTo</a>
Project Types	All
Platforms	All
Related	<a href="#">BinaryStream</a>

## Notes

On iOS, to get a reference to a FolderItem, use [SpecialFolder](#). For example:

```
Dim file As FolderItem = SpecialFolder.Documents("MyFile.txt")
```

## Constructors

### Constructor(path As Text, resolveAlias As Boolean = True)

Creates a new FolderItem using an absolute *path*.

#### Parameters

<i>path</i>	The OS-specific absolute path to the FolderItem to create.
<i>resolveAlias</i>	A boolean that indicates if an alias should be resolved to the file it points to.

#### Note

All components of the path (except the last one) must exist in order for the FolderItem to be created.

#### Exceptions

RuntimeException	If the FolderItem could not be created.
<a href="#">InvalidArgumentException</a>	If <i>path</i> is not an absolute path.

#### Example

Create a FolderItem that points to a file in a folder in the root of the drive:

```
Dim saveFile As New FolderItem("c:\data\savefile.txt")
```

To create a Xojo.IO.FolderItem from a FolderItem returned from a classic framework class or method:

```
Dim userFile As FolderItem = GetOpenFolderItem("")
Dim newFile As New Xojo.IO.FolderItem(userFile.NativePath)</code>
```

## Properties

### Count As UInteger (read-only)

The number of items in the FolderItem if it is a folder. If the item is not a folder then it returns 0.

#### Sample Code

```
Dim f As FolderItem = SpecialFolder.Documents
Dim count As Integer = f.Count</code>
```

---

## CreationDate As Date

The creation date of the FolderItem.

### Notes

In order to change the creation date, you need to create a new Date and then assign it to this property. Changing the CreationDate property directly will not cause the date to actually get changed.

### Exceptions

RuntimeException	If the creation date can not be changed when attempting to set a new date.
------------------	--

### Examples

Get the creation date:

```
Dim createDate As Xojo.Core.Date  
createDate = myFile.CreationDate
```

To change the creation date:

```
Dim newDate As New Xojo.Core.Date(2014, 8, 1)  
myFile.CreationDate = newDate
```

---

## DisplayName As Text (read-only)

The name of the FolderItem as it should be displayed to the user. This is usually the same as the Name property, but you should always use DisplayName rather than Name when displaying the name of the FolderItem to the user.

### Example

Get the display name:

```
Dim name As String  
name = myFile.DisplayName
```

---

## Exists As Boolean (read-only)

Indicates whether the FolderItem exists.

### Notes

You can create a FolderItem that does not refer to actual existing file on disk. When you do so, this property returns False.

## Example

To check if a file exists:

```
Dim f As New Xojo.IO.FolderItem("test.txt")

If f.Exists Then
    // open the file
End If
```

---

## IsAlias As Boolean (read-only)

Indicates whether the FolderItem is a link or shortcut to another FolderItem.

---

## IsExtensionVisible As Boolean (read-only)

Tells you whether the file should have its extension visible or hidden based on the user's OS settings.

---

## IsFolder As Boolean (read-only)

Indicates if the FolderItem is a folder.

---

## IsReadable As Boolean (read-only)

This is True when you have permissions to read from the file or folder. In the case of a folder, it means you can read the contents of the folder.

### Notes

Even if IsReadable is True, it does not provide a guarantee that the read will succeed. If you want to read from a file, you should simply attempt to do so and handle any exceptions that may occur.

---

## IsVisible As Boolean (read-only)

Indicates whether the FolderItem is visible to standard file system tools (Explorer or Finder, for example).

### Sample Code

```
If myFile.Visible Then
    ' open the file
End If</code>
```

---

## IsWriteable As Boolean (read-only)

This is True when you have permissions to write to the file or folder. In the case of a folder, it means you can create files in the folder.

### Notes

Even if IsWriteable is True, an attempt to write may fail for other reasons, such as coding errors or insufficient disk space. For example, a write may fail because disk space runs out midway through the write operation. IsWriteable is not intended to check for these type of conditions. If you intend to write to a file, you should attempt to do so and handle any exceptions that may occur.

---

## Length As UInt64 (read-only)

The size of the file in bytes. For folders, the size is 0.

### Sample Code

```
Dim f As FolderItem = SpecialFolder.Documents("MyFile.txt")
If f.Exists Then
    Dim length As UInt64 = f.Length
End If</code>
```

---

## ModificationDate As Date

The modification date of the FolderItem.

### Notes

In order to change the modification date, you need to create a new Date and then assign it to this property. Changing the ModificationDate property directly will not cause the date to actually get changed.

### Examples

Get the modification date:

```
Dim modDate As Xojo.Core.Date
modDate = myFile.ModificationDate
```

To change the modification date:

```
Dim newDate As New Xojo.Core.Date(2014, 8, 1)
myFile.ModificationDate = newDate
```

---

## Name As Text

The name of the FolderItem. Changing this name renames the file or folder.

### Exceptions

RuntimeException	For these conditions: <ul style="list-style-type: none"> <li>• User does not have permissions to rename the FolderItem</li> <li>• The FolderItem is in use</li> <li>• The name already exists</li> </ul>
------------------	--

### Sample Code

```
Dim f As FolderItem = SpecialFolder.Documents("MyFile.txt")
f.Name = "NewFile.txt" // rename file
```

## Parent As FolderItem (read-only)

This is the parent folder of the FolderItem (based on the file system hierarchy). If the Parent is the root of the file system, this returns Nil.

### Sample Code

```
Dim f As FolderItem = SpecialFolder.Documents
Dim parent As FolderItem = f.Parent</code>
```

## Path As Text (read-only)

The full path to the FolderItem using the path format native to the OS.

### Notes

On OS X and Linux, this returns the POSIX path (separated by slashes).

If the FolderItem is a folder, the Path ends with a backslash on Windows and a forward slash on Linux.

When accessing a drive on Windows that you do not have permissions for or one that does not exist, there is no trailing slash. For example, "f:" is a drive that is not mounted, "f:\" is mounted.

### Sample Code

```
Dim fullPath As Text = myFolderItem.Path</code>
```

## URLPath As Text (read-only)

The URL path of the FolderItem, which uses the form **file://path/to/file**.

## Sample Code

```
Dim urlPath As Text = myFolderItem.URLPath</code>
```

## Methods

### Child(name As Text, resolveAlias As Boolean = True) As FolderItem

Returns a FolderItem with the specified *name* that represents a file or folder within the FolderItem.

<i>name</i>	The name of the child file or folder.
<i>resolveAlias</i>	Indicates if the FolderItem should resolve to its actual item, if the FolderItem is an alias or shortcut.

If the path points to an alias or shortcut, the FolderItem is resolved to the actual item being pointed to only if *resolveAlias* is True.

### Exceptions

RuntimeException	If the child cannot be constructed, which would only be the case if the FolderItem is not a folder, or the last last component of the path does not already exist.
------------------	--

### Example

# IOException

Raised when there is a file input/output error.

## Class Summary

Name	Xojo.Core.IOException
Type	Class
Super	<a href="#">ErrorException</a>
Properties	<a href="#">ErrorNumber</a> <a href="#">Message</a> <a href="#">Reason</a>
Methods	<a href="#">CallStack</a> <a href="#">Stack</a>
Project Types	All
Platforms	All
Related	<a href="#">BinaryStream</a> <a href="#">FolderItem</a>

## Notes

On OS X, [osstatus.com](http://osstatus.com) provides a list of common error codes that could be returned in the ErrorNumber property.

## Sample Code

Catch and display an IOException:

```
Using Xojo.Core
Using Xojo.IO
```

```
Dim f As FolderItem
f = SpecialFolder.Documents.Child("SaveData.txt")
```

```
Dim output As TextOutputStream
Try
    output = TextOutputStream.Append(f, TextEncoding.UTF8)
    output.WriteLine("This text is appended to the end of the file.")
    output.Close
Catch e As IOException
    Label1.Text = "Error: " + e.Reason
End Try
```

# SpecialFolder

The SpecialFolder module provides access to OS-specific folders.

## Module Summary

Name	Xojo.IO.SpecialFolder
Type	Module
Methods	<a href="#">ApplicationSupport</a> <a href="#">Caches</a> <a href="#">Documents</a> <a href="#">GetResource</a> <a href="#">Temporary</a>
Project Types	All
Platforms	All
Related	<a href="#">FolderItem</a>

## Methods

### ApplicationSupport As [FolderItem](#)

On iOS, this is the location of application support files (Library/Application Support).

#### Exceptions

RuntimeException	If the folder does not exist.
------------------	-------------------------------

#### Sample Code

```
Dim appSupport As FolderItem
appSupport = SpecialFolder.ApplicationSupport
```

### Caches As [FolderItem](#)

The OS folder for storing cache data.

#### Exceptions

RuntimeException	If the folder does not exist.
------------------	-------------------------------

#### Sample Code

```
Dim cache As FolderItem
cache = SpecialFolder.Caches
```

---

## Documents As FolderItem

The Document folder is where you will likely want to place most user-created files or data.

### Exceptions

RuntimeException	If the folder does not exist.
------------------	-------------------------------

### Sample Code

```
Dim docFile As FolderItem
docFile = SpecialFolder.Documents.Child("MyDocument.txt")</code>
```

---

## GetResource(name As Text) As FolderItem

Gets the specified file (by name) located in the app's main bundle.

### Notes

Use a Copy Files Build Step to copy files into the app's main bundle.

### Exceptions

InvalidArgumentException	If the file the does exist.
--------------------------	-----------------------------

### Sample Code

```
Dim bundleFile As Folderitem
bundleFile = SpecialFolder.GetResource("AppDatabase.sqlite")
```

---

## Temporary As FolderItem

The user's temporary folder.

### Exceptions

RuntimeException	If the folder does not exist.
------------------	-------------------------------

### Sample Code

```
Dim tmp As FolderItem
tmp = SpecialFolder.Temporary</code>
```

---

</span></font>

# TextInputStream

Used to read text of a specific encoding from a file.

## Class Summary

Name	Xojo.IO.TextInputStream
Type	Class
Properties	<a href="#">BytePosition</a> <a href="#">Encoding</a>
Methods	<a href="#">Close</a> <a href="#">EOF</a> <a href="#">Handle</a> <a href="#">Read</a> <a href="#">ReadAll</a> <a href="#">ReadLine</a>
Shared Methods	<a href="#">Open</a>
Project Types	All
Platforms	All
Related	<a href="#">FolderItem</a> <a href="#">TextOutputStream</a>

## Constructors

### Constructor(handle As Ptr, type As HandleTypes, encoding As TextEncoding)

Creates a new TextInputStream based on the handle that was passed.

#### Exceptions

IOException	If the handle is invalid or the type does not match the handle.
-------------	---

## Properties

### BytePosition As [UInt64](#)

Gets or sets the byte position of the file pointer, not the character position. The first position is numbered zero.

#### Exceptions

IOException	If value > FolderItem.Length.
-------------	-------------------------------

---

## Encoding as TextEncoding

Gets or sets the default encoding for the Read, ReadAll and ReadLine methods.

### Exceptions

IOException	If Encoding is Nil.
-------------	---------------------

### Sample Code

```
Using Xojo.Core  
inputStream.Encoding = TextEncoding.ASCII</code>
```

## Methods

### Close

Closes the stream. The stream is also closed automatically when it goes out of scope.

### Exceptions

IOException	If the stream is not open.
-------------	----------------------------

### Sample Code

```
inputStream.Close</code>
```

---

## EOF As Boolean

Returns True when a Read method reaches the end of the stream.

### Exceptions

IOException	If the stream is not open.
-------------	----------------------------

### Sample Code

```
Dim line As Text  
While Not inputStream.EOF  
  line = inputStream.ReadLine  
Wend
```

## Handle(type As HandleTypes) As Ptr

Returns a handle to the stream for use by OS API calls.

### Exception

IOException	If the stream is not open.
-------------	----------------------------

## Read(numberOfBytes As UInt64) As Text

Reads the specified number of bytes from the stream and returns them as a Text object.

### Exceptions

IOException	If there is not enough memory available or the stream is not open.
-------------	--

## ReadAll As Text

Reads all remaining bytes from the stream and returns them as a Text object.

### Exceptions

IOException	If there is not enough memory available or the stream is not open.
-------------	--

### Sample Code

```
Dim allText As Text
allText = inputStream.ReadAll
```

## ReadLine As Text

Reads bytes from the stream until and EOL character is encountered and returns them as a Text object.

### Exceptions

IOException	If there is not enough memory available or the stream is not open.
-------------	--

### Sample Code

```
Dim line As Text
line = inputStream.ReadLine</code>
```

## Shared Methods

## Open(file As FolderItem, encoding As TextEncoding) As TextInputStream

Creates a TextInputStream and connects it to the specified file.

### Exceptions

IOException	When file = Nil, file does not exist or file is not readable.
-------------	---

### Sample Code

```
Using Xojo.Core
Using Xojo.IO

Dim f As FolderItem
f = SpecialFolder.Documents.Child("SaveData.txt")

If Not f.Exists Then
    // Cannot read the file if it does not exist
    Return
End If

Dim errorText As Text
Try
    Dim input As TextInputStream
    input = TextInputStream.Open(f, TextEncoding.UTF8)
    TextFileArea.Text = input.ReadAll
    input.Close
Catch e As IOException
    errorText = "File IO Error: " + e.Reason
End Try
```

# TextOutputStream

Used for outputting text of a specific encoding to files.

## Class Summary

Name	Xojo.IO.TextOutputStream
Type	Class
Properties	<a href="#">Delimiter</a> <a href="#">Encoding</a>
Methods	<a href="#">Close</a> <a href="#">Flush</a> <a href="#">Handle</a> <a href="#">Write</a> <a href="#">WriteLine</a>
Shared Methods	<a href="#">Append</a> <a href="#">Create</a>
Project Types	All
Platforms	All
Related	<a href="#">FolderItem</a> <a href="#">TextInputStream</a>

## Properties

### Delimiter As [Text](#)

The character used to mark the end of a line of text written to the file. If this value is not set, the OS default for EndOfLine is used.

#### Sample Code

```
// Set the delimiter to Tab  
outputStream.Delimiter = &u09</code>
```

---

### Encoding As [TextEncoding](#)

The encoding used for writing the text. Raises an exception if set to Nil.

#### Sample Code

Using Xojo.Core

```
// Change the encoding to ASCII  
outputStream.Encoding = TextEncoding.ASCII</code>
```

## Methods

### Close

Closes the TextOutputStream. The stream is also closed automatically when it goes out of scope.

Sample Code

```
outputStream.Close</code>
```

---

### Flush

Immediately sends the contents of internal write buffers to disk or to the output stream.

Sample Code

```
outputStream.Flush
```

---

### Handle(type As HandleTypes) As Ptr

Handle returns a handle of the Type passed or -1 if the requested type cannot be retrieved

---

### Write(data As Text)

Writes the passed data to the output stream.

Sample Code

```
Dim outputText As Text = "This is a test."  
outputStream.Write(outputText)
```

---

### WriteLine(data As Text)

Writes the passed data to the output stream followed by the character(s) defined in Delimiter.

Sample Code

```
Dim outputText As Text = "This is a test."
```

```
outputStream.WriteLine("1: " + outputText)
outputStream.WriteLine("2: " + outputText)
```

## Shared Methods

### Append(f As FolderItem, encoding As TextEncoding) As TextOutputStream

Opens the passed file so that text can be appended to existing text.

#### Exceptions

IOException	For these conditions: <ul style="list-style-type: none"> <li>• file does not exist</li> <li>• file is not writeable</li> <li>• containing folder is not writeable</li> <li>• the FolderItem is Nil</li> <li>• A system I/O error occurs</li> </ul>
-------------	--

#### Sample Code

```
Using Xoyo.Core
Using Xoyo.IO
```

```
Dim f As FolderItem
f = SpecialFolder.Documents.Child("SaveData.txt")
```

```
Dim output As TextOutputStream
Try
    output = TextOutputStream.Append(f, TextEncoding.UTF8)
    output.WriteLine("This text is appended to the end of the file.")
    output.Close
Catch e As IOException
    Label1.Text = "Unable to append to file."
End Try
```

### Create(f As FolderItem, encoding As TextEncoding) As TextOutputStream.

Creates a text file for so that text can be written. If the file exists, it will be erased and recreated.

#### Exceptions

IOException	For these conditions: <ul style="list-style-type: none"> <li>• containg folder is not writeable</li> <li>• The FolderItem is Nil</li> <li>• The FolderItem exists and could not be deleted</li> <li>• a system I/O error occurs</li> </ul>
-------------	--

## Sample Code

Create a text file and write data to it:

```
Using Xojo.Core
Using Xojo.IO

Dim f As FolderItem
f = SpecialFolder.Documents.Child("SaveData.txt")

Dim output As TextOutputStream
Try
    output = TextOutputStream.Create(f, TextEncoding.UTF8)
    output.WriteLine("Hello, World!")
    output.Close
Catch e As IOException
    Label1.Text = "Unable to create or write to file."
End Try
```

# Xojo.Math

The Math namespace contains math-related constants and functions.

## Namespace Summary

Name	Xojo.Math
Type	Namespace
Methods	<a href="#">Abs</a> <a href="#">ACos</a> <a href="#">ASin</a> <a href="#">ATan</a> <a href="#">ATan2</a> <a href="#">Ceil</a> <a href="#">Cos</a> <a href="#">Exp</a> <a href="#">Floor</a> <a href="#">Log</a> <a href="#">Max</a> <a href="#">Min</a> <a href="#">RandomInt</a> <a href="#">Round</a> <a href="#">Sign</a> <a href="#">Sin</a> <a href="#">Sqrt</a> <a href="#">Tan</a>
Project Types	All
Platforms	All
Related	<a href="#">Double</a> <a href="#">Integer</a> <a href="#">Using</a>

## Methods

### Abs(value As Double) As Double

Returns the [absolute value](#) of a number.

#### Parameters

<i>value</i>	The number to convert to an absolute value.
--------------	---

#### Return Value

Returns the absolute value of the specified *value*.

## Sample Code

Get the absolute values:

Using `Xojo.Math`

```
Dim d As Double
d = Abs(23.9) // returns 23.9
d = Abs(-23.9) // returns 23.9
```

---

## ACos(value As Double) As Double

The arccosine of the specified *value*. The arccosine is the angle whose cosine is *value*.

### Parameters

<i>value</i>	The number for which you want the arccosine.
--------------	--

### Return Value

Returns the arccosine.

---

## ASin(value As Double) As Double

The arcsine of the specified *value*.

### Parameters

<i>value</i>	The number for which you want the arcsine.
--------------	--

### Return Value

Returns the arcsine.

### Example

---

## ATan(value As Double) As Double

The arctangent of the specified *value*.

### Parameters

<i>value</i>	The number to convert to an arctangent.
--------------	---

### Return Value

Returns the arctangent.

---

## ATan2(y As Double, x As Double) As Double

The arctangent of the point whose coordinates are  $x$  and  $y$ . The arctangent is an angle from the  $x$ -axis to a line drawn through the origin  $(0, 0)$  and a point with coordinates  $x, y$ .

### Parameters

$y$	The $y$ coordinate of the point.
$x$	The $x$ coordinate of the point.

### Return Value

Returns the arctangent.

---

## Ceil(value As Double) As Double

Gets the ceiling of a number by rounding it up to the nearest whole number.

### Parameters

<i>value</i>	The number to round up to the nearest whole number.
--------------	---

### Return Value

Returns the nearest whole number rounded up from *value*.

### Sample Code

Get the ceiling of a number:

```
Using Xojo.Math
```

```
Dim d As Double  
d = Ceil(1.234) // returns 2
```

Ceil only works to whole numbers so if you want to get round using a specific number of decimal places, you first have to multiply the number by  $10^{(\text{number of decimal places})}$ , calculate the ceil and then divide it by the same value to get the number. For example, this example rounds 1.2345 to 1.3:

```
Using Xojo.Math
```

```
Dim d As Double  
Const kDecimalPlaces = 1  
  
d = Ceil(1.2345 * (10 ^ kDecimalPlaces)) / (10 ^ kDecimalPlaces) // d = 1.3
```

---

## Cos(value As Double) As Double

The cosine of the specified value.

### Parameters

<i>value</i>	The number to convert to to a cosine.
--------------	---------------------------------------

### Return Value

Returns the cosine.

### Sample Code

Gets the cosine of a 45 degree angle:

Using `Xojo.Math`

```
Dim angle As Double = 45 * (Pi/180) // Convert 45 degrees to radians
Dim value As Double
value = Cos(angle) // value = 0.707
```

---

## Exp(value As Double) As Double

Calculates the specified exponent of "e".

### Parameters

<i>value</i>	The number to use as the exponent for "e".
--------------	--

### Return Value

Returns the specified exponent of "e".

---

## Floor(value As Double) As Double

Rounds the *value* down to the nearest whole number.

### Parameters

<i>value</i>	The number to round down to the nearest whole number.
--------------	---

### Return Value

Returns the nearest whole number rounded down from *value*.

## Sample Code

The floor of a decimal number:

Using `Xojo.Math`

```
Dim d As Double
d = Floor(1.234) // d = 1
```

---

## Log(value As Double) As Double

The natural logarithm of the specified value.

### Parameters

<i>value</i>	The number to convert to a natural logarithm.
--------------	---

### Return Value

Returns the natural logarithm.

## Sample Code

Log 10:

Using `Xojo.Math`

```
Dim d As Double
d = Log(10) // d = 2.3025851
```

---

## Max(value1 As Double, value2 As Double, ParamArray moreValues As Double) As Double

Determines the maximum value out of all the values passed in as parameters.

### Parameters

<i>value</i>	Two or more values are required.
--------------	----------------------------------

### Return Value

Returns the maximum of all the values.

## Sample Code

Calculates the maximum of multiple values:

Using `Xojo.Math`

---

```
Dim d As Double
d = Max(3.01, 4.05) // d = 4.05

d = Max(3.012, 3.011, 1.56) // d = 3.012

Dim i As Integer
i = Max(4, 8, 1, 9, 13, 11, 4) // i = 13
```

---

## Min(value1 As Double, value2 As Double, ParamArray moreValues As Double) As Double

Determines the minimum value out of all the values passed in as parameters.

### Parameters

<i>value</i>	Two or more values are required.
--------------	----------------------------------

### Return Value

Returns the minimum of all the values.

---

## RandomInt(min As Int64, max As Int64) As Int64

Generates a random integer in the given range (inclusive).

### Parameters

<i>min</i>	The minimum value for the range.
<i>max</i>	The maximum value for the range.

### Exceptions

<a href="#">InvalidArgumentException</a>	If <i>min</i> is greater than <i>max</i> .
--	--

### Sample Code

```
Using Xojo.Math
Dim num As Integer = RandomInt(10, 50)
```

---

## Round(value As Double) As Double

Rounds a value to the nearest whole number.

### Parameters

<i>value</i>	The number to round to the nearest whole number.
--------------	--

## Return Value

Returns the nearest whole number to the value.

## Sample Code

Some rounding examples:

Using `Xojo.Math`

```
Dim d As Double
d = Round(1.499) // d = 1
d = Round(1.500) // d = 2
d = Round(1.1) // d = 1
d = Round(1.7) // d = 2
```

---

## Sign(value As Double) As Double

The sign of the number.

### Parameters

<i>value</i>	The number for which to get the sign.
--------------	---------------------------------------

## Return Value

Returns the sign of the number, -1 if the number is negative, 0 if it is zero and 1 if the number is positive.

## Sample Code

Using `Xojo.Math`

```
Dim numSign As Integer
numSign = Sign(-55) // numSign = -1
numSign = Sign(42) // numSign = 1
numSign = Sign(0) // numSign = 0
```

---

## Sin(value As Double) As Double

The sine of the specified value.

### Parameters

<i>value</i>	The angle (in radians) for which you want the sine.
--------------	---

## Return Value

Returns the sine.

## Sample Code

Calculate the sine:

```
Using Xojo.Math
```

```
Dim d As Double  
d = Sin(0.5)
```

---

## Sqrt(value As Double) As Double

The square root of the specified value.

### Parameters

<i>value</i>	The value for which you want to calculate the square root.
--------------	--

### Return Value

Returns the square root.

## Sample Code

Calculate the sine:

```
Using Xojo.Math
```

```
Dim d As Double  
d = Sqrt(16)
```

---

## Tan(value As Double) As Double

The tangent of the specified value.

### Parameters

<i>value</i>	The value (in radians) for which you want the tangent.
--------------	--

### Return Value

Returns the tangent.

## Sample Code

Calculate the tangent:

```
Using Xojo.Math
```

```
Dim d As Double
```

---

`d = Tan(45 * Pi/180) // d = 1.0` which is the tangent of a 45 degree angle

# Net

Contains network related classes that work across all target platforms.

## Namespace Summary

Name	Xojo.Net
Classes	<a href="#">HTTPSocket</a> <a href="#">NetException</a> <a href="#">SSLSettings</a> <a href="#">TCPSocket</a>
Project Types	All
Platforms	All
Related	<a href="#">Using</a>

## Notes

Not all classes in this namespace are available for all project types. Check the individual classes for specifics.

# HTTPSocket

Used to send and receive data via the HTTP 1.1 protocol.

## Class Summary

Name	Xojo.Net.HTTPSocket
Type	Class
Constants	<a href="#">SizeUnknown</a>
Events	<a href="#">AuthenticationRequired</a> <a href="#">Error</a> <a href="#">FileReceived</a> <a href="#">HeadersReceived</a> <a href="#">PageReceived</a> <a href="#">ReceiveProgress</a> <a href="#">SendProgress</a>
Properties	<a href="#">ValidateCertificates</a>
Methods	<a href="#">ClearRequestHeaders</a> <a href="#">Disconnect</a> <a href="#">RequestHeader</a> <a href="#">ResponseHeader</a> <a href="#">Send</a> <a href="#">SetRequestContent</a>
Project Types	All
Platforms	All
Related	<a href="#">MemoryBlock</a> <a href="#">TCPSocket</a> <a href="#">TextEncoding</a>
Example Projects	iOS/Networking/HTTPSocket Xojo Framework/HTTPSocketGetExample Xojo Framwork/HTTPSocketPostExample
Videos	<a href="#">Web Services</a>

## Notes

 HTTPSocket is not yet compatible with Web Apps.

 Usage on Linux requires libsoup 2.4.

Use an HTTPSocket when you need to upload or download information on the web. You can download files, communicate with REST web services and other APIs and do any type of HTTP 1.1 communication.

## Constants

SizeUnknown	-1
-------------	----

## Events

### AuthenticationRequired(realm As Text, ByRef name As Text, ByRef password As Text) As Boolean

Called when the connection requires HTTP basic authentication. Set the *name* and *password* and return True.

#### Parameters

<i>realm</i>	The realm is an area (not a particular page, it could be a group of pages) for which the credentials are used.
<i>name</i>	The name to use for authentication. Since it is ByRef, set the parameter to the name.
<i>password</i>	The password to use for authentication. Since it is ByRef, set the parameter to the name.

#### Sample Code

Specify a name and password:

```
name = "MyUserName"
password = "MyPassword"
Return True
```

### Error(err As RuntimeException)

Called when an HTTP error occurs.

#### Parameters

<i>err</i>	The RuntimeException for the error. You can use its Reason property for details.
------------	--

#### Sample Code

Display information about the error:

```
ErrorLabel.Text = err.Reason
```

### FileReceived(URL As Text, HTTPStatus As Integer, file As FolderItem)

Called when a download of a file is completed as a result of calling Send.

## Parameters

<i>URL</i>	The URL that sent the response.
<i>HTTPStatus</i>	The HTTP <a href="#">status code</a> .
<i>file</i>	The FolderItem where the file was saved.

**HeadersReceived(URL As Text, HTTPStatus As Integer)**

Called when headers are received from the server.

**PageReceived(URL As Text, HTTPStatus As Integer, content As [MemoryBlock](#))**

Called when a new page has been retrieved from the server as a result of calling Send.

## Parameters

<i>URL</i>	The URL that sent the response.
<i>HTTPStatus</i>	The HTTP <a href="#">status code</a> .
<i>content</i>	A MemoryBlock containing the response. This may contain binary or text data.

## Sample Code

Convert incoming JSON content from a MemoryBlock into a Dictionary:

```
Dim jsonData As Text
' Convert the content returned from an API from a MemoryBlock to Text.
jsonData = Xojo.Core.TextEncoding.UTF8.ConvertDataToText(content)

' Parse the JSON result into a Dictionary
Dim jsonDict As Xojo.Core.Dictionary
jsonDict = Xojo.Data.ParseJSON(jsonData)
```

**ReceiveProgress(bytesReceived As Int64, totalBytes As Int64, newData As [MemoryBlock](#))**

Call periodically as data is received.

## Parameters

<i>bytesReceived</i>	The number of bytes received in this chunk (not the total number of bytes received so far).
<i>totalBytes</i>	The total bytes available to receive.
<i>newData</i>	The new data that was received.

---

## SendProgress(bytesSent As Int64, bytesLeft As Int64)

Called periodically as data is sent/uploaded.

### Parameters

<i>bytesSent</i>	The bytes that have been sent.
<i>bytesLeft</i>	The bytes left to send.

## Properties

### ValidateCertificates As Boolean

When set to True, the socket verifies the supplied certificate for authenticity. The Error event is called if the validation fails.

## Methods

### ClearRequestHeaders

Clears all of the request headers, which is useful if you are using the socket to send a different request.

### Sample Code

```
mySocket.ClearRequestHeaders
```

---

## Disconnect

Disconnects the socket.

---

## RequestHeader(name As Text) As Text

### RequestHeader(name As Text, Assigns value As Text)

Gets or sets the request header with the key of *name*. The Name parameter's encoding is converted to ASCII. If the header does not exist in the request headers, an empty Text is returned. Raises a RuntimeException if the encoding could not be converted to ASCII.

### Parameters

<i>name</i>	The name of the key to get or set.
<i>value</i>	The value for the key.

---

## Sample Code

Sets an API key as a token in the request header:

```
Self.RequestHeader("Authorization") = "MyAPIKey"  
mySocket.Send("GET", "http://www.webservice.com/GetData")
```

---

## ResponseHeader(name As Text) As Text

Gets a response header.

---

## Send(method As Text, URL As Text)

Asynchronously sends a request using an HTTP *method* such as GET or POST.

### Sample Code

```
mySocket.Send("GET", "http://127.0.0.1:8080/GetData")
```

---

## Send(method As Text, URL As Text, file As FolderItem)

Asynchronously sends a request using an HTTP method such as GET or POST and returns the output in *file*.

### Sample Code

Sends a GET request, saving the response to a file:

```
Dim outputFile As Xojo.IO.FolderItem =  
Xojo.IO.SpecialFolder.Documents.Child("data.json")  
mySocket.Send("GET", "http://127.0.0.1:8080/GetData", outputFile)
```

---

## SetRequestContent(data As MemoryBlock, mimeType As Text)

Sets the content data and content type to be sent to the server.

### Sample Code

Sends a POST request, converting JSON data to a MemoryBlock before sending:

```
Using Xojo.Core  
Using Xojo.Data
```

```
' Simple data in a Dictionary  
Dim info As New Dictionary  
info.Value("ID") = 123456
```

```
' Convert to JSON text
Dim json As Text
json = GenerateJSON(info)

' Convert to MemoryBlock
Dim data As MemoryBlock
data = TextEncoding.UTF8.ConvertTextToData(json)

mySocket.SetRequestContent(data, "application/x-www-form-urlencoded")
mySocket.Send("POST", "http://127.0.0.1:8080/GetCustomer")</code>
```

To send form information, you build the text yourself:

```
Using Xojo.Core
Using Xojo.Data
```

```
' Build form text
Dim formText As Text = "firstName=Bob&lastName=Roberts"

' Convert to MemoryBlock
Dim postData As Xojo.Core.MemoryBlock
postData = Xojo.Core.TextEncoding.UTF8.ConvertTextToData(formText)

' POST it
MyHttpSocket.SetRequestContent(postData, "application/x-www-form-urlencoded")
MyHttpSocket.Send("POST", "http://www.webserviceurl.com")
```

# NetException

Indicates a network error.

## Item Summary

Name	Xojo.Net.NetException
Type	Class
Super	<a href="#">ErrorException</a>
Constants	<a href="#">InvalidCertificate</a> <a href="#">OperationTimedOut</a>
Project Types	Mobile
Platforms	iOS
Related	

## Constants

InvalidCertificate	10	
OperationTimedOut	9	

# SSLSettings

Description of the item.

## Item Summary

Name	Xojo.Net.SSLSettings
Type	Class
Enumerations	<a href="#">SecurityLevels</a>
Properties	<a href="#">SecurityLevel</a> <a href="#">ValidateCertificates</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">TCPSocket</a>

## Notes

Although this class is available for all project types, it can only be used with [TCPSocket](#), which is currently only available for iOS.

## Enumerations

### SecurityLevels

Security levels that can be used with sockets.

None	No security settings.
SSLv2	Secure Socket Layer 2
SSLv3	Secure Socket Layer 3
TLSv1	Transport Layer Security 1
Negotiated	Most of the time you should use this setting, which will use the best security level available starting with TLSv1.2.

## Properties

### SecurityLevel As SecurityLevels

Specifies the security level. This defaults to `SecurityLevels.Negotiated`.

## ValidateCertificates As Boolean

Indicates that the related certificates should be validated using the specified security level.

# TCP Socket

This class is used for TCP/IP communication.

## Item Summary

Name	Xojo.Net.TCP Socket
Type	Class
Enumerations	<a href="#">HandleTypes</a>
Events	<a href="#">ConfigureSocket</a> <a href="#">Connected</a> <a href="#">DataAvailable</a> <a href="#">Disconnected</a> <a href="#">Error</a>
Properties	<a href="#">Address</a> <a href="#">BytesAvailable</a> <a href="#">IsConnected</a> <a href="#">Port</a> <a href="#">SSLSettings</a>
Methods	<a href="#">AvailableData</a> <a href="#">Connect</a> <a href="#">Disconnect</a> <a href="#">Handle</a> <a href="#">ReadData</a> <a href="#">WriteData</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">SSLSettings</a>
Example Projects	Examples/iOS/Networking/EasyTCP Socket Examples/iOS/Networking/TCP

## Enumerations

### HandleTypes

Available handle types.

FileDescriptor	
CFReadStream	
CFWriteStream	

---

## Events

### ConfigureSocket

Called when the socket needs to be configured.

---

### Connected

Called when the socket connects to another socket or server.

---

### DataAvailable

Called when additional data has come into the internal receive buffer.

#### Notes

It is your responsibility to read the data from the buffer using the `ReadData` method.

---

### Disconnected

Called when the socket disconnects after previously being connected.

---

### Error(err As RuntimeException)

Called when an error occurs, with *err* containing the exception causing the error.

---

## Properties

### Address As Text

The TCP/IP address to connect with.

---

### BytesAvailable As UInteger

The number of bytes of data that are available in the internal receive buffer. You can get the data by calling the `ReadData` method.

---

### IsConnected As Boolean

Indicates whether the socket is currently connected.

---

---

## Port As Int32

The port to bind on or connect to.

### Notes

On most operating systems, attempting to bind on a port less than 1024 (without administrator privileges) causes an Error event.

You need to set the port property explicitly before any call to Connect as the Port property is changed to reflect the actual port that the OS has bound to.

---

## SSLSettings As SSLSettings

Specifies the SSL settings for the socket.

## Methods

### AvailableData As MemoryBlock

Returns the data available in the receive buffer.

---

### Connect

Attempts to connect to the specified address and port.

---

### Disconnect

Disconnects the socket, resets it, and calls the Disconnected event handler.

---

### Handle(type As HandleTypes) As Ptr

Gets the specified OS handle for the socket.

---

### ReadData(length As Integer) As MemoryBlock

Reads *length* bytes from the socket.

### Notes

This reads the bytes that are in the buffer. Not all data that was sent will necessarily appear in the buffer right away.

### Sample Code

```
// Read all data that is available in the buffer
```

---

```
Dim data As MemoryBlock  
data = myTCPSocket.ReadData(myTCPSocket.BytesAvailable)
```

---

## WriteData(data As MemoryBlock)

Writes *data* to the socket.

# Xojo.System

This namespace contains system-specific functions and classes.

## Namespace Summary

Name	Xojo.System
Methods	<a href="#">DebugLog</a> <a href="#">Microseconds</a> <a href="#">Ticks</a>
Project Types	iOS
Platforms	iOS
Related	
Example Projects	Examples/iOS/Framework/TicksExample

## Methods

### DebugLog(msg As Text)

Outputs information to the system log. This information is displayed in the Messages pane of the Xojo IDE. When running on the iOS Simulator, it is also captured in the Simulator console log, which you can access from its Debug->System Log menu item.

---

### Microseconds As Double

The number of Microseconds (one billionth of a second / 1,000,000 of a second) that have passed since the computer/device was started.

#### Sample Code

Calculate the number of seconds that have elapsed:

```
Dim start As Double = Xojo.System.Microseconds

// Do a long-running process here

Dim finished As Double = Xojo.System.Microseconds
Dim elapsedSeconds As Double = (finished - start) / 1000000
```

## Ticks As Double

The number of ticks (60ths of a second) that have passed since the computer/device was started.

### Sample Code

Calculate the number of seconds that have elapsed:

```
Dim start As Double = Xojo.System.Ticks
```

```
// Do a long-running process here
```

```
Dim finished As Double = Xojo.System.Ticks
```

```
Dim elapsedSeconds As Double = (finished - start) / 60
```

# Threading

Contains classes related to threading.

## Namespace Summary

Name	Xojo.Threading
Classes	<a href="#">CriticalSection</a> <a href="#">IllegalLockingException</a> <a href="#">Semaphore</a> <a href="#">Thread</a>
Project Types	iOS
Platforms	iOS
Related	

# CriticalSection

Used to protect a single resource in a multithreaded environment.

## Class Summary

Name	Xojo.Threading.CriticalSection
Type	Class
Methods	<a href="#">Enter</a> <a href="#">Leave</a> <a href="#">TryEnter</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">Semaphore</a>

## Constructors

## Properties

## Methods

### Enter

Attempts to get a lock on the resource managed by the CriticalSection.

---

### Leave

Call Leave when you are finished using the protected resource and you want to give it back to the CriticalSection.

---

### TryEnter As Boolean

Attempts to get a lock on the resource managed by the CriticalSection. TryEnter is similar to Enter but returns True if it succeeds and the thread has exclusive use of the resource. If it fails, it returns False but does not block execution of the thread.

# IllegalLockingException

Raised when Semaphores and CriticalSections locks are illegal.

## Item Summary

Name	Xojo.Threading.IllegalLockingException
Type	Class
Super	<a href="#">LogicException</a>
Properties	<a href="#">ErrorNumber</a> <a href="#">Message</a> <a href="#">Reason</a>
Methods	<a href="#">CallStack</a> <a href="#">Stack</a>
Project Types	iOS
Platforms	iOS
Related	

# Semaphore

Used to coordinate access to a shared resource.

## Class Summary

Name	Xojo.Threading.Semaphore
Type	Class
Methods	<a href="#">Release</a> <a href="#">Signal</a> <a href="#">TrySignal</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">CriticalSection</a>

## Constructors

### Constructor(Optional numResources As Integer = 1)

Creates a semaphore with an initial count of resources to track.

## Methods

### Release

Call Release to give a locked resource back to the Semaphore.

### Signal

Call Signal to obtain a lock on a resource, blocking until it does. Once you are done with the shared resource, call Release to decrement the internal counter.

### TrySignal As Boolean

Determines whether there is a lock available. If there is a lock available, you are granted the lock and TrySignal returns True. When you are finished with the resource, call Release to give it back to the Semaphore. If the lock is not available, it does not block. Instead, TrySignal returns False to let you know. Do not attempt to use the resource after it returns False because you are likely to collide with another thread.

# Thread

Used to run code in the background. Xojo threads are co-operative (not pre-emptive).

## Class Summary

Name	Xojo.Threading.Thread
Type	Class
Constants	<a href="#">PriorityHigh</a> <a href="#">PriorityLow</a> <a href="#">PriorityNormal</a>
Enumerations	<a href="#">ThreadStates</a>
Events	<a href="#">Run</a>
Properties	<a href="#">Priority</a> <a href="#">StackSize</a> <a href="#">State</a>
Methods	<a href="#">Resume</a> <a href="#">Run</a> <a href="#">Sleep</a> <a href="#">Suspend</a>
Shared Methods	<a href="#">CurrentThread</a>
Project Types	iOS
Platforms	iOS
Related	
Example Projects	<a href="#">Examples/iOS/Framework/AdvancedThreading</a> <a href="#">Examples/iOS/Framework/ThreadExample</a>

## Constants

These constants can be used to set the Priority of the thread:

PriorityHigh	10
PriorityLow	1
PriorityNormal	5

## Enumerations

## ThreadStates

These are the various states of a thread.

Running	The thread is currently running.
Waiting	The thread is waiting to run.
Suspended	The thread is suspended.
Sleeping	The thread is sleeping.
NotRunning	The thread is not running.

## Events

### Run

Called when the thread is started by calling the [Run](#) method. The code you want to run in the thread should be in this event handler or should be called from it.

## Properties

### Priority As Integer

Indicates the priority of the thread.

#### Notes

The main thread for the app has a priority of 5. You can alter the priority of your own threads to give them more or less time relative to the main thread. The default is also 5.

The higher the priority value, the more time the thread is allocated, in relation to the main thread. For example, if you set the Priority = 10, then your thread will run twice as often as the main thread (since 10 is 5\*2). A Priority value that is too high might prevent other threads from running at all.

#### Sample Code

```
Thread1.Priority = Thread.PriorityLow
```

---

### StackSize As UInteger

The size of the thread stack, in bytes.

---

### State As [ThreadStates](#) (read-only)

Indicates the state of the thread using the [ThreadStates](#) enumeration.

## Sample Code

```
Dim status As Text
Select Case state
Case Thread.Running
    status = "Running"
Case Thread.Waiting
    status = "Waiting"
Case Thread.Sleeping
    status = "Waiting"
Case Thread.Suspended
    status = "Suspended"
Case Thread.NotRunning
    status = "Not running"
End Select
```

## Methods

### Resume

Wakes a thread that is sleeping or suspended so that it may continue running.

## Sample Code

```
LongRunningThread.Resume
```

---

### Run

Starts the thread and calls its [Run](#) event handler.

---

### Sleep(milliSeconds As Integer, wakeEarly As Boolean = False)

Puts the thread to sleep for the specified amount of *milliSeconds*, optionally allowing it to be woken early.

#### Notes

If *wakeEarly* is True, a thread may be woken before *milliSeconds* is reached if there are no other threads to run.

## Sample Code

```
LongRunningThread.Sleep(300)

LongRunningThread.Sleep(1000, True)
```

---

## Suspend

Puts the thread in a suspended state where it will no longer run. You can allow the thread to continue by calling [Resume](#).

### Notes

When a thread is suspended, it is not allocated any processing time, regardless of its priority.

### Sample Code

```
LongRunningThread.Suspend  
  
// Other code goes here  
  
LongRunningThread.Resume</code>
```

## Shared Methods

### CurrentThread As Thread

The currently running thread.

### Sample Code

```
Thread.CurrentThread.Sleep(500)
```

# iOS Framework

These are the controls, classes and interfaces used by iOS apps.

## Framework Summary

Controls	<a href="#">UIButton</a> <a href="#">UICanvas</a> <a href="#">UIControl</a> <a href="#">UIDatePicker</a> <a href="#">UIHTMLViewer</a> <a href="#">UIImageView</a> <a href="#">UILabel</a> <a href="#">UILayoutConstraint</a> <a href="#">UILine</a> <a href="#">UIOval</a> <a href="#">UIProgressbar</a> <a href="#">UIProgressWheel</a> <a href="#">UIRectangle</a> <a href="#">UISegmentedControl</a> <a href="#">UISegmentedControlItem</a> <a href="#">UISeparator</a> <a href="#">UISlider</a> <a href="#">UISwitch</a> <a href="#">UITable</a> <a href="#">UITextArea</a> <a href="#">UITextField</a> <a href="#">UIToolbar</a> <a href="#">UIToolButton</a> <a href="#">UIUserControl</a>
Classes	<a href="#">UIApplication</a> <a href="#">UIImage</a> <a href="#">UIBlock</a> <a href="#">UIEventInfo</a> <a href="#">UIFont</a> <a href="#">UIGraphics</a> <a href="#">UIImage</a> <a href="#">UIMessageBox</a> <a href="#">UIPath</a> <a href="#">UISound</a> <a href="#">UITableCellData</a> <a href="#">SQLiteDatabase</a> <a href="#">SQLiteDatabaseField</a> <a href="#">SQLiteException</a> <a href="#">SQLiteRecordSet</a>
Interfaces	<a href="#">iOSScreenContent</a> <a href="#">iOSSplitContent</a> <a href="#">iOSTableDataSource</a> <a href="#">iOSViewController</a>

---

Layout	<a href="#">iOSScreen</a> <a href="#">iOSSplitView</a> <a href="#">iOSTabBar</a> <a href="#">iOSView</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">Xojo namespace</a>

# UIApplication

The base class for iOS apps.

## Class Summary

Name	UIApplication
Type	Class
Events	<a href="#">LowMemoryWarning</a> <a href="#">Open</a> <a href="#">UnhandledException</a>
Properties	<a href="#">DefaultiPhoneScreen</a> <a href="#">DefaultiPadScreen</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">UIScreen</a> <a href="#">iOS Apps</a>

## Events

### LowMemoryWarning

Called when iOS reports a low-memory situation. Applications should attempt to free up objects and other resources when this event is raised. If the low-memory situation persists, iOS may kill your app.

### Open

Called when the iOS app is first launched.

### UnhandledException(exc As [RuntimeException](#)) As [Boolean](#)

Called if the app had an exception and did not handle it. If the event is not implemented or the event handler returns `False`, the application will terminate.

#### Parameters

exc	The unhandled <a href="#">RuntimeException</a> .
-----	--

## Properties

### DefaultiPhoneScreen (design property)

Set to the default [iOSScreen](#) to use when the app is launched on an iPhone-sized device. If you do not specify a `DefaultiPhoneScreen`, then the app will not run on iPhone-sized devices.

#### Notes

One or both of `DefaultiPhoneScreen` and `DefaultiPadScreen` must be specified.

By specifying separate screens for `DefaultiPhoneScreen` and `DefaultiPadScreen`, you can have completely different layouts for each type of device.

---

### DefaultiPadScreen (design property)

Set to the default [iOSScreen](#) to use when the app is launched on an iPad-sized device. If no `DefaultiPadScreen` is specified, the app will still run on an iPad but will use the `DefaultiPhoneScreen`, running it in "scaled" mode.

#### Notes

One or both of `DefaultiPhoneScreen` and `DefaultiPadScreen` must be specified.

By specifying separate screens for `DefaultiPhoneScreen` and `DefaultiPadScreen`, you can have completely different layouts for each type of device.

# iOSBitmap

A bitmap image that can be modified using the Graphics property.

## Class Summary

Name	iOSBitmap
Type	Class
Properties	<a href="#">Graphics</a> <a href="#">Height</a> <a href="#">Image</a> <a href="#">Scale</a> <a href="#">Width</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSGraphics</a>

## Constructors

Constructor(width As [Integer](#), height As [Integer](#), scale As [Double](#), opaque As [Boolean](#) = False)

Creates a bitmap image using the specified *width*, *height*, *scale* and *opaqueness*.

### Parameters

<i>width</i>	The width of the bitmap.
<i>height</i>	The height of the bitmap.
<i>scale</i>	The scale ratio of the bitmap.
<i>opaque</i>	Indicates if the bitmap is opaque.

### Notes

Setting *opaque* to True disables the alpha channel. If you know that you will be filling every point in the bitmap, set *opaque* to True to improve performance.

The *scale* indicates how the image appears on the device. For example, bitmaps drawn with a scale of 1.0 will appear at half the size when drawn on a device with a 2x screen. When creating an `iOSBitmap` you intend to put on screen, you want your bitmap to have the same scale factor as the screen and the number of pixels in each dimension multiplied by the scale factor. Use the example code on the [Declare](#) page to check the device scale.

## Sample Code

```
Dim pic As New iOSBitmap(100, 100, 2.0, False)
pic.Graphics.DrawRect(10, 10, 60, 60)
```

## Properties

### Graphics As [iOSGraphics](#) (read-only)

The [iOSGraphics](#) object for the bitmap that is used for drawing.

## Sample Code

```
Dim pic As New iOSBitmap(100, 100, 2.0, False)
pic.Graphics.DrawRect(10, 10, 60, 60)
```

---

### Height As [UInteger](#) (read-only)

The height of the bitmap that was created.

---

### Image As [UIImage](#) (read-only)

The image that has been drawn to the bitmap. If you need to modify the image, use the [Graphics](#) property.

---

### Scale As [Double](#) (read-only)

The scale of the image that has been drawn to the bitmap.

---

### Width As [UInteger](#) (read-only)

The width of the image that has been drawn to the bitmap.

# iOSBlock

Used to help call declares taking Obj-C blocks. An iOSBlock is created from a Xojo delegate and in turn creates an Objective-C block that will call into the delegate. This Objective-C block is retrieved from the Handle property and passed into declares.

## Control Summary

Name	iOSBlock
Type	Class
Properties	<a href="#">Handle</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">Declare</a> <a href="#">Ptr</a>

## Notes

iOSBlock is only suitable for blocks that execute on non-background queues/threads.

## Constructors

### Constructor(theDelegate As Object)

Used to supply the delegate to create for the block.

## Properties

### Handle As [Ptr](#) (read-only)

Gets the Objective-C block.

## Sample Code

This example iterates over all of the subviews of a view using a block:

This code goes in a button's Action event:

```
Declare Function view Lib "UIKit" Selector "view" ( obj as Ptr ) As Integer
Declare Function subviews Lib "UIKit" Selector "subviews" ( obj as Integer ) As
Integer
```

```
Declare Sub enumerateObjectsUsingBlock Lib "Foundation" Selector  
"enumerateObjectsUsingBlock:" ( obj As Integer, block As Integer )
```

```
Dim block As New iOSBlock( AddressOf Enumerator )  
enumerateObjectsUsingBlock( subviews( view( self.Handle ) ), block.Handle )
```

And add a new function to the view:

```
Sub Enumerator(viewObj As Integer)  
Declare Function description Lib "Foundation" Selector "description" ( obj As Integer  
) As CFStringRef  
  
Dim viewDescription As Text = description(viewObj)  
DebugLog viewDescription  
End Sub
```

# iOSButton

This is the standard Button control for iOS.

## Control Summary

Name	<a href="#">iOSButton</a>
Type	Control
Super	<a href="#">iOSControl</a>
Events	<a href="#">Action</a> <a href="#">Close</a> <a href="#">Open</a>
Properties	<a href="#">AccessibilityHint</a> <a href="#">AccessibilityLabel</a> <a href="#">Caption</a> <a href="#">Enabled</a> <a href="#">Height</a> <a href="#">Left</a> <a href="#">Name</a> <a href="#">Parent</a> <a href="#">TextColor</a> <a href="#">TextFont</a> <a href="#">Top</a> <a href="#">Visible</a> <a href="#">Width</a>
Methods	<a href="#">Handle</a> <a href="#">Invalidate</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSFont</a> <a href="#">iOS Buttons</a>
UIKit	<a href="#">UIButton</a>

## Events

### Action

The Action event handler is called when the button is pressed, tapped, clicked, etc.

### Sample Code

Change Label Text when a Button is pressed:

```
Label1.Text = "You pressed the button."
```

## Properties

### Caption As Text

Gets or sets the caption for the Button.

Sample Code

```
Me.Caption = "Save"
```

---

### Enabled As Boolean

Indicates whether the control is enabled or disabled.

Sample Code

```
Button1.Enabled = False
```

---

### TextColor As Color

The color to display the text.

Sample Code

Change the text color to red:

```
Button1.TextColor = Color.Red
```

---

### TextFont As iOSFont

The font use for displaying the Caption.

Sample Code

```
Button1.TextFont = iOSFont.ItalicSystemFont  
Label1.TextFont = iOSFont.SystemFont(32)
```

# iOSCanvas

Used to draw graphics.

## Control Summary

Name	iOSCanvas
Type	Control
Super	<a href="#">iOSControl</a>
Events	<a href="#">Close</a> <a href="#">Open</a> <a href="#">Paint</a> <a href="#">PointerDown</a> <a href="#">PointerDrag</a> <a href="#">PointerUp</a>
Properties	<a href="#">AccessibilityHint</a> <a href="#">AccessibilityLabel</a> <a href="#">Height</a> <a href="#">Left</a> <a href="#">Name</a> <a href="#">Parent</a> <a href="#">Top</a> <a href="#">Visible</a> <a href="#">Width</a>
Methods	<a href="#">Handle</a> <a href="#">Invalidate</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSEventInfo</a> <a href="#">iOSGraphics</a> <a href="#">iOS Decorations</a>
Example Projects	<a href="#">Examples/iOS/Controls/SwipeExample</a> <a href="#">Examples/iOS/Controls/TouchCanvas</a>

## Events

### Paint(g As [iOSGraphics](#))

Called when the Canvas needs to update its graphical display.

## Parameters

<i>g</i>	The graphics object to use for all drawing.
----------	---

## Sample Code

To draw a blue rectangle in the Canvas:

```
g.FillColor = Color.Blue
g.FillRect(0, 0, g.Width, g.Height)
```

## PointerDown(pos As Point, eventInfo As iOSEventInfo)

Called when the pointing device is tapped, touched or clicked.

## Parameters

<i>pos</i>	A Point that specifies the location of the touch.
<i>eventInfo</i>	Provides specific information about the touch event.

## Sample Code

This code saves all touch points to an array, which is used to display circles in the Paint event:

```
// mTouches is a property
// mTouches() As Point

If eventInfo.PointerCount > 0 Then
    // Save multi-touch points
    For i As Integer = 0 To eventInfo.PointerCount-1
        mTouches.Append(eventInfo.PointerPosition(i))
    Next
Else
    mTouches.Append(pos)
End If
```

## Me.Invalidate

This code goes in the Paint event handler:

```
For Each p As Point In mTouches
    g.FillColor = Color.Blue
    g.FillOval(p.X, p.Y, 10, 10)
Next
```

## PointerDrag(pos As Point, eventInfo As iOSEventInfo)

Called when the pointing device is dragged.

### Parameters

<i>pos</i>	A Point that specifies the location of the touch.
<i>eventInfo</i>	Provides specific information about the touch event.

### Sample Code

Saves the drag positions to draw as circles:

```
// mTouches is a property
// mTouches() As Point
```

```
mTouches.Append(pos)
```

```
Me.Invalidate
```

## PointerUp(pos As Point, eventInfo As iOSEventInfo)

Called when the pointing device is raised or released.

### Parameters

<i>pos</i>	A Point that specifies the location of the touch.
<i>eventInfo</i>	Provides specific information about the touch event.

### Sample Code

This code saves all touch points in an array where circles are displayed:

```
// mTouches is a property
// mTouches() As Point
```

```
If eventInfo.PointerCount > 0 Then
    // Save multi-touch points
    For i As Integer = 0 To eventInfo.PointerCount-1
        mTouches.Append(eventInfo.PointerPosition(i))
    Next
Else
    mTouches.Append(pos)
End If
```

```
Me.Invalidate
```

# iOSContainerControl

An `iOSContainerControl` is a way to create a group of controls that are treated as a single control. It is a great way to simplify complex view layouts. In addition, with `iOSContainerControl`, you can create reusable controls to include on multiple views or even other `iOSContainerControls`.

## Control Summary

Name	<code>iOSContainerControl</code>
Type	Class
Super	<a href="#">iOSCanvas</a>
Events	<a href="#">Close</a> <a href="#">Open</a>
Properties	
Project Types	iOS
Platforms	iOS
Related	

# iOSControl

Control is the base class for most Xojo UI controls. You cannot instantiate iOSControl directly.

## Class Summary

Name	iOSControl
Type	Class
Events	<a href="#">Close</a> <a href="#">Open</a>
Properties	<a href="#">AccessibilityHint</a> <a href="#">AccessibilityLabel</a> <a href="#">Height</a> <a href="#">Left</a> <a href="#">Name</a> <a href="#">Parent</a> <a href="#">Top</a> <a href="#">Visible</a> <a href="#">Width</a>
Methods	<a href="#">AddControl</a> <a href="#">Control</a> <a href="#">ControlCount</a> <a href="#">Handle</a> <a href="#">Invalidate</a> <a href="#">RemoveControl</a> <a href="#">SetTintColor</a>
Targets	Mobile
Platforms	iOS
Related	

## Events

### Close

Called when the control is closed by calling its Close method.

### Open

Called when the control is first displayed. This is where you typically put initialization code.

### Sample Code

```
// Set label text  
Me.Text = "Hello"
```

## Properties

### AccessibilityHint As Text

The accessibility hint is a longer description that is read aloud when VoiceOver is enabled.

Sample Code

```
Me.AccessibilityHint = "Click to calculate the value and display the next view."
```

---

### AccessibilityLabel As Text

The accessibility label of of a control is a short name that is read aloud when VoiceOver is enabled.

Sample Code

```
Me.AccessibilityLabel = "Calculate the value."
```

---

### Height As Double (read-only)

The height of the control.

---

### Left As Double (read-only)

The left position of the control.

---

### Name As Text (design-time only)

The name of the control. This can only be set in the Inspector. Use the name to refer to the control.

---

### Parent As iOSControl (read-only)

Indicates the control's parent object, if it has one. If there is no parent, this is Nil.

---

### Top As Double (read-only)

The top position of the control.

---

## Visible As Boolean

Indicates whether the control is visible.

### Sample Code

Make a button invisible:

```
Button1.Visible = False</code>
```

---

## Width As Double (read-only)

The width of the control.

## Methods

### AddControl(child As iOSControl)

Adds a control to the control.

---

### Control(index As Integer) As iOSControl

Gets the control at the specified *index*.

---

### ControlCount As Integer

The number of controls in the control.

---

### Handle As Ptr

The handle is used to get a reference to the control for interfacing directly with the iOS API.

---

## Invalidate

Marks the control so that it will be redrawn during the next event loop.

### Sample Code

Call Invalidate to force a Canvas to redraw itself:

```
Canvas1.Invalidate
```

---

## RemoveControl(child As iOSControl)

Removes the control from the control.

---

## SetTintColor(value As Color)

Changes a control's tint color. This varies by control. For example, with an [iOSTextField](#) this changes the cursor color.

# iOSDatePicker

This is the standard Date Picker control for iOS.

## Control Summary

Name	<a href="#">iOSDatePicker</a>
Type	Control
Super	<a href="#">iOSControl</a>
Enumerations	<a href="#">DatePickerMode</a>
Events	<a href="#">Close</a> <a href="#">Open</a> <a href="#">ValueChanged</a>
Properties	<a href="#">AccessibilityHint</a> <a href="#">AccessibilityLabel</a> <a href="#">CountDownDuration</a> <a href="#">DefaultDate</a> <a href="#">Enabled</a> <a href="#">Height</a> <a href="#">Left</a> <a href="#">MaximumDate</a> <a href="#">MinimumDate</a> <a href="#">MinuteInterval</a> <a href="#">Mode</a> <a href="#">Name</a> <a href="#">Parent</a> <a href="#">Top</a> <a href="#">Visible</a> <a href="#">Width</a>
Methods	<a href="#">DefaultDate</a> <a href="#">Handle</a> <a href="#">Invalidate</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">Date</a> <a href="#">iOS Pickers</a>
UIKit	<a href="#">UIDatePicker</a>

## Enumerations

## DatePickerMode

The modes used for display of the date picker.

Time	Displays a timer picker.
Date	Displays a date picker
DateAndTime	Displays a date and time picker.
CountDownTimer	Displays a hour/minute picker.

## Events

### ValueChanged

Called when the selection in the DatePicker is changed.

## Properties

### CountDownDuration As Double

The countdown duration (in seconds) to display when DatePickerMode = CountDownTimer. Contains the selected value.

#### Sample Code

Set a date picker (in CountDownTimer mode) to 5 minutes:

```
CountDownPicker.CountDownDuration = 60 * 5
```

### DefaultDate As Date

The default date to display when DatePickerMode = Date or DateAndTime. Contains the selected date. By default, the current date and/or time is displayed (for date and time display modes).

#### Sample Code

Gets the selected date from the date picker:

```
Dim selectedDate As Xojo.Core.Date
selectedDate = StartDatePicker.DefaultDate
```

### Enabled As Boolean

Indicates if the DatePicker is enabled or disabled.

## Sample Code

Disable a date picker:

```
StartDatePicker.Enabled = False
```

---

## MaximumDate As Date

The maximum date in the scrolling list. Applicable when Mode is Date or DateAndTime.

### Notes

Dates after the maximum date still appear in the scrolling list, but cannot be selected.

## Sample Code

Limit the maximum date to 50 years in the future:

```
Dim futureDate As Xojo.Core.Date  
Dim interval As New Xojo.Core.DateInterval  
interval.Years = 50  
futureDate = Xojo.Core.Date.Now + interval  
StartDatePicker.MaximumDate = futureDate
```

---

## MinimumDate As Date

The minimum date in the scrolling list. Applicable when Mode is Date or DateAndTime.

### Notes

Dates before the minimum date still appear in the scrolling list, but cannot be selected.

## Sample Code

Limit the minimum date to 100 years in the past:

```
Dim pastDate As Xojo.Core.Date  
Dim interval As New Xojo.Core.DateInterval  
interval.Years = -100  
pastDate = Xojo.Core.Date.Now + interval  
StartDatePicker.MinimumDate = pastDate
```

---

## MinuteInterval As Integer

The minute interval. The default is 1. This is only applicable when Mode is CountdownTimer.

### Sample Code

Change the minute interval to 15 minutes:

```
CountdownPicker.MinuteInterval = 15
```

---

## Mode As DatePickerMode

Uses DatePickerMode to specify how the DatePicker should appear. Changing between date/time modes and CountdownTimer resets property values to their defaults.

### Sample Code

Change the date picker to show dates:

```
StartDatePicker.Mode = iOSDatePicker.Modes.Date
```

## Methods

### DefaultDate(animate As Boolean, Assigns value As Date)

Allows you to set a date with animation for scrolling the list.

# UIEventInfo

Contains information about a the UI event.

## Class Summary

Name	UIEventInfo
Type	Class
Properties	<a href="#">Handled</a> <a href="#">PointerCount</a> <a href="#">Timestamp</a>
Methods	<a href="#">Handle</a> <a href="#">PointerPosition</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">UICanvas</a>

## Properties

### Handled As Boolean

TBD

---

### PointerCount As [Integer](#) (Read-Only)

The number of pointers that are part of the event.

#### Returns

The number of pointers.

#### Notes

This is often the number of touch points. Refer to [Canvas.PointerDown](#) for an example of its usage.

---

### Timestamp As [Double](#) (Read-Only)

A time stamp for when the event occurred.

## Methods

### Handle As Ptr

Used for interfacing directly with the native OS API.

---

### PointerPosition(index As Integer) As Point

Gets the position of the specified pointer.

#### Parameters

<i>index</i>	The specific point to get.
--------------	----------------------------

#### Returns

The specified point.

# iOSFont

Provides access to fonts on iOS.

## Class Summary

Name	iOSFont
Type	Class
Properties	<a href="#">Ascent</a> <a href="#">CapHeight</a> <a href="#">Descent</a> <a href="#">Leading</a> <a href="#">LineHeight</a> <a href="#">Name</a> <a href="#">Size</a> <a href="#">XHeight</a> ,
Shared Methods	<a href="#">BoldSystemFont</a> <a href="#">ItalicSystemFont</a> <a href="#">SmallSystemFontSize</a> <a href="#">SystemFont</a> <a href="#">SystemFontSize</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">UILabel</a> <a href="#">UITextView</a> <a href="#">UITextField</a>
Example Projects	Examples/iOS/Controls/FontExample

## Notes

You can find a list of fonts available on iOS devices at [iOSFonts.com](http://iOSFonts.com).

## Constructors

### Constructor(*postscriptName* As Text, *pointSize* As Double)

Creates an `iOSFont` from the *postscriptName*.

#### Notes

Apple has a list of [Fonts Installed with iOS7](#).

iOS adds a few additional fonts:

- KhmerSangamMN

- `DamascusLight`
- `LaoSangamMN`
- `AppleSDGothicNeo-UltraLight`
- `KohinoorDevanagari-Light`
- `KohinoorDevanagari-Medium`
- `KohinoorDevanagari-Book`

## Sample Code

Sets a button text to use Courier:

```
Dim courier As New iOSFont("Courier", 18)
Button1.TextFont = courier
```

## Properties

### Ascent As Double (read-only)

The ascent of the font. The ascent of a font is the distance from the tops of the tallest glyphs to the baseline.

---

### CapHeight As Double (read-only)

The height of a capital letter above the baseline for the font.

---

### Descent As Double (read-only)

The descent of the font. The descent of a font is the distance from the baseline to the bottom of the lowest descenders on the glyphs.

---

### Leading As Double (read-only)

The leading for the font. The leading is the recommended vertical distance from the bottom of the descenders to the top of the next line in a multiline setting.

---

### LineHeight As Double (read-only)

The line height for the font.

---

### Name As Text (read-only)

The name of the font.

---

## Size As Double (read-only)

The size of the font.

---

## XHeight As Double (read-only)

The distance between the baseline and the mean line for the font. Typically, this is the height of the letter x in the font.

## Shared Methods

### **BoldSystemFont**(size As Double = 0) As iOSFont

Gets the bold system font.

#### Parameters

<i>size</i>	The point size of the font, which defaults to size 0 (the standard system font size).
-------------	---

#### Returns

An iOSFont that is typically assigned to a TextFont property.

#### Sample Code

```
Label1.TextFont = iOSFont.BoldSystemFont
```

---

### **ItalicSystemFont**(size As Double = 0) As iOSFont

Gets the italic system font.

#### Parameters

<i>size</i>	The point size of the font, which defaults to size 0 (the standard system font size).
-------------	---

#### Returns

An iOSFont that is typically assigned to a TextFont property.

#### Sample Code

Sets a Label to use a large system italic font:

```
Label1.TextFont = iOSFont.ItalicSystemFont(48)
```

---

## SmallSystemFontSize As Double

Gets the small system font size. Use this to provide the size parameter to `BoldSystemFont`, `ItalicSystemFont` and `SystemFont` to get those fonts in the small size.

### Returns

The small system font size as a `Double`.

### Sample Code

```
Label1.TextFont = iOSFont.SystemFont(iOSFont.SmallSystemFontSize)
```

---

## SystemFont(size As Double = 0) As iOSFont

Gets the system font.

### Parameters

size	The point size of the font, which defaults to size 0 (the standard system font size).
------	---

### Returns

An `iOSFont` that is typically assigned to a `TextFont` property.

### Sample Code

```
Label1.TextFont = iOSFont.SystemFont(48)
```

---

## SystemFontSize As Double

Gets the system font size.

### Returns

The system font size as a `Double`.

### Sample Code

```
Label1.TextFont = iOSFont.ItalicSystemFont(iOSFont.SystemFontSize)
```

# iOSGraphics

Used to draw text, lines, rectangle and images.

## Class Summary

Name	iOSGraphics
Type	Class
Properties	<a href="#">Alpha</a> <a href="#">AntiAlias</a> <a href="#">FillColor</a> <a href="#">Handle</a> <a href="#">Height</a> <a href="#">LineColor</a> <a href="#">LineWidth</a> <a href="#">TextFont</a> <a href="#">TextUnderline</a> <a href="#">Width</a>
Methods	<a href="#">ClipToPath</a> <a href="#">ClipToRect</a> <a href="#">DrawImage</a> <a href="#">DrawLine</a> <a href="#">DrawOval</a> <a href="#">DrawPath</a> <a href="#">DrawRect</a> <a href="#">DrawRoundRect</a> <a href="#">DrawTextBlock</a> <a href="#">DrawTextLine</a> <a href="#">FillOval</a> <a href="#">FillPath</a> <a href="#">FillRect</a> <a href="#">FillRoundRect</a> <a href="#">RestoreState</a> <a href="#">Rotate</a> <a href="#">SaveState</a> <a href="#">Scale</a> <a href="#">TextBlockSize</a> <a href="#">TextLineSize</a> <a href="#">Translate</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOscanvas</a> <a href="#">iOSBitmap</a> <a href="#">iOSImage</a> <a href="#">iOSSPath</a>

UIKit

[CGContextRef](#)

## Notes

You cannot instantiate `iOSGraphics`. You can get an `iOSGraphics` instance from either an [iOSCanvas](#) or an [iOSBlitmap](#).

Changing the graphics state only affects subsequent drawing commands.

## Properties

### Alpha As Double

The Alpha value, which ranges from 0.0 (transparent) to 1.0 (opaque).

#### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.Alpha = 0.5
```

---

### AntiAlias As Boolean

Specifies whether anti-aliasing is used. The default is `True`.

#### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.AntiAlias = False
```

---

### FillColor As Color

The color used when drawing filled shapes and text.

#### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.FillColor = Color.Blue
```

---

### Handle As Ptr (read-only)

The handle for the [CGContextRef](#) used by the graphics object.

---

## Height As Integer (read-only)

The height of the graphics area, in points.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.LineColor = Color.Blue
g.DrawLine(0, 0, g.Width, g.Height)
```

---

## LineColor As Color

The color to use when drawing lines and shapes.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.LineColor = Color.Blue
g.DrawLine(0, 0, g.Width, g.Height)
```

---

## LineWidth As Double

The width (in points) to use when drawing lines and shapes.

### Sample Code

```
g.LineWidth = 10
g.DrawLine(0, 20, 50,20)
```

---

## TextFont As [iOSFont](#)

The font to use when drawing text.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.TextFont = New iOSFont("Helvetica Neue", 12)
g.FillColor = Color.Blue
g.DrawTextBlock("Hello, World!", 30, 30)
```

---

## TextUnderline As Boolean

Indicates if drawn text should be underlined.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.TextFont = New iOSFont("Helvetica Neue", 12)
g.TextUnderline = True
g.DrawTextBlock("Hello, World!", 30, 30)
```

---

## Width As Integer (read-only)

The width of the graphics area, in points.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.LineColor = Color.Blue
g.DrawLine(0, 0, g.Width, g.Height)
```

## Methods

### ClipToPath(path As [iOXPath](#))

Clips the drawing to the specified *path*.

#### Parameters

<i>path</i>	The path to use for clipping.
-------------	-------------------------------

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
// Clip to an iOXPath
// Path is a triangle
Dim p As New iOXPath
p.MoveToPoint(0, 0) // Start location
p.LineToPoint(20, 44)
p.LineToPoint(40, 0)
p.LineToPoint(0, 0)

g.ClipToPath(p)
g.FillOval(0, 0, 50, 50)
```

## ClipToRect(x As Double, y As Double, width As Double, height As Double)

Clips the drawing to the specified rectangle.

### Parameters

<i>x</i>	The x coordinate in points.
<i>y</i>	The y coordinate in points.
<i>width</i>	The width in points.
<i>height</i>	The height in points.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
// The part of the circle drawn outside the clip area
// is not displayed.
g.ClipToRect(0, 0, 50, 50)

g.FillColor = Color.Blue
g.FillOval(25, 25, 50, 50)
```

## DrawImage(pic As [UIImage](#), x As Double, y As Double)

## DrawImage(pic As [UIImage](#), x As Double, y As Double, width As Double, height As Double)

## DrawImage(pic As [IBitmap](#), x As Double, y As Double)

## DrawImage(pic As [IBitmap](#), x As Double, y As Double, width As Double, height As Double)

Draws the specified image (an [UIImage](#) or an [IBitmap](#)) at the x and y coordinates, optionally scaling the image to the specified *width* and *height*.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
// MyPicture is an image added to the project
g.DrawImage(MyPicture, 0, 0)

// Draw image at half its original size
g.DrawImage(MyPicture, 0, 0, MyPicture.Width/2, MyPicture.Height/2)
```

---

## DrawLine(x1 As Double, y1 As Double, x2 As Double, y2 As Double)

Draws a line from x1, y1 to x2, y2.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.LineColor = Color.Blue
g.DrawLine(0, 0, g.Width, g.Height)
```

---

## DrawOval(x As Double, y As Double, width As Double, height As Double)

Draws an oval where x and y are the top left corner. If *width* and *height* are the same, then a circle is drawn.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.DrawOval(50, 50, 25, 25)

// Draw circle in center of graphics area
Const kSize As Integer = 100
g.DrawOval((g.Width - kSize) / 2, (g.Height - kSize) / 2, kSize, kSize)
```

---

## DrawPath(path As iOSPath)

Draws the specified *path*.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
// Draw a triangle
Dim p As New iOSPath
p.MoveToPoint(0, 0) // Start location
p.LineToPoint(20, 44)
p.LineToPoint(40, 0)
p.LineToPoint(0, 0)

g.LineColor = Color.Blue
g.DrawPath(p)
```

---

## DrawRect(x As Double, y As Double, width As Double, height As Double)

Draws a rectangle.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.LineColor = Color.Blue
g.DrawRect(10, 10, 20, 20)
```

---

## DrawRoundRect(x As Double, y As Double, width As Double, height As Double, cornerWidth As Double, cornerHeight As Double)

Draws a rounded rectangle.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.LineColor = Color.Blue
g.DrawRoundRect(10, 10, 50, 50, 5, 5)
```

---

## DrawTextBlock(value As Text, x As Double, y As Double, maxWidth As Double = -1, maxHeight As Double = -1, alignment As iOSTextAlignment = iOSTextAlignment.Left, truncateLastLine As Boolean = False)

Draws a block of text.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.FillColor = Color.Blue
Dim t As Text
t = "How much wood would a woodchuck chuck if a woodchuck could chuck would?"
g.DrawTextBlock(t, 0, 10, g.Width, g.Height, iOSTextAlignment.Left, False)
```

---

## DrawTextLine(value As Text, x As Double, y As Double, maxWidth As Double = -1, alignment As iOSTextAlignment = iOSTextAlignment.Left, truncate As Boolean = False)

Draws a single line of text.

## Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.FillColor = Color.Blue
Dim t As Text
t = "Hello, World!"
g.DrawTextLine(t, 0, 10, -1, iOSTextAlignment.Left, False) // Left-aligned
g.DrawTextLine(t, 0, 10, -1, iOSTextAlignment.Center, False) // Centered
g.DrawTextLine(t, 0, 10, 100, iOSTextAlignment.Right, False) // Right-aligned
```

---

## FillOval(x As Double, y As Double, width As Double, height As Double)

Draws an oval, filling it with the `FillColor`.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.FillColor = Color.Blue
g.FillOval(10, 10, 20, 20)
```

---

## FillPath(path As iOSPath)

Draws a path, filling it with the `FillColor`.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
// Draw a triangle
Dim p As New iOSPath
p.MoveToPoint(0, 0) // Start location
p.LineToPoint(20, 44)
p.LineToPoint(40, 0)
p.LineToPoint(0, 0)

g.FillColor = Color.Blue
g.FillPath(p)
```

---

## FillRect(x As Double, y As Double, width As Double, height As Double)

Draws a rectangle, filling it with the `FillColor`.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.FillColor = Color.Blue  
g.FillRect(10, 10, 20, 20)
```

---

## FillRoundRect(x As Double, y As Double, width As Double, height As Double, cornerWidth As Double, cornerHeight As Double)

Draws a rounded rectangle, filling it with the FillColor.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.FillColor = Color.Blue  
g.FillRoundRect(10, 10, 50, 50, 5, 5)
```

---

## RestoreState

Restores graphics context previously saved with [SaveState](#).

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.FillColor = Color.Blue  
g.FillRect(10, 10, 20, 20)
```

```
g.SaveState  
g.FillColor = Color.Red  
g.FillRect(50, 50, 20, 20)
```

```
// Restore to state where FillColor is Blue  
g.RestoreState  
g.FillRect(10, 50, 20, 20)
```

---

## Rotate(angle As Double)

Rotates the drawing context by the specified *angle* (in radians) around the origin point (usually 0,0). This only affects subsequent drawing. Any drawing done before the Rotate method call is not rotated.

### Notes

Use `Translate` to move the origin.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
// Rotate square in center of graphics area
Const Pi = 3.14159
g.Translate(g.Width / 2, g.Height / 2)
g.Rotate(Pi / 4) // 45 degrees or 1/8 of a circle
g.FillColor = Color.Blue
g.FillRect(10, 10, 50, 50)
```

---

## Rotate(angle As Double, x As Double, y As Double)

Rotates the drawing context at the *x* and *y* coordinates by the specified *angle* (in radians). This only affects subsequent drawing. Any drawing done before the Rotate method called is not rotated.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
// Rotate the entire drawing area around its center
// and draw a rectangle
Const Pi = 3.14159
g.Rotate(Pi / 4, g.Width / 2, g.Height / 2) // 45 degrees or 1/8 of a circle
g.FillColor = Color.Blue
g.FillRect(g.Width / 2 - 20, g.Height / 2 - 20, 20, 20)
```

---

## SaveState

Saves graphics state of the graphics context so that it can be restored later.

### Notes

These values are saved with `SaveState` and are restored when you call [RestoreState](#).

- Translation
- Clip region
- LineWidth
- AntiAlias
- FillColor
- LineColor
- Alpha value
- TextFont
- TextUnderline
- Rotation

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.FillColor = Color.Blue
g.FillRect(10, 10, 20, 20)
```

```
g.SaveState
g.FillColor = Color.Red
g.FillRect(50, 50, 20, 20)

// Restore to state where FillColor is Blue
g.RestoreState
g.FillRect(10, 50, 20, 20)
```

---

## Scale(scaleX As Double, scaleY As Double)

Sets the scale for the graphics context as specified *scaleX* and *scaleY*. This only affects subsequent drawing. Any drawing done before the Scale method is called is not scaled.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
g.Scale(2.0, 4.0)
g.FillColor = Color.Blue
g.FillRect(10, 10, 10, 10) // The squares scales to a rectangle
```

---

**TextBlockSize** (value As Text, maxWidth As Double = -1, maxHeight As Double = -1, alignment As [iOSTextAlignment](#) = `iOSTextAlignment.Left`, truncateLastLine As Boolean = False) As [Size](#)

Calculates the size of a `TextBlock`.

---

**TextLineSize** (value As Text, maxWidth As Double = -1, alignment As [iOSTextAlignment](#) = `iOSTextAlignment.Left`, truncate As Boolean = False) As [Size](#)

Calculates the size of a `TextLine`.

---

## Translate(translateX As Double, translateY As Double)

Translates the origin point by the specified *translateX* and *translateY* values. Useful with `Rotate`.

### Sample Code

From within an `iOSCanvas.Paint` event handler:

```
// Rotate square in center of graphics area
Const Pi = 3.14159
g.Translate(g.Width / 2, g.Height / 2)
```

```
g.Rotate(Pi / 4) // 45 degrees or 1/8 of a circle  
g.FillColor = Color.Blue  
g.FillRect(10, 10, 50, 50)
```

# iOSHTMLViewer

Displays HTML in a scrollable area.

## Control Summary

Name	iOSHTMLViewer
Type	Control
Super	<a href="#">iOSControl</a>
Events	<a href="#">Close</a> <a href="#">Open</a>
Properties	<a href="#">AccessibilityHint</a> <a href="#">AccessibilityLabel</a> <a href="#">Height</a> <a href="#">Left</a> <a href="#">Name</a> <a href="#">Parent</a> <a href="#">Top</a> <a href="#">Visible</a> <a href="#">Width</a>
Methods	<a href="#">LoadURL</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">HTTPSsocket</a>
Example Projects	Examples/iOS/Controls/HTMLViewerExample

## Methods

### LoadURL(url As [Text](#))

Displays the specified URL. Be sure to include the appropriate prefix, such as "http://".

#### Parameters

<i>url</i>	The URL to display.
------------	---------------------

#### Notes

To load HTML from a file on the device, use pass [FolderItem.URLPath](#) to LoadURL.

## Sample Code

Display wikipedia:

```
HTMLViewer1.LoadURL("http://www.wikipedia.org")
```

To save HTML to a file and then load it:

```
Dim html As Text = "<html><body>Hello!</body></html>"
```

```
Dim htmlFile As FolderItem = SpecialFolder.Documents.Child("hello.html")
```

```
Dim output As TextOutputStream
```

```
output = TextOutputStream.Create(htmlFile, TextEncoding.UTF8)
```

```
output.WriteLine(html)
```

```
output.Close
```

```
HTMLViewer1.LoadURL(htmlFile.URLPath)
```

# iOSImage

A pre-existing image that was either added to the project or is loaded from another source.

## Class Summary

Name	iOSImage
Type	Class
Properties	<a href="#">Handle</a> <a href="#">Height</a> <a href="#">Scale</a> <a href="#">Width</a>
Methods	<a href="#">ToData</a> <a href="#">WriteToFile</a>
Shared Methods	<a href="#">FromData</a> <a href="#">FromFile</a> <a href="#">FromHandle</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSBitmap</a>
UIKit	<a href="#">UIImage</a>

## Notes

An `iOSImage` is immutable, which means that it cannot be changed. Because of this it can be cached by iOS for speed. If you need to change an image, you should instead create it as an [iOSBitmap](#).

## Properties

### Handle As Ptr (read-only)

The [UIImage](#) handle to use when calling OS APIs through `Declares`.

### Height As UInteger (read-only)

The height of the image.

## Scale As Double (read-only)

The scale of the image.

---

## Width As UInteger (read-only)

The width of the image.

## Methods

### ToData(type As Text) As MemoryBlock

Converts the image to data that can be used for storing in databases, for example. Use type to specify the uniform type identifier for the data. For example, use "public.png" for PNG data.

#### Notes

Common types include "public.jpeg", "public.tiff" and "public.png".

Apple provides a list of [uniform type identifiers](#).

---

### WriteToFile(file As FolderItem, type As Text)

Writes the image to a file. Use type to specify the uniform type identifier for the data. For example, use "public.png" for PNG data.

#### Notes

Common types include "public.jpeg", "public.tiff" and "public.png".

Apple provides a list of [uniform type identifiers](#).

## Shared Methods

### FromData(data As MemoryBlock) As UIImage

Creates an image from *data* in a MemoryBlock.

#### Sample Code

```
// Creates an image from image data stored in a database field
Dim photo As UIImage
photo = UIImage.FromData(customerRecordSet.Field("Photo").NativeValue)
```

---

### FromFile(file As FolderItem) As UIImage

Creates an image from a file.

## Sample Code

```
Dim imageFile As FolderItem
imageFile = SpecialFolder.Documents.Child("MyImage.png")
Dim image As UIImage
image = UIImage.FromFile(file)
ImageView1.Image = image
```

---

## FromHandle(imageHandle As Ptr) As UIImage

Creates an image from an iOS [UIImage](#).

As of iOS 8.1, here are the supported formats that can be used with [UIImage.ToData](#) and [WriteToFile](#).

## Input Formats

Portable Network Graphics image	public.png
JPEG image	public.jpeg
Graphics Interchange Format (GIF)	com.compuserve.gif
Canon TIF raw image	com.canon.tif-raw-image
Adobe raw image	com.adobe.raw-image
Canon CR2 raw image	com.canon.cr2-raw-image
Leaf raw image	com.leafamerica.raw-image
Hasselblad FFF raw image	com.hasselblad.fff-raw-image
Hasselblad 3FR raw image	com.hasselblad.3fr-raw-image
Nikon raw image	com.nikon.raw-image
Nikon NRW raw image	com.nikon.nrw-raw-image
Pentax raw image	com.pentax.raw-image
Samsung raw image	com.samsung.raw-image
Sony raw image	com.sony.raw-image
Sony SR2 raw image	com.sony.sr2-raw-image
Sony ARW raw image	com.sony.arw-raw-image
Epson raw image	com.epson.raw-image
Kodak raw image	com.kodak.raw-image
TIFF image	public.tiff
Canon CRW raw image	com.canon.crw-raw-image
Fuji raw image	com.fuji.raw-image
Panasonic raw image	com.panasonic.raw-image
Panasonic raw 2 image	com.panasonic.rw2-raw-image
Leica raw image	com.leica.raw-image
Leica RWL raw image	com.leica.rwl-raw-image
Minolta raw image	com.konicaminolta.raw-image

Olympus SR raw image	com.olympus.sr-raw-image
Olympus OR raw image	com.olympus.or-raw-image
Windows icon image	com.microsoft.ico
Windows bitmap image	com.microsoft.bmp
Windows cursor image	com.microsoft.cur
TGA image	com.truevision.tga-image
JPEG 2000 image	public.jpeg-2000
Multiple Picture Object image	public.mpo-image
Netpbm	public.pbm

## Output formats

Portable Network Graphics image	public.png
JPEG image	public.jpeg
Graphics Interchange Format (GIF)	com.compuserve.gif
TIFF image	public.tiff
Windows icon image	com.microsoft.ico
Windows bitmap image	com.microsoft.bmp
Portable Document Format (PDF)	com.adobe.pdf
TGA image:	com.truevision.tga-image
JPEG 2000 image	public.jpeg-2000
Netpbm	public.pbm

# UIImageView

Displays an image.

## Control Summary

Name	UIImageView
Type	Control
Super	<a href="#">iOSControl</a>
Enumerations	<a href="#">ContentModes</a>
Events	<a href="#">Close</a> <a href="#">Open</a>
Properties	<a href="#">AccessibilityHint</a> <a href="#">AccessibilityLabel</a> <a href="#">ContentMode</a> <a href="#">Height</a> <a href="#">Image</a> <a href="#">Left</a> <a href="#">Name</a> <a href="#">Parent</a> <a href="#">Top</a> <a href="#">Visible</a> <a href="#">Width</a>
Methods	<a href="#">Handle</a> <a href="#">Invalidate</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">UIImage</a>

## Enumerations

### ContentModes

Indicates how the image is stretched, scaled or aligned when it is displayed.

ScaleToFill	The image's aspect ratio is ignored and the image is stretched to fit within the bounds of the frame.
ScaleAspectFit	Scales the image until it fits entirely, maintaining aspect.
ScaleAspectFill	Scales the image until it fills the UIImageView entirely, maintaining aspect.
Center	The image is centered within the frame.

Top	The image is positioned with its top at the top of the frame.
Bottom	The image is positioned with its bottom at the bottom of the frame.
Left	The image is positioned with its left at the left of the frame.
Right	The image is positioned with its right at the right of the frame.
Topleft	The image is positioned with its top left at the top left of the frame.
TopRight	The image is positioned with its top right at the top right of the frame.
BottomLeft	The image is positioned with its bottom left at the bottom left of the frame.
BottomRight	The image is positioned with its bottom right at the bottom right of the frame.

## Properties

### ContentMode

Specifies the way the picture gets displayed using a [ContentModes](#) enumeration.

#### Sample Code

Center the image within the frame:

```
Me.ContentMode = UIImageView.ContentModes.Center
```

---

### Image As [UIImage](#)

The image to display.

#### Sample Code

Display an image in the project:

```
Me.Image = MyCompanyLogo
```

# iOSLabel

This is the standard Label control for iOS.

## Control Summary

Name	<a href="#">iOSLabel</a>
Type	Control
Super	<a href="#">iOSControl</a>
Events	<a href="#">Close</a> <a href="#">Open</a>
Properties	<a href="#">AccessibilityHint</a> <a href="#">AccessibilityLabel</a> <a href="#">Enabled</a> <a href="#">Height</a> <a href="#">Left</a> <a href="#">LineBreakMode</a> <a href="#">Name</a> <a href="#">Parent</a> <a href="#">Text</a> <a href="#">TextAlignment</a> <a href="#">TextColor</a> <a href="#">TextFont</a> <a href="#">Top</a> <a href="#">Visible</a> <a href="#">Width</a>
Methods	<a href="#">Handle</a> <a href="#">Invalidate</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSFont</a> <a href="#">iOS Decorations</a>
UIKit	<a href="#">UILabel</a>

## Properties

### Enabled As [Boolean](#)

Indicates if the label is enabled or disabled.

### Sample Code

Disable a label:

```
Label1.Enabled = False
```

---

## LineBreakMode As iOSLineBreakMode

Specifies how text contents break or wrap when the text is too long to fit. The default is `iOSLineBreakMode.BreakByWordWrapping`.

Sample Code

```
Label1.LineBreakMode = Xojo.iOSLineBreakMode.BreakByTruncatingMiddle
```

---

## Text As Text

The text to display in the Label.

Sample Code

Set the label text:

```
Label1.Text = "Hello, World!"
```

---

## TextAlignment As iOSTextAlignment

Specifies the alignment for the text. The default value is `iOSTextAlignment.Left`.

Sample Code

```
Label1.TextAlignment = iOSTextAlignment.Center
```

---

## TextColor As Color

The color of the text.

Sample Code

```
Label1.TextColor = Color.Blue
```

---

## TextFont As iOSFont

The font used to display the text.

## Sample Code

```
Label1.TextFont = iOSFont.BoldSystemFont(40)
```

# iOSLayoutConstraint

Used to add new auto-layout constraints or modify existing ones.

## Class Summary

Name	iOSLayoutConstraint
Type	Class
Enumerations	<a href="#">AttributeTypes</a> <a href="#">RelationTypes</a>
Properties	<a href="#">Active</a> <a href="#">FirstAttribute</a> <a href="#">FirstItem</a> <a href="#">Offset</a> <a href="#">Priority</a> <a href="#">Relation</a> <a href="#">SecondAttribute</a> <a href="#">SecondItem</a> <a href="#">Scale</a>
Shared Properties	<a href="#">StandardGap</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSView.Constraint</a> <a href="#">Auto-Layout in iOS Guide</a>

## Constructors

Constructor(*firstItem* As Object, *firstAttribute* As [AttributeTypes](#), *relation* As [RelationTypes](#), *secondItem* As Object, *secondAttribute* As [AttributeTypes](#), *multiplier* As Double, *addend* As Double)

Creates a new auto-layout rule for the *firstItem* control with an *addend* offset value.

### Parameters

<i>firstItem</i>	The Control to add the constraint to.
<i>firstAttribute</i>	The attribute of the control to constrain.
<i>relation</i>	The relationship to the <i>secondItem</i> 's attribute. Typically this is <a href="#">RelationTypes.Equal</a>
<i>secondItem</i>	The related control or containing view. This can also be the <a href="#">TopLayoutGuide</a> or <a href="#">BottomLayoutGuide</a> of the View.

<i>secondAttribute</i>	The attribute of the related item to use.
<i>multiplier</i>	A multiplier to adjust the <i>secondAttribute</i> value. Typically this is just 1.0.
<i>addend</i>	The offset. If you need the <i>StandardGap</i> , use the Constructor below.

### Sample Code

Adds a right constraint to a *TextField* with an offset of 50:

```
Dim right As New iOSLayoutConstraint(RightConstraintField, _
    iOSLayoutConstraint.AttributeTypes.Right, _
    iOSLayoutConstraint.RelationTypes.Equal, _
    Self, iOSLayoutConstraint.AttributeTypes.Right, _
    1.0, 50)
```

**Constructor**(*firstItem* As Object, *firstAttribute* As AttributeTypes, *relation* As RelationTypes, *secondItem* As Object, *secondAttribute* As AttributeTypes, *multiplier* As Double, *gap* As StandardGap)

Creates a new auto-layout rule for the *firstItem* control with a *StandardGap* offset value.

### Parameters

<i>firstItem</i>	The Control to add the constraint to.
<i>firstAttribute</i>	The attribute of the control to constrain.
<i>relation</i>	The relationship to the <i>secondItem</i> 's attribute. Typically this is <i>RelationTypes.Equal</i>
<i>secondItem</i>	The related control or containing view. This can also be the <u>TopLayoutGuide</u> or <u>BottomLayoutGuide</u> of the View.
<i>secondAttribute</i>	The attribute of the related item to use.
<i>multiplier</i>	A multiplier to adjust the <i>secondAttribute</i> value. Typically this is just 1.0.
<i>gap</i>	Always <i>iOSLayoutConstraint.StandardGap</i> . If you need specific offset, use the Constructor above.

### Sample Code

Adds a right constraint to a *TextField* with an offset of the *StandardGap*:

```
Dim right As New iOSLayoutConstraint(RightConstraintField, _
    iOSLayoutConstraint.AttributeTypes.Right, _
    iOSLayoutConstraint.RelationTypes.Equal, _
    Self, iOSLayoutConstraint.AttributeTypes.Right, _
    1.0, iOSLayoutConstraint.StandardGap)
```

## Enumerations

### AttributeTypes

The auto-layout attributes.

None	No constraint.
Left	The left position of the control.
Right	The right position of the control.
Top	The top position of the control.
Bottom	The bottom position of the control.
Leading	
Trailing	
Width	The width of the control.
Height	The height of the control.
CenterX	Horizontal center
CenterY	Vertical center
Baseline	
LeftMargin	*iOS 8 only
RightMargin	*iOS 8 only
TopMargin	*iOS 8 only
BottomMargin	*iOS 8 only
LeadingMargin	*iOS 8 only
TrailingMargin	*iOS 8 only

### RelationTypes

The auto-layout relation.

LessThanOrEqualTo	The value must be less than or equal to the value of the related control.
EqualTo	The value must be equal to the value of the related control.
GreaterThanOrEqualTo	The value must be greater than or equal to the value of the related control.

## Properties

## Active As Boolean

Indicates if this auto-layout rule is active.

### Sample Code

De-activate an existing named constraint:

```
// "TAWidth" is a width constraint for a TextField that has been given
// a name in the auto-layout Inspector properties.
Dim c As iosLayoutConstraint = Self.Constraint("TAWidth")
c.Active = False
```

## FirstAttribute As AttributeTypes (read-only)

The attribute of the first control.

## FirstItem As Object (read-only)

The control to which the auto-layout rule applies.

## Offset As Double

The offset that can be applied to the rule.

### Sample Code

Change the offset (in this case the width) or an existing named constraint:

```
// "TAWidth" is a width constraint for a TextField that has been given
// a name in the auto-layout Inspector properties.
Dim c As iosLayoutConstraint = Self.Constraint("TAWidth")
c.Offset = 200
```

## Priority As Double (read-only)

The priority of the rule.

### Notes

Higher priority constraints are met before lower priority constraints.

Priority	Value
Priority Required	1000 or greater
Optional Priority	Less than 1000

Priority	Value
High Priority	750
Low Priority	250
Auto	50?

 You cannot change a priority from a required value (1000 or greater) to an optional value (less than 1000).

## Relation As RelationTypes (read-only)

The type of relation between the first control and the second control. This is the "Is" item in the Inspector.

## SecondAttribute As AttributeTypes (read-only)

The attribute to use from the second (related) control. This is the "Edge" item in the Inspector.

## SecondItem As Object (read-only)

The related control for the rule. This is the "Relative To" item in the Inspector.

## Scale As Double (read-only)

The scale (aka multiplier) for the rule.

## Shared Properties

### StandardGap

Used to supply a standard gap value when creating a new auto-layout rule. This item can only be used for the *gap* parameter in the constructor. It has no defined type.

# iOSLine

This is the standard Line control for iOS.

## Control Summary

Name	iOSLine
Type	Control
Super	<a href="#">IOSControl</a>
Events	<a href="#">Close</a> <a href="#">Open</a>
Properties	<a href="#">Direction</a> <a href="#">Left</a> <a href="#">LineColor</a> <a href="#">LineWidth</a> <a href="#">Top</a> <a href="#">Visible</a>
Methods	<a href="#">Handle</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSSoval</a> <a href="#">iOSRectangle</a> <a href="#">iOSSeparator</a> <a href="#">iOS Decorations</a>

## Properties

### Direction As Integer

The direction of the line.

0	Horizontal
1	Vertical
2	Lower left to upper right
3	Upper left to lower right

### LineColor As Color

The color of the line.

## LineWidth As Double

The width of the line.

# iOSMessageBox

Used to asynchronously display alerts.

## Class Summary

Name	iOSMessageBox
Type	Class
Events	<a href="#">ButtonAction</a>
Properties	<a href="#">Message Title</a>
Methods	<a href="#">Buttons Show</a>
Project Types	iOS
Platforms	iOS
Related	

## Notes

Since `iOSMessageBox` is asynchronous, any code after the `Show` method is executed after the message box is displayed. Use the `ButtonAction` event handler to get the results of a button selection and to run code based on the selection.

## Events

### `ButtonAction(buttonIndex As Integer)`

Called when the button at *buttonIndex* (0-based) is selected.

Sample Code

```
Label1.Text = "You selected button " + buttonIndex.ToText</code>
```

## Properties

### Message As Text

A longer message that appears below the title in the message box.

---

## Title As Text

The title of the message box, which appears at the top.

## Methods

### Buttons As Text()

### Buttons(Assigns value() As Text)

Gets or sets the buttons for the message box.

### Sample Code

Displays a message box with multiple buttons. HelloMessage is an `iOSMessageBox` that was dragged onto the View from the Library:

```
HelloMessage.Title = "Hello"  
HelloMessage.Message = "Hello, World!"
```

```
Dim buttons() As Text  
buttons.Append("Yes")  
buttons.Append("No")
```

```
HelloMessage.Buttons = buttons
```

```
HelloMessage.Show
```

---

## Show

Displays the message box. This is not modal, so your code continues to run. When the user selects a button, the [ButtonAction](#) event handler is called.

### Sample Code

Displays a message box. HelloMessage is an `iOSMessageBox` that was dragged onto the View from the Library:

```
HelloMessage.Title = "Hello"  
HelloMessage.Message = "Hello, World!"  
HelloMessage.Show // Not modal, so code does not pause here</code>
```

# iOSOval

This is the standard Oval control for iOS.

## Control Summary

Name	iOSOval
Type	Control
Super	<a href="#">iOSControl</a>
Events	<a href="#">Close</a> <a href="#">Open</a>
Properties	<a href="#">BorderColor</a> <a href="#">BorderWidth</a> <a href="#">FillColor</a> <a href="#">Height</a> <a href="#">Left</a> <a href="#">Top</a> <a href="#">Visible</a> <a href="#">Width</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">Color</a> <a href="#">iOSLine</a> <a href="#">iOSRectangle</a> <a href="#">iOS Decorations</a>

## Properties

### BorderColor As [Color](#)

The color of the border.

### BorderWidth As [Double](#)

The width of the border.

### FillColor As [Color](#)

The fill color.

# iOSPath

A graphics path is a mathematical description of a series of shapes or lines.

## Class Summary

Name	iOSPath
Type	Class
Properties	<a href="#">CurrentPoint</a> <a href="#">IsEmpty</a> <a href="#">IsRectangle</a>
Methods	<a href="#">AddArc</a> <a href="#">AddCurveToPoint</a> <a href="#">AddQuadraticCurveToPoint</a> <a href="#">AddRect</a> <a href="#">AddRoundRect</a> <a href="#">LineToPoint</a> <a href="#">MoveToPoint</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSGraphics</a>
UIKit	<a href="#">UIBezierPath</a>
Examples	Examples/iOS/Graphics/PathExample

## Properties

### CurrentPoint As Point (read-only)

The current point.

### IsEmpty As Boolean (read-only)

Checks if the path is empty. An empty path contains no elements.

### IsRectangle As Boolean (read-only)

Checks if the path is a rectangle.

## Methods

### AddArc(x As Double, y As Double, radius As Double, startAngle As Double, endAngle As Double, clockwise As Boolean)

Adds an arc to the path.

#### Notes

The ending point of the Arc becomes the start point for the next element added to the path. In particular, if you add two arcs in a row, there will be a line that connects the ending point of the first arc to the starting point of the second arc. If you do not want this behavior, set the new starting point manually using [MoveToPoint](#).

#### Parameters

<i>x</i>	The x-coordinate of the center point of the arc.
<i>y</i>	The y-coordinate of the center point of the arc.
<i>radius</i>	The radius of the arc.
<i>startAngle</i>	The angle (in radians) that determines the starting point of the arc, measured from the x-axis.
<i>endAngle</i>	The angle (in radians) that determines the ending point of the arc, measured from the x-axis.
<i>clockwise</i>	A Boolean value that specifies whether or not to draw the arc in the clockwise direction.

#### Sample Code

```
// Draw an arc that is 1/4 of a circle
Const Pi = 3.14159
Dim arc As New iOPath
arc.AddArc(50, 50, 20, 0, Pi/2, False)
g.DrawPath(arc)
```

### AddCurveToPoint(cp1x As Double, cp1y As Double, cp2X As Double, cp2Y As Double, x As Double, y As Double)

Adds a [Bézier curve](#) to the point in the path.

#### Parameters

<i>cp1x</i>	The x-coordinate of the first control point.
<i>cp1y</i>	The y-coordinate of the first control point.
<i>cp2X</i>	The x-coordinate of the second control point.
<i>cp2Y</i>	The y-coordinate of the second control point.

<i>x</i>	The x-coordinate of the end point of the curve.
<i>y</i>	The y-coordinate of the end point of the curve.

### Sample Code

```
Dim curve As New iOPath
curve.MoveToPoint(20, 20)
curve.AddCurveToPoint(20, 100, 200, 100, 200, 20)
```

```
g.DrawPath(curve)
```

```
// Draw a fluffy white cloud
Dim cloud As New iOPath
cloud.MoveToPoint(170, 80)
cloud.AddCurveToPoint(130, 100, 130, 150, 230, 150)
cloud.AddCurveToPoint(250, 180, 320, 180, 340, 150)
cloud.AddCurveToPoint(420, 150, 420, 120, 390, 100)
cloud.AddCurveToPoint(430, 40, 370, 30, 340, 50)
cloud.AddCurveToPoint(320, 5, 250, 20, 250, 50)
cloud.AddCurveToPoint(200, 5, 150, 20, 170, 80)
```

```
g.LineColor = Color.Blue
g.LineWidth = 5
g.DrawPath(cloud)
```

## AddQuadraticCurveToPoint(*cpX* As Double, *cpY* As Double, *x* As Double, *y* As Double)

Adds a quadratic curve to the point in the path.

### Parameters

<i>cpX</i>	The x-coordinate of the control point.
<i>cpY</i>	The y-coordinate of the control point.
<i>x</i>	The x-coordinate of the end point of the curve.
<i>y</i>	The y-coordinate of the end point of the curve.

### Sample Code

```
Dim qCurve As New iOPath
qCurve.MoveToPoint(38, 150)
qCurve.AddQuadraticCurveToPoint(138, 0, 238, 150)
```

```
g.LineWidth = 10  
g.DrawPath(qCurve)
```

---

## AddRect(x As Double, y As Double, width As Double, height As Double)

Adds a rectangle to the path.

### Parameters

<i>x</i>	The left position of the rectangle.
<i>y</i>	The top position of the rectangle.
<i>width</i>	The width of the rectangle.
<i>height</i>	The height of the rectangle.

### Sample Code

```
Dim rect As New iOPath  
rect.AddRect(10, 10, 100, 150)  
g.DrawPath(rect)
```

---

## AddRoundRect(x As Double, y As Double, width As Double, height As Double, cornerWidth As Double, cornerHeight As Double)

Adds a rounded rectangle to the path.

### Parameters

<i>x</i>	The left position of the rectangle.
<i>y</i>	The top position of the rectangle.
<i>width</i>	The width of the rectangle.
<i>height</i>	The height of the rectangle.
<i>cornerWidth</i>	The width of the rounded corner sections.
<i>cornerHeight</i>	The height of the rounded corner sections.

### Sample Code

```
Dim rect As New iOPath  
rect.AddRoundRect(10, 10, 100, 150, 10, 10)  
g.DrawPath(rect)
```

---

## LineToPoint(x As Double, y As Double)

Draws a line from the [CurrentPoint](#) to the specified point.

### Parameters

x	The x coordinate for the end of the line.
y	The y coordinate of the end of the line.

### Sample Code

```
// Draw a triangle
Dim p As New iOPath
p.MoveToPoint(0, 0) // Start location
p.LineToPoint(20, 44)
p.LineToPoint(40, 0)
p.LineToPoint(0, 0)

g.LineColor = Color.Blue
g.DrawPath(p)
```

---

## MoveToPoint(x As Double, y As Double)

Moves to the point without drawing anything.

### Parameters

x	The x coordinate of the point.
y	The y coordinate of the point.

### Sample Code

```
Dim p As New iOPath
p.MoveToPoint(50, 50)
```

## Operators

The comparison operator is supported.

# UIProgressView

The `UIProgressView` is used to display progress using a horizontal line.

## Control Summary

Name	<code>UIProgressView</code>
Type	Control
Super	<a href="#">UIView</a>
Events	<a href="#">Close</a> <a href="#">Open</a>
Properties	<a href="#">AccessibilityHint</a> <a href="#">AccessibilityLabel</a> <a href="#">CurrentValue</a> <a href="#">Height</a> <a href="#">Left</a> <a href="#">MaxValue</a> <a href="#">MinValue</a> <a href="#">Name</a> <a href="#">Parent</a> <a href="#">Top</a> <a href="#">Visible</a> <a href="#">Width</a>
Methods	<a href="#">Handle</a> <a href="#">SetMinCurrentMax</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">UIProgressWheel</a> <a href="#">iOS Indicators</a>
Example Projects	Examples/iOS/Controls/ProgressExample

## Properties

### CurrentValue As Double

The current position of the progress bar.

#### Sample Code

Increase the current position:

```
MyProgressBar.CurrentValue = MyProgressBar.CurrentValue + 1</code>
```

---

## MaxValue As Double

The maximum value of the progress bar. When `CurrentValue` reaches `MaxValue`, the progress bar appears "full".

### Sample Code

Only update the current value if it is below the maximum:

```
If MyProgressBar.Value < MyProgressBar.MaxValue Then
    MyProgressBar.Value = MyProgressBar.Value + 1
End If</code>
```

---

## MinValue As Double

The minimum value of the progress bar. When `CurrentValue` is `MinValue`, the progress bar appears "empty".

### Sample Code

Use a non-zero minimum value:

```
Me.MinValue = 15</code>
```

## Methods

### SetMinCurrentMax(minValue As Double, currentValue As Double, maxValue As Double)

Sets all the properties at one time.

#### Parameters

<i>minValue</i>	Sets the <code>MinValue</code> property.
<i>currentValue</i>	Sets the <code>CurrentValue</code> property.
<i>maxValue</i>	Sets the <code>MaxValue</code> property.

#### Notes

Using this method is more efficient than setting the properties individually. It also avoids the animation that may show when you update the properties separately.

### Sample Code

Set all values at once:

```
Me.SetMinCurrentMax(0, 15, 100)
```

# iOSProgressWheel

The ProgressWheel is an animated spinning wheel used to indicate that a task is in process.

## Control Summary

Name	iOSProgressWheel
Type	Control
Super	<a href="#">iOSControl</a>
Enumerations	<a href="#">Shade</a>
Events	<a href="#">Close</a> <a href="#">Open</a>
Properties	<a href="#">AccessibilityHint</a> <a href="#">AccessibilityLabel</a> <a href="#">Height</a> <a href="#">Left</a> <a href="#">Name</a> <a href="#">Parent</a> <a href="#">Shade</a> <a href="#">Top</a> <a href="#">Visible</a> <a href="#">Width</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSProgressBar</a> <a href="#">iOS Indicators</a>
Example Projects	Examples/iOS/Controls/ProgressExample

## Enumerations

### Shade

Shade options for the ProgressWheel.

Light	Visible on darker backgrounds.
Dark	Visible on lighter backgrounds.

## Properties

## Shade As Shade

The shade of the progress wheel (light or dark). The default is Dark.

### Sample Code

Set the shade to dark:

```
ProgressWheel1.Shade = iOSProgressWheel.Shade.Dark
```

# iOSRectangle

The standard rectangle control for iOS.

## Control Summary

Name	iOSRectangle
Type	Control
Super	<a href="#">iOSControl</a>
Events	<a href="#">Close</a> <a href="#">Open</a>
Properties	<a href="#">AccessibilityHint</a> <a href="#">AccessibilityLabel</a> <a href="#">BorderColor</a> <a href="#">BorderWidth</a> <a href="#">CornerHeight</a> <a href="#">CornerWidth</a> <a href="#">FillColor</a> <a href="#">Height</a> <a href="#">Left</a> <a href="#">Name</a> <a href="#">Parent</a> <a href="#">Top</a> <a href="#">Visible</a> <a href="#">Width</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSLine</a> <a href="#">iOSSoval</a> <a href="#">iOS Decorations</a>

## Properties

### BorderColor As [Color](#)

The color of the border.

### BorderWidth As [Double](#)

The width of the border.

## CornerHeight As Double

The height of the corner. Used to create rounded corners.

---

## CornerWidth As Double

The width of the corner. Used to create rounded corners.

---

## FillColor As Color

The fill color.

# iOSScreen

An iOS screen that controls specific layouts depending on the type of device being used.

## Class Summary

Name	iOSScreen
Type	Class
Properties	<a href="#">Content</a>
Methods	<a href="#">Handle</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSSplitView</a> <a href="#">iOSTabBar</a> <a href="#">iOSView</a>

## Notes

You cannot directly add controls to an iOSScreen.

An iOSScreen allows you to specify how your app looks depending on the type of device being used. For example, on an iPhone you can have your app display a single View that contains a Table with a list of items. To do this, you select the View for the Content property in the Inspector for the [DefaultiPhoneScreen](#). The iPhoneScreen is set in the App Inspector as the default screen to use for an iPhone.

On an iPad, you may instead want to have a split screen with a list of items on the left and detail information on the right. You can do this by setting the Content property for [DefaultiPadScreen](#) to be Split and then setting the views for the Main and Detail sections.

## Properties

### Content As [iOSScreenContent](#)

The content to display on the screen. This can be a single [View](#), a [SplitView](#) (consisting of a main and a detail view) or a [TabBar](#) (consisting of multiple Views).

## Methods

### Handle As [Ptr](#)

The iOS pointer to the screen handle.

# iOSScreenContent (Interface)

Identifies iOS UI layouts that can be used as a screen.

## Interface Summary

Name	iOSScreenContent
Type	Interface
Implements	<a href="#">iOSViewController</a>
Methods	<a href="#">ViewControllerHandle</a> , <a href="#">ViewResized</a> , <a href="#">ViewWillAppear</a> , <a href="#">ViewWillDisappear</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSSplitView</a> , <a href="#">iOSTabBar</a> , <a href="#">iOSView</a>

## Notes

A screen can contain a single [iOSView](#), an [iOSSplitView](#) (consisting of a main view and a detail view) and an [iOSTabBar](#), which consists of multiple views.

# iOSSegmentedControl

A horizontal button made up of multiple, independent segments.

## Control Summary

Name	iOSSegmentedControl
Type	Control
Super	<a href="#">iOSControl</a>
Events	<a href="#">Close</a> <a href="#">Open</a> <a href="#">ValueChanged</a>
Properties	<a href="#">AccessibilityHint</a> <a href="#">AccessibilityLabel</a> <a href="#">Height</a> <a href="#">Left</a> <a href="#">Name</a> <a href="#">Parent</a> <a href="#">Top</a> <a href="#">Value</a> <a href="#">Visible</a> <a href="#">Width</a>
Methods	<a href="#">Add</a> <a href="#">Count</a> <a href="#">Handle</a> <a href="#">Insert</a> <a href="#">Item</a> <a href="#">RemoveAll</a> <a href="#">RemoveByIndex</a> <a href="#">RemoveByValue</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSSegmentedControlItem</a> <a href="#">iOS Buttons</a>
Example Projects	Examples/iOS/Controls/SegmentedControl

## Events

### ValueChanged

Called when the selected segment changes.

## Properties

### Value As Integer

The currently selected segment.

## Methods

### Add(item As iOSSegmentedControlItem)

Adds a segment to the end.

#### Sample Code

Add segments:

```
Me.Add(New iOSSegmentedControlItem("Earth"))
Me.Add(New iOSSegmentedControlItem("Sun"))
Me.Add(New iOSSegmentedControlItem("Moon"))
```

---

### Count As Integer

The number of segments.

---

### Insert(index As Integer, item As iOSSegmentedControlItem)

Inserts a segment at the specified *index*.

---

### Item(index As Integer) As iOSSegmentedControlItem

### Item(index As Integer, Assigns newItem As iOSSegmentedControlItem)

Gets or Sets a specific segment.

---

### RemoveAll

Removes all segments.

---

### RemoveByIndex(index As Integer)

Removes a specific segment by its index.

## RemoveByValue(item As iOSSegmentedControlItem)

Removes a specific segment.

# UISegmentedControlItem

An item to display within the SegmentedControl.

## Item Summary

Name	UISegmentedControlItem
Type	Control
Properties	<a href="#">Icon</a> <a href="#">Selected</a> <a href="#">Title</a> <a href="#">Width</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">UISegmentedControl</a> <a href="#">iOS Buttons</a>
Example Projects	Examples/iOS/Controls/SegmentedControl

## Constructors

### Constructor(icon As Image)

Creates an item using the specified image.

#### Sample Code

In the Open event handler of an UISegmentedControl, adds a segment with an icon:

```
Me.Add(New UISegmentedControlItem(MyIcon))
```

### Constructor(title As Text = "")

Creates an item using the optional title.

#### Sample Code

In the Open event handler of an UISegmentedControl, adds a segment with a title:

```
Me.Add(New UISegmentedControlItem("Moon"))
```

## Properties

### Icon As UIImage

The icon to display.

#### Notes

Although you can have both an Icon and a Title, iOS only shows one of them, preferring the icon if available.

If you use an icon that is too large (about 28x28 seems to be the maximum), you will only see an empty square.

Only the mask for the icon is used, displayed using the tint color for iOS (currently blue), so be sure to use a PNG with an alpha channel.

#### Sample Code

Changes the icon for the first segment:

```
MySegmentedControl.Item(0).Icon = MyNewIcon</code>
```

---

### Selected As Boolean

Indicates if the item is selected.

#### Notes

Only one segment can display as selected. If multiple segments have Selected = True, then the last one with Selected = True is the one that displays as selected.

#### Sample Code

Set the first segment as selected:

```
MySegmentedControl.Item(0).Selected = True</code>
```

---

### Title As Text

The title text of the item.

#### Notes

Although you can have both an Icon and a Title, iOS only shows one of them, preferring the icon if available.

#### Sample Code

Changes the title for the first segment:

```
MySegmentedControl.Item(0).Title = "First!"
```

## Width As Double

The width of the item. Leave at 0 to have the segment use the available space. If the widths of all segments are 0, then the space will be divided equally amongst the segments.

### Sample Code

Change the first segment width:

```
MySegmentedControl.Item(0).Width = 50
```

# iOSSeparator

Used to separate sections of a layout.

## Control Summary

Name	iOSSeparator
Type	Control
Super	<a href="#">iOControl</a>
Events	<a href="#">Close</a> <a href="#">Open</a>
Properties	<a href="#">Height</a> <a href="#">Left</a> <a href="#">Top</a> <a href="#">Visible</a> <a href="#">Width</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSLine</a> <a href="#">iOS Decorations</a>

# iOSSound

Plays sounds.

## Control Summary

Name	iOSSound
Type	Class
Properties	<a href="#">IsPlaying</a> <a href="#">Pan</a> <a href="#">Volume</a>
Methods	<a href="#">Clone</a> <a href="#">Play</a> <a href="#">PlayLooping</a> <a href="#">Stop</a>
Project Types	iOS
Platforms	iOS
Related	
Example Projects	Examples/iOS/Sound/GameBuzzer Examples/iOS/Sound/NatureSounds

## Constructors

### Constructor(file As [FolderItem](#))

Creates a new sound from a sound file.

## Properties

### IsPlaying As Boolean (read-only)

Indicates if the sound is currently playing.

### Pan As Integer

Specifies the relative volume (balance) between the left and right sound channels.

#### Notes

The value ranges from -100 to +100, with 0 indicating equal volume in both channels. -100 plays sound in only the left channel, 100 plays sound only in the right channel.

## Volume As Integer = 100

Specifies the volume level of the sound.

### Notes

The value ranges from 0 to 100. 0 mutes the sound and 100 plays the sound at the "normal" volume set in the OS volume settings.

## Methods

### Clone As iOSSound

Creates a clone of the current sound that can be used independently of the original.

---

### Play

Plays the sound once.

---

### PlayLooping

Plays the sound, repeating it indefinitely.

---

### Stop

Stops the sound from playing.

# iOSSlider

A control with a draggable slider than can be used to change a value.

## Control Summary

Name	iOSSlider
Type	Control
Super	<a href="#">iOSControl</a>
Events	<a href="#">Close</a> <a href="#">Open</a> <a href="#">ValueChanged</a>
Properties	<a href="#">AccessibilityHint</a> <a href="#">AccessibilityLabel</a> <a href="#">Enabled</a> <a href="#">Height</a> <a href="#">Left</a> <a href="#">MaxValue</a> <a href="#">MinValue</a> <a href="#">Name</a> <a href="#">Parent</a> <a href="#">Top</a> <a href="#">Value</a> <a href="#">Visible</a> <a href="#">Width</a>
Methods	<a href="#">Handle</a> <a href="#">SetMinCurrentMax</a>
Project Types	Mobile
Platforms	iOS
Related	<a href="#">iOS Pickers</a>
UIKit	<a href="#">UISlider</a>

## Events

### ValueChanged

Called when CurrentValue is changed.

#### Sample Code

Update an [iOSLabel](#) with the value of the Slider as it changes:

```
MyTextField.Text = Me.Value.ToText</code>
```

## Properties

### Enabled As Boolean

Enables or disables the slider.

#### Sample Code

Disable a Slider:

```
MySlider.Enabled = False</code>
```

---

### MaxValue As Double

The maximum value of the slider. The default value is 100.

#### Sample Code

Change the maximum slider value:

```
MySlider.MaxValue = 50</code>
```

---

### MinValue As Double

The minimum value of the slider. The default value is 0.

#### Sample Code

Change the minimum slider value:

```
MySlider.MinValue = -50</code>
```

---

### Value As Double

The current value of the slider.

#### Notes

Set this value to move the slider position. If Value is greater than MaxValue, the MaxValue is used. If Value is less than MinValue, then MinValue is used.

#### Sample Code

Set the initial slider value:

```
MySlider.Value = 25</code>
```

## Methods

**SetMinCurrentMax(minValue As Double, currentValue As Double, maxValue As Double)**

For convenience, sets all the properties at one time.

### Sample Code

Sets values for the Slider:

```
MySlider.SetMinCurrentMax(-50, 25, 50)
```

# iOSSplitContent (Interface)

Indicates the iOS UI layouts that can be used in a [SplitView](#).

## Interface Summary

Name	iOSSplitContent
Type	Interface
Methods	
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSSplitView</a> , <a href="#">iOSTabBar</a> , <a href="#">iOSView</a>

# iOSSplitView

A SplitView allows you to present master/detail views on a single iPad screen.

## Control Summary

Name	iOSSplitView
Type	Control
Interfaces	<a href="#">iOSScreenContent</a> <a href="#">iOSViewController</a> <a href="#">iOSTabContent</a>
Properties	<a href="#">Detail</a> <a href="#">Master</a>
Methods	<a href="#">ViewControllerHandle</a>
Shared Methods	<a href="#">Available</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSView</a>
Example Projects	Examples/iOS/Controls/SplitView

## Properties

### Detail As iOSSplitContent

The detail (or right-hand side) view or tab bar to display.

### Master As iOSSplitContent

The master (or left-hand side) view or tab bar to display.

## Methods

### ViewControllerHandle As Ptr

Used when calling OS APIs.

## Shared Methods

### Available As Boolean

Indicates that a SplitView is available. This typically means that an iPad is being used.

# iOSSwitch

A Switch control is used to indicate an on/off, yes/no or true/false selection.

## Control Summary

Name	iOSSwitch
Type	Control
Super	<a href="#">iOSControl</a>
Events	<a href="#">Close</a> <a href="#">Open</a> <a href="#">ValueChanged</a>
Properties	<a href="#">AccessibilityHint</a> <a href="#">AccessibilityLabel</a> <a href="#">Enabled</a> <a href="#">Height</a> <a href="#">Left</a> <a href="#">Name</a> <a href="#">Parent</a> <a href="#">Top</a> <a href="#">Value</a> <a href="#">Visible</a> <a href="#">Width</a>
Methods	<a href="#">Handle</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOS Pickers</a>
UIKit	<a href="#">UISwitch</a>
Example Projects	Examples/iOS/Controls/SwitchExample

## Events

### ValueChanged

Called when the value property is changed either by code or by the user.

#### Sample Code

Disable an `iOSTextField` when the switch is off and enable it when on:

```
If Me.Value Then
```

```
MyTextField.Enabled = True
Else
    MyTextField.Enabled = False
End If</code>
```

## Properties

### Enabled As Boolean

Enables or disables the switch.

#### Sample Code

Disable a Switch:

```
MySwitch.Enabled = False
```

---

### Value As Boolean

The value of the switch. On = True, Off = False.

#### Sample Code

If the Tax switch is on, then set the tax rate:

```
Dim tax As Currency
If TaxSwitch.Value Then
    tax = 0.08
End If
```

# iOSTabBar

An iOS TabBar.

## Class Summary

Name	iOSTabBar
Type	Control
Interfaces	<a href="#">iOSScreenContent</a> <a href="#">iOSSplitContent</a> <a href="#">iOSViewController</a>
Methods	<a href="#">AddTab</a> <a href="#">ViewControllerHandle</a> <a href="#">ViewResized</a> <a href="#">ViewWillAppear</a> <a href="#">ViewWillDisappear</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSTabContent</a> <a href="#">iOSView</a>
Example Projects	iOS/Controls/TabBarExample/TabBarExample iOS/Framework/AdvancedThreadingExample iOS/Apps/EddiesElectronics/EEiOS

## Notes

You set up a tab bar using the Inspector to change the Content property for a [Screen](#). Refer to [Getting Started with iOS](#) for information on how to do this.

From the [Apple User Interface Guidelines](#), a tab bar:

- Is translucent
- Always appears at the bottom edge of the screen
- Displays no more than five tabs at one time in a horizontally compact environment (if there are more tabs, the tab bar displays four of them and adds the More tab, which reveals the additional tabs in a list)
- Maintains the same height in all orientations
- Can display a badge on a tab to communicate app-specific information (a badge is a red oval containing white text and either a number or exclamation point)

Regardless of the icon's visual style, create a tab bar icon in the following sizes: About 50 x 50 pixels (96 x 64 pixels maximum). Don't include text in a custom tab bar icon.

## Methods

### AddTab(content As iOSTabContent)

Adds an [iOSView](#) to the tab bar.

#### Sample Code

```
// This code (on a View) adds View3 to the Tab Bar
If Self.ParentTabBar <> Nil Then
    Dim v As New View3
    Self.ParentTabBar.AddTab(v)
End If
```

# iOSTabContent (Interface)

Identifies iOS controls that can be used on a [TabBar](#).

## Interface Summary

Name	iOSTabContent
Type	Interface
Methods	
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSTabBar</a> , <a href="#">iOSView</a> , <a href="#">iOSSplitView</a>

## Notes

You set up a tab bar on the Screens for the iOS devices. Refer to [Getting Started with iOS](#) for information on how to do this.

# iOSTable

An iOS Table used for displaying a list of data.

## Control Summary

Name	iOSTable
Type	Control
Super	<a href="#">iOSControl</a>
Enumerations	<a href="#">Formats</a>
Events	<a href="#">AccessoryAction</a> <a href="#">Action</a>
Properties	<a href="#">AccessibilityHint</a> <a href="#">AccessibilityLabel</a> <a href="#">DataSource</a> <a href="#">Format</a> <a href="#">Height</a> <a href="#">Left</a> <a href="#">Name</a> <a href="#">Parent</a> <a href="#">SectionCount</a> <a href="#">Top</a> <a href="#">Visible</a> <a href="#">Width</a>
Methods	<a href="#">AddRow</a> <a href="#">AddSection</a> <a href="#">Handle</a> <a href="#">InsertRow</a> <a href="#">InsertSection</a> <a href="#">ReloadData</a> <a href="#">ReloadDataInSection</a> <a href="#">ReloadRow</a> <a href="#">RemoveAll</a> <a href="#">RemoveRow</a> <a href="#">RemoveSection</a> <a href="#">RowCount</a> <a href="#">RowData</a> <a href="#">SectionTitle</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSTableCellData</a> <a href="#">iOSTableDataSource</a> <a href="#">User Guide: Table</a>

UIKit	<a href="#">UITableView</a>
Example Projects	iOS/Controls/Table/GroupTableExample iOS/Controls/Table/SectionTableExample iOS/Controls/Table/SimpleTableExample iOS/Controls/Table/TableCheckmark iOS/Controls/Table/TableViewCell iOS/Controls/Table/TableDetail iOS/Controls/Table/TableDisclosure iOS/Apps/EddiesElectronics/EEiOS iOS/Apps/XojoNotes/XojoNotes

## Notes

An `iOSTable` can only show a single column of data.

A table can have data added to it in one of two ways: manually or from a data source. Some methods only work with data that was added manually, some methods only work with data added using a data source.

A table must have at least one section (0-based). If there is only one section, then the section title does not appear in the table.

## Sample Code

```
' Simple usage
Table1.AddSection("")
For i As Integer = 0 To 4
    Table1.AddRow(0, "Row " + i.ToText)
Next
```

## Enumerations

### Formats

Indicates the formatting of the table.

Plain	A plain table consists of just the rows and section headings. This is the default.
Grouped	A grouped table has the section headings displayed outside of the table in visually distinct sections.

## Events

### AccessoryAction(section As Integer, row As Integer)

Called when the accessory (currently only [Detail](#)) for a row is tapped.

## Parameters

<i>section</i>	The section containing the row.
<i>row</i>	The row that was selected.

## Action(section As Integer, row As Integer)

Called when a row has been selected.

## Parameters

<i>section</i>	The section containing the row.
<i>row</i>	The row that was selected.

## Sample Code

Get the selected row:

```
Dim selectedCell As iOSTableCellData = Me.RowData(section, row)
```

## Properties

### DataSource As iOSTableDataSource

The data to display in the table.

### Notes

For data sets that are large, a data source allows you to efficiently provide rows without the memory cost of having an `iOSTableCell` in memory at all times for every single one of them.

`DataSource` is a weak reference so you need to make sure your data source instance does not go out of scope. The easiest way to ensure this is to make it a property of your `iOSView`.

### Format As Formats (design-only)

The format type for the table. This can only be specified at design-time in the Inspector. The default is `Formats.Plain`.

### SectionCount As Integer (read-only)

The count of sections in the table.

## Sample Code

Get the number of sections in the table:

```
Dim sections As Integer = Table1.SectionCount
```

## Methods

### AddRow(section As Integer, rowText As Text)

Adds the *rowText* to the *section*.

#### Notes

There has to be at least one section before you can add rows or you will get an `OutOfBoundsException`. If you do not want the section to appear, you can add it with a blank title.

#### Exceptions

<a href="#">OutOfBoundsException</a>	When the section does not exist. At least one section must be added to the table before adding rows.
<a href="#">UnsupportedOperationException</a>	When the table has a <code>DataSource</code> specified.

#### Sample Code

```
Table1.AddSection("") ' No section will appear
For i As Integer = 0 To 4
    Table1.AddRow(0, "Row " + i.ToText)
Next
```

### AddRow(section As Integer, data As [iOSTableCellData](#))

Adds the *data* to the *section*.

#### Notes

There has to be at least one section before you can add rows or you will get an `OutOfBoundsException`. If you do not want the section to appear, you can add it with a blank title.

#### Exceptions

<a href="#">OutOfBoundsException</a>	When the section does not exist. At least one section must be added to the table before adding rows.
<a href="#">UnsupportedOperationException</a>	When the table has a <code>DataSource</code> specified.

#### Sample Code

```
Table1.AddSection("") ' No section will appear
Dim cell As iOSTableCellData

For i As Integer = 0 To 4
```

```

    cell = New iOSTableCellData("Row " + i.ToText)
    Table1.AddRow(0, cell)
Next

```

---

## AddSection(title As Text = "")

### AddSection(title As Text = "") As Integer

Adds a section, with the optional *title*, to the table. Can optionally return the number for the new section.

#### Notes

For a Plain table format, section headings appear in the table itself. Only non-blank section titles appear. If you add a section with a blank title, you cannot change the title later.

For a Group table format, the sections appear outside of the table.

#### Exceptions

UnsupportedOperationException	When the table has a DataSource specified.
-------------------------------	--

#### Sample Code

Add a section called "Tasks":

```
Table1.AddSection("Tasks")
```

You may want to add a section and save the returned section number so you can use it to subsequently add rows:

```

Dim newSection As Integer = Table1.AddSection("Tasks")
Table1.AddRow(newSection, "Task 1")
Table1.AddRow(newSection, "Task 2")

```

---

## InsertRow(section As Integer, index As Integer)

Inserts a single row into the table without forcing a full table refresh from the DataSource.

#### Notes

When you insert the row, its value is fetched from the DataSource. This method only works when the table has a DataSource specified.

#### Exceptions

UnsupportedOperationException	When the table does not have a DataSource specified.
-------------------------------	--

## InsertRow(section As Integer, index As Integer, cell As UITableViewCellData)

Inserts cell data for the *section* at the row *index*.

---

## InsertRow(section As Integer, index As Integer, rowText As Text)

Inserts the *rowText* for the *section* at the row *index*.

### Sample Code

Insert a row in the first position of the Table:

```
Table1.InsertRow(0, 0, "First Row")
```

---

## InsertSection(index As Integer)

Inserts a section at the *index*.

### Notes

When you insert a section like this, its value is fetched from the DataSource. This method only works when the table has a DataSource specified.

---

## InsertSection(index As Integer, title As Text)

Inserts the section *title* at the specified *index*.

### Sample Code

Inserts a section at the top of the table:

```
Table1.InsertSection(0, "First!")
```

---

## ReloadData

Reloads all the table data from the DataSource.

### Notes

It is faster to only reload the specific data that need to be refreshed, so consider using `ReloadDataInSection` or `ReloadRow`.

---

## ReloadDataInSection(section As Integer)

Reloads the table data for the *section*.

### Notes

Use this method to reload a section after changes have been made to its data (either from a DataSource or by changing the RowData).

### Sample Code

Reload all the data in section from the related DataSource:

```
Table1.ReloadDataInSection(0)
```

---

## ReloadRow(section As Integer, row As Integer)

Reloads the table data for the *section* and *row*.

### Notes

Use this method to reload a single cell row after changes have been made to its data (either from a DataSource or by changing the RowData).

### Sample Code

In the Action event handler, reload the cell row after changing its AccessoryType:

```
Dim cell As iOSTableCellData = Me.RowData(section, row)
cell.AccessoryType = iOSTableCellData.AccessoryTypes.Checkmark
Me.ReloadRow(section, row)
```

---

## RemoveAll

Removes all rows from the table.

### Sample Code

Remove all rows:

```
Table1.RemoveAll</code>
```

---

## RemoveRow(section As Integer, row As Integer)

Removes the *row* from the *section*.

## Sample Code

```
Table1.RemoveRow(0, 4) // Removes row 4 from section 0
```

---

## RemoveSection(section As Integer)

Removes the *section* from the table (and all its rows).

### Sample Code

Remove section 0 and all the rows it contains:

```
Table1.RemoveSection(0)
```

---

## RowCount(section As Integer) As Integer

The count of rows in a *section*.

### Sample Code

Get the counts of rows in section 0:

```
Dim rowCount As Integer  
rowCount = Table1.RowCount(0)
```

---

## RowData(section As Integer, row As Integer) As iOSTableCellData

Returns the cell data for the *section* and *row*.

### Notes

You can use this method to get at the cell data to change its contents. After changing the contents, call `ReloadRow` to display the new information.

### Sample Code

Get all the cell data for the first row in the table and change its text:

```
Dim cell As iOSTableCellData  
cell = Table1.RowData(0, 0)  
cell.Text = "Changed"  
Table1.ReloadRow(0, 0)
```

---

## SectionTitle(section As Integer) As Text

### SectionTitle(section As Integer, Assigns value As Text)

Gets or sets the title for the section.

#### Notes

You cannot change the title section that was originally given a blank title.

#### Sample Code

Change the first section title:

```
Table1.SectionTitle(0) = "New Title"
```

# iOSTableCellData

A cell within an [iOSTable](#).

## Class Summary

Name	iOSTableCellData
Type	Class
Enumerations	<a href="#">AccessoryTypes</a>
Properties	<a href="#">AccessoryType</a> <a href="#">DetailText</a> <a href="#">Image</a> <a href="#">Tag</a> <a href="#">Text</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSTable</a> <a href="#">iOSTableDataSource</a> <a href="#">UserGuide: Table</a>
Example Projects	iOS/Controls/Table/TableCheckmark iOS/Controls/Table/TableDetail iOS/Controls/Table/TableDisclosure

## Enumerations

### AccessoryTypes

Additional UI controls for a cell.

None		A standard plain cell. This is the default for cells.
Disclosure		You use the disclosure indicator when selecting a cell results in the display of another table view reflecting the next level in the data model hierarchy. This does not track touches, so it does not call the <a href="#">iOSTable.AccessoryAction</a> event handler.
Detail		You use the detail disclosure button when selecting a cell results in a detail view of that item (which may or may not be a table view). This tracks touches and touching it calls the <a href="#">iOSTable.AccessoryAction</a> event handler.
Checkmark		You use a checkmark when a touch on a row results in the selection of that item. This kind of table view is known as a selection list, and it is analogous to a pop-up list. Selection lists can limit selections to one row, or they can allow multiple rows with checkmarks. This does not track touches, so it does not call the <a href="#">iOSTable.AccessoryAction</a> event handler.

## Constructors

Constructor(text As Text, detail As Text = "", image As UIImage = nil, accessory As AccessoryTypes = AccessoryTypes.None)

Creates a new cell data using the *text*, *detail*, *image*, *accessory* values.

## Properties

AccessoryType As AccessoryTypes

The accessory type that is displayed in the cell.

### Sample Code

Create a new cell with a checkmark accessory:

```
Dim cell As New iOSTableCellData
cell.Text = "Sample"
cell.AccessoryType = iOSTableCellData.AccessoryTypes.Checkmark
```

---

DetailText As Text

The the smaller text that appears below the main Text in the cell.

### Sample Code

Set the detail text:

```
Dim cell As New iOSTableCellData
cell.Text = "Acme Widget"
cell.DetailText = "Price: $3.00"
```

---

Image As UIImage

The image for the cell.

---

Tag As Auto

A tag value for the cell.

---

Text As Text

The text of the cell. This is the main (large) text that appears for the row.

## Sample Code

Set the text:

```
Dim cell As New iOSTableCellData  
cell.Text = "Acme Widget"
```

# iOSTableDataSource

A data source to use to populate a [Table](#).

## Interface Summary

Name	iOSTableDataSource
Type	Interface
Methods	<a href="#">RowCount</a> <a href="#">RowData</a> <a href="#">SectionCount</a> <a href="#">SectionTitle</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSTable</a> <a href="#">iOSTableCellData</a> <a href="#">User Guide: Table</a>

## Notes

To use the interface, create a class that implements the interface to manage the data. The data could be in a database, dictionary, array or anything else you want. Your class implements the methods below to return the appropriate values based on the data it is managing.

## Methods

### RowCount(section As Integer) As Integer

The number of rows for the *section*.

---

### RowData(section As Integer, row As Integer) As [iOSTableCellData](#)

The actual cell data for the *section* and *row*.

---

### SectionCount As Integer

The number of sections.

---

## SectionTitle(section As Integer) As Text

The title for the *section*.

# iOSTextArea

A multiline text area for displaying text.

## Control Summary

Name	iOSTextArea
Type	Control
Super	<a href="#">iOSControl</a>
Events	<a href="#">Close</a> <a href="#">Open</a> <a href="#">SelChange</a> <a href="#">TextChange</a>
Properties	<a href="#">AccessibilityHint</a> <a href="#">AccessibilityLabel</a> <a href="#">Editable</a> <a href="#">Height</a> <a href="#">Left</a> <a href="#">Name</a> <a href="#">Parent</a> <a href="#">Text</a> <a href="#">TextAlignment</a> <a href="#">TextColor</a> <a href="#">TextFont</a> <a href="#">Top</a> <a href="#">Visible</a> <a href="#">Width</a>
Methods	<a href="#">Handle</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSTextField</a> <a href="#">iOS Inputs</a>

## Events

### SelChange

Called when the text selection changes.

### TextChange

Called when the text changes.

## Sample Code

Update a label with the text as it changes:

```
DescLabel.Text = Me.Text
```

## Properties

### Editable As Boolean

Indicates if the text is editable or read-only.

#### Notes

If the text area is not editable, the user can still select and copy the text within it.

## Sample Code

Make the text field read-only:

```
DescArea.Editable = False
```

---

## Text As Text

The text of the text area.

## Sample Code

Set the text of the text area:

```
TextArea1.Text = "Hello, World!"</code>
```

Get the text in the text area:

```
Dim desc As Text  
desc = TextArea1.Text</code>
```

---

## TextAlignment As iOSTextAlignment

The alignment of the text.

## Sample Code

Center-align the text:

```
TextArea1.TextAlignment = iOSTextAlignment.Center
```

## TextColor As Color

The color of the text.

### Sample Code

Make the text red:

```
TextArea1.TextColor = Color.Red
```

---

## TextFont As iOSFont

The font used to display the text.

### Sample Code

Use the bold system font:

```
TextArea1.TextFont = iOSFont.BoldSystemFont
```

# iOSTextField



A single-line text field for text.

## Control Summary

Name	iOSTextField
Type	Control
Super	<a href="#">iOSControl</a>
Events	<a href="#">Close</a> <a href="#">Open</a> <a href="#">TextChange</a>
Properties	<a href="#">AccessibilityHint</a> <a href="#">AccessibilityLabel</a> <a href="#">Enabled</a> <a href="#">Height</a> <a href="#">KeyboardType</a> <a href="#">Left</a> <a href="#">Name</a> <a href="#">Parent</a> <a href="#">Password</a> <a href="#">Placeholder</a> <a href="#">Text</a> <a href="#">TextAlignment</a> <a href="#">TextColor</a> <a href="#">TextFont</a> <a href="#">Top</a> <a href="#">Visible</a> <a href="#">Width</a>
Methods	<a href="#">Handle</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSTextArea</a> <a href="#">iOS Inputs</a>
UIKit	<a href="#">UITextField</a>

## Events

### TextChange

Called when the text changes from user entry. It is not called when the text is changed via code.

## Sample Code

Update a label with the text as it is being entered:

```
NameLabel.Text = Me.Text
```

## Properties

### Enabled As Boolean

Indicates if the field is enabled or disabled.

#### Sample Code

Disable the text field

```
TextField1.Enabled = False</code>
```

---

### KeyboardType As [iOSKeyboardTypes](#)

Specifies the type of keyboard to use with the text field. The various types are specified using the [iOSKeyboardTypes](#) enumeration.

#### Sample Code

Use the number pad for the text field:

```
Me.KeyboardType = iOSKeyboardTypes.NumberPad
```

---

### Password As Boolean

Indicates if this is a password, which means the text is obscured as it is entered.

#### Sample Code

Activate password entry mode for the text field:

```
PasswordField.Password = True
```

---

### Placeholder As Text

The placeholder text that appears when the field is empty.

#### Notes

Placeholders are not the actual text so they do not need to be cleared before typing. Placeholders are useful for

smaller screens because they can replace having a separate label to describe what to enter in the field.

### Sample Code

Set the placeholder text:

```
NameField.PlaceHolder = "Enter your name:"
```

---

## Text As Text

The text in the field.

### Sample Code

Sets the text in the field:

```
TextField1.Text = "Hello, World!"</code>
```

Gets the text from the field:

```
Dim desc As Text  
desc = TextField1.Text</code>
```

---

## TextAlignment As iOSTextAlignment

The alignment for the text.

### Sample Code

Center-align the text:

```
TextField1.TextAlignment = iOSTextAlignment.Center
```

---

## TextColor As Color

The color of the text.

### Sample Code

Change the text color to red:

```
TextField1.TextColor = Color.Red
```

---

## UIFont As UIFont

The font used to display the text.

### Sample Code

Use the bold system font:

```
TextField1.Font = UIFont.BoldSystemFont
```

# iOSToolbar

A toolbar can be displayed in a view as a [Toolbar](#) or [Navigation Bar](#).

## Class Summary

Name	iOSToolbar
Type	Class
Methods	<a href="#">Add</a> <a href="#">Count</a> <a href="#">Insert</a> <a href="#">RemoveAll</a> <a href="#">RemoveAllByValue</a> <a href="#">RemoveByIndex</a> <a href="#">RemoveByValue</a> <a href="#">Value</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSToolButton</a> <a href="#">iOSView</a> <a href="#">iOS Toolbars</a>
Example Projects	Examples/iOS/Controls/ToolbarExample

## Notes

Accessed using the [LeftNavigationToolbar](#), [RightNavigationToolbar](#) and [Toolbar](#) properties of the [iOSView](#).

## Methods

### Add(value As [iOSToolButton](#))

Adds the button to the end of the toolbar.

#### Sample Code

Add the SystemAdd button to the right Navigation Bar:

```
RightNavigationToolbar.Add(iOSToolButton.NewSystemItem(iOSToolButton.Types.SystemAdd))</code>
```

## Count As Integer

The number of buttons on the toolbar. If Count = 0 then no toolbar is displayed.

### Sample Code

Get the count of buttons on an iOSView:

```
Dim buttonCount As Integer = Toolbar.Count</code>
```

---

## Insert(index As Integer, value As iOSToolButton)

Inserts the BarButton at the specified position on the bar.

### Sample Code

Insert a button in the first position of the toolbar:

```
Toolbar.Insert(0, iOSToolButton.NewPlain("Save"))</code>
```

---

## RemoveAll

Removes all the BarButtons from the bar.

### Sample Code

Remove all the buttons from the toolbar:

```
Toolbar.RemoveAll</code>
```

---

## RemoveAllByValue(value As iOSToolButton)

Removes all of a specific button from the toolbar.

## RemoveByIndex(index As Integer)

Removes the button at the specified index from the toolbar.

### Sample Code

Remove the first button from the toolbar:

```
Toolbar.RemoveByIndex(0)</code>
```

---

## RemoveByValue(value As iOSToolButton)

Removes a specific button from the toolbar.

### Sample Code

Remove the first button from the toolbar:

```
Dim removeButton As iOSToolButton
removeButton = Toolbar.Value(0)
Toolbar.RemoveByValue(removeButton)
```

---

## Value(index As Integer) As iOSToolButton

### Value(index As Integer, Assigns newValue As iOSToolButton)

References the button at the specified index on the toolbar.

### Sample Code

Get the first button on the toolbar:

```
Dim button As iOSToolButton
button = Toolbar.Value(0)
```

Change the Caption of the last button on the toolbar:

```
Toolbar.Value(Toolbar.Count - 1).Caption = "New"
```

# iOSToolButton

A button to add to a Toolbar or Navigation Bar on a View.

## Class Summary

Name	iOSToolButton
Type	Control
Enumerations	<a href="#">Types</a>
Properties	<a href="#">Caption</a> <a href="#">Enabled</a> <a href="#">Image</a> <a href="#">Tag</a> <a href="#">Type</a> <a href="#">Width</a>
Method	<a href="#">Handle</a>
Shared Methods	<a href="#">FixedSpace</a> <a href="#">FlexibleSpace</a> <a href="#">NewBordered</a> <a href="#">NewDone</a> <a href="#">NewFromHandle</a> <a href="#">NewPlain</a> <a href="#">NewSystemItem</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSToolbar</a> <a href="#">iOSView</a> iOS Toolbars

## Enumerations

### Types

System icons to use when creating buttons using [NewSystemItem](#).

SystemAction	
SystemAdd	
SystemBookmarks	

SystemCamera	
SystemCancel	"Cancel"
SystemCompose	
SystemDone	"Done"
SystemEdit	"Edit"
SystemFastForward	
SystemFixedSpace	n/a
SystemFlexibleSpace	n/a
SystemOrganize	
SystemPageCurl	n/a
SystemPause	
SystemPlay	
SystemRedo	"Redo"
SystemRefresh	
SystemReply	
SystemRewind	
SystemSave	"Save"
SystemSearch	
SystemStop	
SystemTrash	
SystemUndo	"Undo"

## Constructors

Constructor(type As Types, title As Text = "", image As UIImage = Nil)

Creates a button with the specified type, title or image.

### Sample Code

Adds the SystemRefresh icon as a toolbar button:

```
Dim tb As New iOSToolbutton(iOSToolbutton.Types.SystemRefresh)
Self.Toolbar.Add(tb)</code>
```

## Properties

### Caption As Text

Sets the caption for the button. If you have both a Caption and Image specified, then only the Image appears.

---

### Enabled As Boolean

Enables or disables the button.

#### Sample Code

Add a disabled "Save" button to the toolbar:

```
Dim saveButton As iOSToolbutton
saveButton = iOSToolButton.NewSystemItem(iOSToolbutton.Types.SystemSave)
saveButton.Enabled = False
Self.Toolbar.Add(saveButton)</code>
```

---

### Image As [UIImage](#)

Specifies an image to display for the button. If you have both a Caption and Image specified, then only the Image appears.

#### Notes

Only the mask for the image is used, displayed using the tint color for iOS (currently blue). Also refer to these topics in the Apple iOS Human Interface Guidelines:

- [Icon Sizes](#)
  - [Bar Button Icons](#)
- 

### Tag As [Auto](#)

A tag value.

---

### Type As [Types](#)

The button type.

---

## Width As Double

The button width.

## Methods

### Handle As Ptr

Returns the iOS handle for use with Declares.

## Shared Methods

### FixedSpace As iOSToolButton

Creates a fixed space button.

Returns

A new iOSToolButton.

#### Sample Code

Add a fixed space to separate buttons on a toolbar:

```
Self.Toolbar.Add(iOSToolButton.NewPlain("Save"))
Self.Toolbar.Add(iOSToolButton.FixedSpace)
Self.Toolbar.Add(iOSToolButton.NewPlain("Edit"))</code>
```

---

### FlexibleSpace As iOSToolButton

Creates a flexible space button.

Returns

A new iOSToolButton.

#### Sample Code

Add a flexible space to separate buttons:

```
Self.Toolbar.Add(iOSToolButton.NewPlain("Save"))
Self.Toolbar.Add(iOSToolButton.FlexibleSpace)
Self.Toolbar.Add(iOSToolButton.NewPlain("Reports"))</code>
```

## NewBordered(title As Text) As iOSToolButton

## NewBordered(image As [UIImage](#)) As iOSToolButton

Creates a new bordered button with the specified title or image.

### Parameters

<i>title</i>	The text value to display in the button.
<i>image</i>	An Image to display in the button.

### Returns

A new iOSToolButton.

### Notes

On current iOS versions, this button looks the same as a button created with [NewPlain](#).

Refer to the [Image](#) property for image suggestions.

---

## NewDone(title As Text) As iOSToolButton

## NewDone(image As [UIImage](#)) As iOSToolButton

Creates a new "Done" button with the specified title or image.

### Parameters

<i>title</i>	The text value to display in the button.
<i>image</i>	An Image to display in the button.

### Returns

A new iOSToolButton.

### Notes

This button has a bold appearance.

Refer to the [Image](#) property for image suggestions.

### Sample Code

Adds a "Done" button the to Navigation Bar:

```
Dim b As iOSToolButton
b = iOSToolButton.NewDone("Done")
Self.RightNavigationBar.Add(b)</code>
```

---

## NewFromHandle(handle As Ptr) As iOSToolButton

Creates a button from an iOS handle.

### Notes

Used when creating buttons from Declares to Cocoa Touch.

---

## NewPlain(title As Text) As iOSToolButton

## NewPlain(image As UIImage) As iOSToolButton

Creates a simple plain button with the specified title or image.

### Parameters

<i>title</i>	The text value to display in the button.
<i>image</i>	An Image to display in the button.

### Notes

Refer to the [Image](#) property for image suggestions.

### Sample Code

Add a simple button:

```
Dim b As iOSToolButton
b = iOSToolButton.NewPlain("Invoices")
Self.Toolbar.Add(b) ' Add to toolbar on view</code>
```

Add a simple button with an image:

```
Dim b As iOSToolButton
b = iOSToolButton.NewPlain(ButtonImage) ' ButtonImage is an image in the project
Self.Toolbar.Add(b) ' Add to toolbar on view
```

---

## NewSystemItem(type As Types) As iOSToolButton

Creates a button using a built-in system item icon.

### Sample Code

This code in the Open event handler of a view places an Add button in the left Navigation Bar:

```
' Create the button
```

```
Dim button As iOSToolButton
button = iOSToolButton.NewSystemItem(iOSToolButton.Types.SystemAdd)

' Add it to the view
LeftNavigationBar.Add(button)
```

# iOSUserControl (Control)

Used to embed UIViews created via declares into the Xojo control hierarchy.

## Control Summary

Name	iOSUserControl
Type	Class
Super	<a href="#">iOSControl</a>
Events	<a href="#">CreateView</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">Declare</a>
Example Projects	Examples/iOS/Declares/UIDatePicker

## Notes

Each iOSControl has an underlying UIView. The iOSUserControl class lets you hand the framework explicitly what you want it to use for the UIView.

In the CreateView event, you should create and configure whatever UIView you want. If the UIView's initializer requires a frame rectangle, pass in anything arbitrary (it will be placed correctly via auto layout). The UIView should be returned autoreleased and not just the result of alloc+init.



You cannot place an iOSUserControl (from the Library) directly on a layout. Instead you must subclass it, add code to the CreateView event handler and then drag the subclass onto the layout.

## Events

### CreateView As Integer

Use this event to create the UIView to use for the control. Return the pointer to the UIView.

# iOSView

The View is where you design your iOS layouts.

## Class Summary

Name	iOSView
Type	Control
Interfaces	<a href="#">iOSScreenContent</a> <a href="#">iOSSplitContent</a> <a href="#">iOSTabContent</a> <a href="#">iOSViewController</a>
Events	<a href="#">Activate</a> <a href="#">Close</a> <a href="#">Deactivate</a> <a href="#">Open</a> <a href="#">Resized</a> <a href="#">ToolbarPressed</a>
Properties	<a href="#">BackButtonTitle</a> <a href="#">BottomLayoutGuide</a> <a href="#">LeftNavigationToolbar</a> <a href="#">NavigationBarVisible</a> <a href="#">ParentSplitView</a> <a href="#">ParentTabBar</a> <a href="#">RightNavigationToolbar</a> <a href="#">TabIcon</a> <a href="#">TabTitle</a> <a href="#">Title</a> <a href="#">Toolbar</a> <a href="#">TopLayoutGuide</a>
Methods	<a href="#">AddConstraint</a> <a href="#">AddControl</a> <a href="#">Close</a> <a href="#">Constraint</a> <a href="#">ContentSize</a> <a href="#">Control</a> <a href="#">ControlCount</a> <a href="#">Handle</a> <a href="#">PushTo</a> <a href="#">RemoveConstraint</a> <a href="#">RemoveControl</a> <a href="#">Size</a> <a href="#">ViewControllerHandle</a>
Project Types	iOS
Platforms	iOS

Related	<a href="#">iOSLayoutConstraint</a> <a href="#">iOSSplitView</a> <a href="#">iOSTabBar</a> <a href="#">iOSToolbar</a> <a href="#">iOSViewController</a>
Example Projects	iOS/Controls/MultipleViews iOS/Controls/SplitViewExample

## Events

### Activate

Called when the view is activated.

#### Notes

This occurs when:

- the view first opens
- the view is displayed again after a view that was pushed onto it is closed

---

### Close

Called when the View is closed by calling the [Close](#) method.

#### Notes

This is called by the Destructor.

---

### Deactivate

Called when the view is deactivated. This occurs when:

- the view is closed
- another view is opened (pushed) onto this view

---

### Open

Called when the view first appears and typically used for initialization.

#### Notes

This event is called by the Constructor.

---

### Resized

Called when the view is resized, which can occur when the device orientation changes.

---

## ToolBarPressed(button As [iOSToolButton](#))

Called when a button on the left or right navigation bar or the toolbar is pressed.

### Sample Code

Check for a button pressed base on its caption:

```
Select Case button.Caption
  Case "Save"
    ' call save code
  Case "Edit"
    ' call edit code
End Select</code>
```

Check for a button pressed base on its type:

```
Select Case button.Type
Case iOSToolButton.Types.SystemSave
  ' call save code
Case iOSToolButton.Types.SystemEdit
  ' call edit code
End Select</code>
```

## Properties

### BackButtonTitle As [Text](#)

The back button that gets back to this view uses the text specified here.

### Notes

This title does not appear on the back button for this view, but instead appears on the back button for any views shown by this view (using [PushTo](#)). It is the title for the back button that goes back to *this* view.

### Sample Code

```
Self.BackButtonTitle = "Customers"
```

---

## BottomLayoutGuide As Object (read-only)

The bottom layout guide for the view. For use when creating [iOSLayoutConstraints](#).

### Notes

For a View, [iOSLayoutConstraint.AttributeTypes.Bottom](#) is the absolute bottom of the View. BottomLayoutGuide is adjusted for things such as a Toolbar or TabBar.

## LeftNavigationBar As iOSToolbar

The toolbar to display at the left of the navigation bar. Only visible when `NavigationBarVisible = True`.

### Sample Code

Add a button to the left Navigation Bar:

```
LeftNavigationBar.Add(iOSToolButton.NewSystemItem(iOSToolButton.Types.SystemAdd))
```

## NavigationBarVisible As Boolean

Indicates whether the navigation bar is visible. The navigation bar must be visible in order to see the [Title](#), [LeftNavigationBar](#) or [RightNavigationBar](#).

### Sample Code

Display the Navigation Bar:

```
Self.NavigationBarVisible = True
```

## ParentSplitView As iOSSplitView

Indicates the split view that is the owner of this view. It is Nil if there is no split view. A split view can only be used on iPad devices.

### Notes

Use this property to determine if a `SplitView` is displayed. You can then use it to get access to the Master and Detail views that are displayed.

### Sample Code

If a `SplitView` is used, then populate the detail side, otherwise, push a new view onto the screen:

```
If Self.ParentSplitView <> Nil Then
    ' On Action event for a Table on a Master view of the SplitView.
    ' Gets the Text for the selected row and
    ' assigns it to a Label on the DetailView of the SplitView.
    DetailView(Self.ParentSplitView.Detail).Label1.Text = Me.RowData(section, row).Text
Else
    ' No SplitView, so this is a phone.
    ' Display the Detail view and update the text for its label.
    Dim d As New DetailView
    d.Label1.Text = Me.RowData(section, row).Text
    Self.PushTo(d)
```

End If

---

## ParentTabBar As iOSTabBar

Indicates the tab bar that is the owner of this view. It is Nil if there is no tab bar.

### Sample Code

Add a new View to the Tab Bar:

```
' This code (on a View) adds View3 to the Tab Bar
If Self.ParentTabBar <> Nil Then
    Dim v As New View3
    Self.ParentTabBar.AddTab(v)
End If
```

---

## RightNavigationToolbar As iOSToolbar

The toolbar to display at the right of the navigation bar. Only visible when NavigationBarVisible = True.

### Sample Code

Add a button to the right Navigation Toolbar:

```
RightNavigationToolbar.Add(iOSToolButton.NewSystemItem(iOSToolButton.Types.SystemAdd))
```

---

## TabIcon As UIImage

The icon for the tab that this view is displayed on.

---

## TabTitle As Text

The title for the tab that this view is displayed on.

---

## Title As Text

The title for the Navigation Bar. This only appears if NavigationBarVisible is True.

### Sample Code

Change the title for the View:

```
Self.Title = "My View"
```

---

## Toolbar As [iOSToolbar](#)

The toolbar that is displayed in the view, typically at the bottom.

### Sample Code

Add a button to the Toolbar:

```
Toolbar.Add(iOSToolButton.NewSystemItem(iOSToolButton.Types.SystemAdd))
```

---

## TopLayoutGuide As Object (read-only)

The top layout guide for the view. For use when creating [iOSLayoutConstraints](#).

### Notes

For a View, [iOSLayoutConstraint.AttributeTypes.Top](#) is the absolute top of the View. TopLayoutGuide is adjusted for things such as a NavigationBar.

## Methods

### AddConstraint(constraint As [iOSLayoutConstraint](#))

Adds the *constraint* to the view.

---

### AddControl(child As [iOSControl](#))

Adds a control to the view.

### Sample Code

Add a control to the View:

```
Dim ctrl As New iOSSwitch
' Send its ValueChanged event to the SwitchValueChanged method on the View
AddHandler ctrl.ValueChanged, AddressOf SwitchValueChanged
Self.AddControl(ctrl)
```

---

## Close

For views that were displayed using PushTo, this closes the view causing the previous view to display.

### Notes

You cannot close the main View (the one that is specified as the Content for the Screen).

---

## Constraint(name As Text) As UILayoutConstraint

Gets a reference to a named constraint so that you can modify its settings in code.

### Sample Code

Change an existing (and named) constraint of a control on the View:

```
' "TAWidth" is a width constraint for a TextField that has been given  
' a name in the auto-layout Inspector properties.  
Dim c As UILayoutConstraint = Self.Constraint("TAWidth")  
c.Addend = 200
```

---

## ContentSize As Size

The size (in points) of the content area of the View. This excludes the Navigation Bar, Tab Bar and Toolbar if they are on the View.

### Sample Code

Get the content size of the View:

```
Dim screenSize As Text = Self.ContentSize.Width.ToText + " by " +  
Self.ContentSize.Height.ToText</code>
```

---

## Control(index As Integer) As IOSControl

Gets the control at the 0-based *index*.

### Sample Code

Get the name of the first control on the view:

```
Dim controlName As Text = Self.Control(0).Name</code>
```

---

## ControlCount As Integer

The number of controls on the view.

### Sample Code

Get the number of controls on the View:

```
Dim count As Integer = Self.ControlCount</code>
```

---

## Handle As Ptr

The handle to use when creating `Declares` to `UIView`.

---

## PushTo(newView As iOSView)

Displays a new view by "pushing" it onto the current view.

### Sample Code

Displays a new View on the screen:

```
Dim newView As New View2
Self.PushTo(newView)</code>
```

---

## RemoveConstraint(constraint As iOSLayoutConstraint)

Removes the *constraint* from the view.

---

## RemoveControl(child As iOSControl)

Removes the control from the view.

---

## Size As Size

The size (in points) of the entire View area.

### Sample Code

Get the size of the View:

```
Dim screenSize As Text = Self.Size.Width.ToText + " by " +
Self.Size.Height.ToText</code>
```

---

## ViewControllerHandle As Ptr

The handle to use when creating `Declares` to `UIView`.

---

# UIViewController

Identifies controls that can be on a screen.

## Interface Summary

Name	UIViewController
Type	Interface
Methods	<a href="#">ViewControllerHandle</a> , <a href="#">ViewResized</a> , <a href="#">ViewWillAppear</a> , <a href="#">ViewWillDisappear</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">iOSSplitView</a> , <a href="#">iOSTabBar</a> , <a href="#">iOSView</a>

## Methods

### ViewControllerHandle As Ptr

The handle to use for OS API calls.

---

### ViewResized

TBD

---

### ViewWillAppear

TBD

---

### ViewWillDisappear

TBD

# SQLiteDatabase (Class)

Provides access to SQLite databases.

## Class Summary

Name	SQLiteDatabase
Type	Class
Properties	<a href="#">DatabaseFile</a> <a href="#">EncryptionKey</a> <a href="#">LibraryVersion</a> <a href="#">ThreadYieldInterval</a> <a href="#">Timeout</a>
Methods	<a href="#">AttachDatabase</a> <a href="#">Connect</a> <a href="#">CreateDatabaseFile</a> <a href="#">Decrypt</a> <a href="#">DetachDatabase</a> <a href="#">Encrypt</a> <a href="#">LastRowID</a> <a href="#">SQLExecute</a> <a href="#">SQLSelect</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">SQLiteDatabaseField</a> <a href="#">SQLiteRecordSet</a> <a href="#">SQLiteException</a>
Example Projects	<a href="#">SQLiteInMemory</a> <a href="#">SQLiteVersion</a>

## Notes

SQLiteDatabase uses SQLite 3.8.5. For more information about SQLite:

- [SQL Syntax](#)
- [Pragma Commands](#)
- [Unsupported SQL](#)

If your database object goes out of scope, the database is closed. This means you are typically going to want your database object be somewhat global to your app. A common technique is to have a SQLiteDatabase property on the App object that you refer throughout your app as **App.DB**.

### Transactions

SQLiteDatabase does an auto-commit after each SQL command if you do not manually start a transaction. You can

start a transaction using this command:

```
myDB.SQLExecute("BEGIN TRANSACTION")
```

When the transaction is complete, you commit your changes with this command:

```
myDB.SQLExecute("COMMIT")</code>
```

You can cancel a transaction by calling rollback with this command:

```
myDB.SQLExecute("ROLLBACK")</code>
```

## Properties

### DatabaseFile As FolderItem

Specifies the FolderItem for the SQLite database file. If DatabaseFile is Nil, calling the Connect method creates an in-memory database.

---

### EncryptionKey As Text

Specifies the encryption key used to open an encrypted database or to encrypt a database.

#### Notes

To encrypt a new database, specify this value before calling CreateDatabaseFile.

To connect to an encrypted database, specify this value before calling Connect.

To encrypt an existing database, use the Encrypt method after you have created or connected to the database.

---

### LibraryVersion As Text (read-only)

The version of the SQLite library.

#### Sample Code

Display the SQLite version:

```
Dim db As New SQLiteDatabase  
VersionLabel.Text = db.LibraryVersion
```

---

### ThreadYieldInterval As Integer

Yields time back to your application every N virtual machine instructions. The unit is virtual machine instructions.

## Notes

This is provided for advanced usage of SQLite.

---

## Timeout As Double

The maximum number of seconds that an operation may wait before a lock is cleared (if any).

## Notes

A SQLiteDatabase can be locked while it is being modified. On iOS, you will not need to worry about the timeout unless you have multiple threads accessing a database using separate connections.

## Methods

### AttachDatabase(file As FolderItem, databaseName As Text, encryptionKey As Text = "") As Boolean

Attaches the SQLite database referred to by *file* to the database. It gives the newly attached database the name *databaseName*. If the database is encrypted, be sure to specify the *encryptionKey*.

## Notes

You can attach one or more databases to a currently connected SQLiteDatabase so that you can work with all the databases at one time. This can be useful to copy data between databases, for example.

If tables in the attached database have the same name as tables in the primary database, you can refer to them using *databaseName* as the prefix.

---

## Connect As Boolean

Connects to the database so that you can begin using it. Before proceeding with database operations, test to be sure that Connect returns True.

## Notes

To create an in-memory SQLite database, call Connect without specifying a DatabaseFile.

## Sample Code

Connect to an existing database:

```
Dim db As New SQLiteDatabase

Dim dbFile As FolderItem
dbFile = SpecialFolder.Documents.Child("MyDB.sqlite")

If dbFile.Exists Then
    db.DatabaseFile = dbFile
```

```
If db.Connect Then
    //proceed with database operations here...
Else
    Dim err As Text = "Could not open the database."
End If
End If
```

---

## CreateDatabaseFile As Boolean

Creates a database file. Returns True if the database was successfully created.

### Notes

You are automatically connected if CreateDatabase returns True, so you do not need to also call Connect.

### Sample Code

Create a new SQLiteDatabase:

```
Dim dbFile As FolderItem
dbFile = SpecialFolder.Documents.Child("MyDB.sqlite")

Dim db As New SQLiteDatabase
db.DatabaseFile = dbFile
If db.CreateDatabaseFile Then
    // proceed with database operations here...
Else
    Dim err As Text = "Database could not be created."
End If
```

---

## Decrypt

Removes the encryption from an encrypted database. You must be connected to the database in order to decrypt it.

### Notes

Decrypting a database removes the encryption. You do not need to decrypt the database for normal use. Only use this command if you want to actually remove the encryption so that you can later connect to the database without providing the encryption key.

---

## DetachDatabase(databaseName As Text)

Detaches the passed database that was previously attached with AttachDatabase.

---

## Encrypt(encryptionKey As Text)

Encrypts the database using password as the encryption key. If you are already connected to an encrypted database and call Encrypt with an empty string, the database is decrypted.

## LastRowID As Int64

Returns an Int64 containing the value of the last RowID added to any table in the database.

## SQLExecute(sqlstatement As Text, ParamArray values() As Auto)

Used to execute a SQL command. Use this for commands that do not return any data, such as CREATE TABLE or INSERT. You can optionally supply a list of values that will replace parameters (specified with "?") in *sqlstatement*.

### Sample Code

Creates a table in a SQLiteDatabase:

```
Dim sql As Text

sql = "CREATE TABLE Team (ID INTEGER, Name TEXT," + _
" Coach TEXT, City TEXT, PRIMARY KEY(ID));"

Try
    DB.SQLExecute(sql) ' DB is a previously created SQLiteDatabase
Catch e As SQLiteException
    Dim err As Text = e.Reason
End Try
```

Update data in a table:

```
Dim sql As Text
sql = "UPDATE Team SET Name = ?1, Coach = ?2 WHERE ID = ?3"

' Pass in values after sql instead of doing string replacement
Try
    Dim name As Text = "Mud Hens"
    Dim coach As Text = "Dave Roberts"
    Dim ID As Integer = 100 ' Primary key for row
    DB.SQLExecute(sql, firstName, lastName, ID)
Catch e As SQLiteException
    Dim err As Text = e.Reason
End Try
```

Add data to a table:

```
Dim sql As Text
```

```
sql = "INSERT INTO Team (Name, Coach, City) VALUES (?1, ?2, ?3)"
```

```
' Pass in values after sql instead of doing string replacement
```

```
Try
  Dim name As Text = "Flying Squirrels"
  Dim coach As Text = "Tim Roberts"
  Dim city As Text = "Springfield"
  ' ID is created automatically because it is a primary key
  DB.SQLiteExecute(sql, name, coach, city)
Catch e As SQLiteException
  Dim err As Text = e.Reason
End Try
```

Remove data from a table:

```
Dim sql As Text
sql = "DELETE FROM Team WHERE ID = ?!"

' Pass in values after sql instead of doing string replacement
Try
  Dim ID As Integer = 100 ' Primary key for row
  DB.SQLiteExecute(sql, ID) ' Previously connected database
Catch e As SQLiteException
  Dim err As Text = e.Reason
End Try
```

---

## SQLSelect(sqlstatement As Text, ParamArray values() As Auto) As SQLiteRecordSet

Used to run an SQL statement that returns a `SQLiteRecordSet`, such as `SELECT` statement. You can optionally supply a list of values that will replace parameters (specified with "?") in *sqlstatement*.

### Sample Code

Get the Name and City values for the Team table:

```
Dim sql As Text = "SELECT Name, City FROM Team"

Dim teams As SQLiteRecordSet
Try
  teams = DB.SQLiteSelect(sql) ' DB is a previously created SQLiteDatabase
Catch e As SQLiteException
  Dim err As Text = e.Reason
End Try
```

# SQLiteDatabase Example Projects

Location	Examples/iOS/Database
----------	-----------------------

## SQLiteInMemory

This example demonstrates these features:

- Create an in-memory SQLite database
- Add a table to the database using [SQLiteDatabase.SQLExecute](#)
- Add data to the table using [SQLiteDatabase.SQLExecute](#)
- Display the data in an `iOSTable` using [SQLiteDatabase.SQLSelect](#)

Refer to the `CreateTableWithData` and `LoadData` methods.

## SQLiteVersion

Uses [SQLiteDatabase.LibraryVersion](#) to display the version of SQLite being used.

# SQLiteDatabaseField (Class)

Used to access the values of fields in a row of a SQLite database table.

## Item Summary

Name	SQLiteDatabaseField
Type	Class
Properties	<a href="#">BooleanValue</a> <a href="#">CurrencyValue</a> <a href="#">DateValue</a> <a href="#">DoubleValue</a> <a href="#">Int64Value</a> <a href="#">IntegerValue</a> <a href="#">Name</a> <a href="#">NativeValue</a> <a href="#">TextValue</a> <a href="#">Value</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">SQLiteDatabase</a> <a href="#">SQLiteRecordSet</a> <a href="#">SQLiteException</a>

## Notes

You cannot modify the data in SQLiteDatabaseField. If you need to modify the data in your SQLite database you'll need to use SQL with [SQLiteDatabase.SQLExecute](#).

## Properties

### BooleanValue As [Boolean](#) (read-only)

Gets the boolean value for a column.

#### Sample Code

```
// rs is a SQLiteRecordSet with a Boolean column called "AllowEmails"
If rs.Field("AllowEmails").BooleanValue Then
    MessageLabel.Text = "Emails are allowed."
End If
```

## CurrencyValue As Currency (read-only)

Gets the currency value for a column.

### Sample Code

```
// rs is a SQLiteRecordSet with a currency column called "Amount"  
Dim amount As Currency  
amount = rs.Field("Amount").CurrencyValue
```

---

## DateValue As Date (read-only)

Gets the date value for a column.

### Sample Code

```
// rs is a RecordSet with a date column called "InvoiceDate"  
Dim invoiceDate As Date  
invoiceDate = rs.Field("InvoiceDate").DateValue
```

---

## DoubleValue As Double (read-only)

Gets the double value for a column.

### Sample Code

```
// rs is a RecordSet with a double column called "InterestRate"  
Dim interestRate As Double  
interestRate = rs.Field("InterestRate").DoubleValue
```

---

## Int64Value As Int64 (read-only)

Gets the Int64 value for a column.

### Sample Code

```
// rs is a RecordSet with an Int64 column called "ID"  
Dim id As Int64  
id = rs.Field("ID").Int64Value
```

---

## IntegerValue As Integer (read-only)

Gets the Integer value for a column.

## Sample Code

```
// rs is a RecordSet with an Integer column called "Quantity"  
Dim quantity As Integer  
quantity = rs.Field("Quantity").IntegerValue
```

---

## Name As Text (read-only)

Gets the name of the column.

---

## NativeValue As MemoryBlock (read-only)

Gets the native value for a column. Useful for reading BLOB columns.

## Sample Code

```
// rs is a RecordSet with a BLOB column called "FileData"  
Dim data As MemoryBlock  
data = rs.Field("FileData").NativeValue
```

---

## TextValue As Text (read-only)

Gets the text value for a column.

## Sample Code

```
// rs is a RecordSet with a Text column called "ProductName"  
Dim productName As Text  
productName = rs.Field("ProductName").TextValue
```

---

## Value As Auto (read-only)

Gets the value for a column.

# SQLiteException

Raised when there are errors when using a [SQLiteDatabase](#).

## Class Summary

Name	SQLiteException
Type	Class
Super	<a href="#">LogicException</a>
Properties	<a href="#">CallStack</a> <a href="#">Reason</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">SQLiteDatabase</a> <a href="#">SQLiteDatabaseField</a> <a href="#">SQLiteRecordSet</a>

## Notes

You should always check for an SQLiteException after calling commands such as [SQLExecute](#) or [SQLSelect](#).

## Sample Code

```
Dim sql As Text
sql = "SELECT * FROM Team"

Dim data As SQLiteRecordSet

Try
    data = DB.SQLSelect(sql)
Catch e As SQLiteException
    ErrorLabel.Text = e.Reason
    Return
End Try

// Now you can use the results
```

# SQLiteRecordSet

A SQLiteRecordSet is a group of SQLiteDatabase rows and is the result of a SELECT statement.

## Class Summary

Name	SQLiteRecordSet
Type	Class
Properties	<a href="#">BOF</a> <a href="#">EOF</a> <a href="#">FieldCount</a>
Methods	<a href="#">Close</a> <a href="#">ColumnType</a> <a href="#">Field</a> <a href="#">IdxField</a> <a href="#">MoveFirst</a> <a href="#">MoveLast</a> <a href="#">MoveNext</a> <a href="#">MovePrevious</a> <a href="#">RecordCount</a>
Project Types	iOS
Platforms	iOS
Related	<a href="#">SQLiteDatabase</a> <a href="#">SQLiteDatabaseField</a> <a href="#">SQLiteException</a>

## Notes

You cannot use a SQLiteRecordSet to modify the data it contains. Instead you'll need to use SQL statements with [SQLiteDatabase.SQLExecute](#).

## Sample Code

Get the first column of data from the results of a table SELECT:

```
Dim sql As Text
sql = "SELECT * FROM Team"

Dim data As SQLiteRecordSet

Try
    data = DB.SQLSelect(sql)
Catch e As SQLiteException
```

```
ErrorLabel.Text = e.Reason
Return
End Try

If data <> Nil Then
    While Not data.EOF
        TextArea1.Text = TextArea1.Text + Text.FromUnicodeCodepoint(10) +
data.IdxField(0).TextValue

        data.MoveNext
    Wend
    data.Close
End If
```

## Properties

### BOF As Boolean (read-only)

Indicates if the RecordSet is pointing before the first row.

---

### EOF As Boolean (read-only)

Indicates if the RecordSet is pointing after the last row.

---

### FieldCount As Integer (read-only)

The number of columns in the RecordSet data.

---

## Methods

### Close

Closes the RecordSet.

---

### ColumnType(index As Integer) As Integer

Identifies the type of the specific column *index* (0-based).

---

### Field(name As Text) As SQLiteDatabaseField

Gets the SQLiteDatabaseField information for the specified column *name*.

---

## Field(index As Integer) As SQLiteDatabaseField

Gets the SQLiteDatabaseField information for the specified column index (0-based).

---

## IdxField(index As Integer) As SQLiteDatabaseField

Gets the SQLiteDatabaseField information for the specified column *index* (0-based).

---

## MoveFirst

Moves the RecordSet pointer to the first row of the RecordSet.

---

## MoveLast

Moves the RecordSet pointer to the last row in the RecordSet.

---

## MoveNext

Moves the RecordSet pointer to the next row in the RecordSet. If the pointer was at the last row, then this moves the pointer to EOF.

---

## MovePrevious

Moves the RecordSet pointer to the previous row in the RecordSet. If the pointer was at the first row, then this moves the pointer to BOF.

---

## RecordCount As Integer

The number of rows in the RecordSet.

---

# Exceptions

In Xojo, exceptions are used for error handling. If your code causes or raises an exception and you do not handle it, `iOSApplication.UnhandledException` will be called with information about the exception and your app will then crash.

Use `Try...Catch` blocks around code that could raise exceptions.

# FunctionNotFoundException (Class)

Raised when a soft declared function is invoked but the function could not be loaded.

## Class Summary

Name	FunctionNotFoundException
Type	Class
Inherits	<a href="#">RuntimeException</a>
Implements	n/a
Properties	<a href="#">ErrorNumber</a> , <a href="#">Message</a> , <a href="#">Reason</a>
Methods	<a href="#">CallStack</a> , <a href="#">Stack</a>
Targets	All
Platforms	All
Related	<a href="#">Declare</a>

## Notes

This exception is raised when a Soft Declare statement fails to find or load the specified function or library.

For example, this code displays a message if the function cannot be loaded:

```
Try
  Soft Declare Function getpid Lib "libc" () As Integer
Catch e As FunctionNotFoundException
  ErrorLabel.Text = "The getpid function was not found in the libc library."
End Try
```

# KeyNotFoundException (Class)

Raised when you try to access an item in a [Dictionary](#) using a key that is not in the Dictionary.

## Class Summary

Name	KeyNotFoundException
Type	Class
Inherits	<a href="#">RuntimeException</a>
Implements	n/a
Properties	<a href="#">ErrorNumber</a> , <a href="#">Message</a> , <a href="#">Reason</a>
Methods	<a href="#">CallStack</a> , <a href="#">Stack</a>
Targets	All
Platforms	All
Related	<a href="#">Dictionary</a>

# NilObjectException (Class)

Raised when invoking a method on a Nil base expression or passing a Nil object to a function that expects a non-Nil object.

## Class Summary

Name	NilObjectException
Type	Class
Inherits	<a href="#">RuntimeException</a>
Properties	<a href="#">ErrorNumber</a> , <a href="#">Message</a> , <a href="#">Reason</a>
Methods	<a href="#">CallStack</a> , <a href="#">Stack</a>
Targets	All
Platforms	All
Related	

## Notes

A NilObjectException occurs when you try to access an object that does not have an instance. The most common occurrence of this error when an object is not instantiated using the New operator before accessing its members.

For example, this code causes a NilObjectException:

```
Dim d As Date
If d.Year = 2014 Then // NilObjectException on this line because d was not instantiated
    // Do something
End If
```

Instead, always remember to use the New operator:

```
Dim d As New Date
If d.Year = 2014 Then
    // Do something
End If
```

# ObjCException (Class)

Raised when an Objective-C declare throws an exception.

## Class Summary

Name	ObjCException
Type	Class
Inherits	<a href="#">RuntimeException</a>
Implements	n/a
Properties	<a href="#">ErrorNumber</a> , <a href="#">Handle</a> , <a href="#">Message</a> , <a href="#">Reason</a>
Methods	<a href="#">CallStack</a> , <a href="#">Stack</a>
Targets	All
Platforms	All
Related	<a href="#">Declare</a>

## Properties

### Handle As Ptr

The handle for the exception.

# OutOfBoundsException (Class)

Raised when accessing an element that is out of bounds of the container.

## Class Summary

Name	OutOfBoundsException
Type	Class
Inherits	<a href="#">RuntimeException</a>
Implements	n/a
Properties	<a href="#">ErrorNumber</a> , <a href="#">Message</a> , <a href="#">Reason</a>
Methods	<a href="#">CallStack</a> , <a href="#">Stack</a>
Targets	All
Platforms	All
Related	

## Notes

Examples for when this occur include:

- attempting to access an array element larger (or smaller) than the size of the array
- Attempting to access a row that it outside the available rows for a container

# OutOfMemoryException (Class)

Raised when trying to allocate more memory than is available.

## Class Summary

Name	OutOfMemoryException
Type	Class
Inherits	<a href="#">RuntimeException</a>
Implements	n/a
Properties	<a href="#">ErrorNumber</a> , <a href="#">Message</a> , <a href="#">Reason</a>
Methods	<a href="#">CallStack</a> , <a href="#">Stack</a>
Targets	All
Platforms	All
Related	<a href="#">MemoryBlock</a> , <a href="#">MutableMemoryBlock</a>

## Notes

Typically used with [MemoryBlock](#) or [MutableMemoryBlock](#) when trying to allocate large blocks of memory that exceeds what is available.

Low-level memory situations are inherently problematic if you are frequently requesting small blocks of memory. Because of this, there is no guarantee that an `OutOfMemoryException` can even be raised. If the system memory is critically low, the process itself may crash as it attempts to reserve memory in order to display an exception message. In this situation, you might consider reserving a single large block of memory up front and using that within your application rather than repeatedly requesting smaller blocks.

# RuntimeException (Class)

The base class of all exceptions.

## Class Summary

Name	RuntimeException
Type	Class
Inherits	n/a
Implements	n/a
Properties	<a href="#">ErrorNumber</a> , <a href="#">Message</a> , <a href="#">Reason</a>
Methods	<a href="#">CallStack</a> , <a href="#">Stack</a>
Targets	All
Platforms	All
Related	<a href="#">StackFrame</a>

## Notes

Exceptions are raised when an error condition occurs. Unhandled exceptions cause your app to terminate. You can handle exceptions using a [Try...Catch](#) code block. Generally, you always want to check for specific exceptions that could be caused by your code.

For example, a Dictionary raises an exception if you try to remove a key that does not exist. You can catch that exception like this:

```
Try
    d1.RemoveByKey("Test")
Catch e As KeyNotFoundException
    // You may want to report the error to the user
    ErrorLabel.Text = "The key 'Test' does not exist."
End Try
```

Now in the case of Dictionary, it has a method that can check if a key exists. It is more efficient to use this method than to catch the exception:

```
If d1.HasKey("Test") Then
    d1.RemoveByKey("Test")
Else
    ErrorLabel.Text = "The key 'Test' does not exist."
End If
```

## Properties

### ErrorNumber As Integer

The error number for the exception, if appropriate.

---

### Message As String

A description of why the exception occurred.

 Not available on iOS. Use Reason instead.

---

### Reason As Text (read-only)

A non-localized description of why the exception occurred. This is not meant to be presented to end users.

## Methods

### CallStack As StackFrame() (read-only)

The stack when the exception was first raised.

---

### Stack As As String()

The stack when the exception was first raised.

 Not available on iOS. Use CallStack instead.

# StackOverflowException (Class)

Raised when the program is close to running out of stack space and crashing.

## Class Summary

Name	StackOverflowException
Type	Class
Inherits	<a href="#">RuntimeException</a>
Implements	n/a
Properties	<a href="#">ErrorNumber</a> , <a href="#">Message</a> , <a href="#">Reason</a>
Methods	<a href="#">CallStack</a> , <a href="#">Stack</a>
Targets	All
Platforms	All
Related	

## Notes

This exception can occur when the calling chain gets too long. This can easily happen when your code makes a recursive call without providing a way to terminate the recursion (or the condition to terminate the recursion takes too many calls to occur).

This can also occur in situations where two methods or events call each other repeatedly because of UI actions (such as a Canvas Paint event handler calling a method that causes the Canvas to refresh).

# TypeMismatchException (Class)

Raised when assigning to a variable from an incompatible type.

## Class Summary

Name	TypeMismatchException
Type	Class
Inherits	<a href="#">RuntimeException</a>
Implements	n/a
Properties	<a href="#">ErrorNumber</a> , <a href="#">Message</a> , <a href="#">Reason</a>
Methods	<a href="#">CallStack</a> , <a href="#">Stack</a>
Targets	All
Platforms	All
Related	

## Notes

This exception occurs when you try to assign a value of an incorrect type to an object. The error occurs only if the value cannot be determined at run-time, for example when using [Auto](#) or [Variant](#). Normally, errors with incompatible type are caught at compile-time.

# Classic Framework

The classic framework contains the desktop, web and console frameworks. The [Xojo Framework](#) is an updated and more modern framework that you can also use in your apps.

The framework methods and classes are described in their appropriate sections.

- [Console](#)
- [Desktop](#)
- [Web](#)

# Classic Console Framework

These are the methods, classes and interfaces used by console, desktop and web apps.

## Framework Summary

AddressBook	<a href="#">AddressBook</a> <a href="#">AddressBookAddress</a> <a href="#">AddressBookContact</a> <a href="#">AddressBookData</a> <a href="#">AddressBookGroup</a> <a href="#">AddressBookRecord</a>
AppleEvent	<a href="#">AppleEvent</a> <a href="#">AppleEventDescList</a> <a href="#">AppleEventObjectSpecifier</a>
Databases	<a href="#">DatabaseField</a> <a href="#">DatabaseRecord</a> <a href="#">MSSQLServerDatabase</a> <a href="#">MSSQLServerPreparedStatement</a> <a href="#">MySQLCommunityServer</a> <a href="#">MySQLPreparedStatement</a> <a href="#">ODBCConstant</a> <a href="#">ODBCDatabase</a> <a href="#">ODBCPreparedStatement</a> <a href="#">PostgreSQLDatabase</a> <a href="#">PostgreSQLLargeObject</a> <a href="#">PostgreSQLPreparedStatement</a> <a href="#">PreparedStatement</a> <a href="#">RecordSet</a> <a href="#">SQLiteBackupInterface</a> <a href="#">SQLiteBlob</a> <a href="#">SQLiteDatabase</a> <a href="#">SQLitePreparedStatement</a>
Files	<a href="#">BinaryStream</a> <a href="#">FileType</a> <a href="#">FolderItem</a> <a href="#">Permissions</a> <a href="#">SpecialFolder</a> <a href="#">SpotlightQuery</a> <a href="#">VirtualVolume</a>
Game	<a href="#">GameInputDevice</a> <a href="#">GameInputElement</a> <a href="#">GameInputManager</a>

Graphics	<a href="#">ArcShape</a> <a href="#">CMY</a> <a href="#">CurveShape</a> <a href="#">DarkTingeColor</a> <a href="#">DisabledTextColor</a> <a href="#">FigureShape</a> <a href="#">FillColor</a> <a href="#">FrameColor</a> <a href="#">Graphics</a> <a href="#">Group2D</a> <a href="#">HighlightColor</a> <a href="#">HSV</a> <a href="#">LightBevelColor</a> <a href="#">LightTingeColor</a> <a href="#">Picture</a> <a href="#">Object2D</a> <a href="#">OpenPrinter</a> <a href="#">OvalShape</a> <a href="#">PixmapShape</a> <a href="#">RGB</a> <a href="#">RGBSurface</a> <a href="#">RoundRectShape</a> <a href="#">StringShape</a> <a href="#">TextColor</a>
Global Methods	<a href="#">Beep</a> <a href="#">Font</a> <a href="#">FontCount</a> <a href="#">GetTemporaryFolderItem</a> <a href="#">Input</a> <a href="#">IsNumeric</a> <a href="#">Print</a> <a href="#">Speak</a> <a href="#">StdErr</a> <a href="#">StdIn</a> <a href="#">StdOut</a> <a href="#">Volume</a> <a href="#">VolumeCount</a>
Introspection	<a href="#">Introspection</a> <a href="#">AttributeInfo</a> <a href="#">ConstructorInfo</a> <a href="#">MemberInfo</a> <a href="#">ParameterInfo</a> <a href="#">PropertyInfo</a> <a href="#">TypeInfo</a> <a href="#">GetTypeInfo</a> <a href="#">ObjectIterator</a> <a href="#">GetType</a>
Math	<a href="#">Random</a> <a href="#">Val</a>

MS Office Automation	<a href="#">ExcelApplication</a> <a href="#">Office Automation</a> <a href="#">Office Enumerations</a> <a href="#">PowerPointApplication</a> <a href="#">WordApplication</a>
Modules	Crypto Keyboard COM Encodings REALbasic System
Networking	AutoDiscovery Datagram DecodeQuotedPrintable DecodeURLComponent EasyTCPSocket EasyUDPSocket EmailAttachment EmailHeaders EmailMessage EncodeQuotedPrintable EncodeURLComponent HTTPSecureSocket HTTPSocket InternetHeaders MD5 MD5Digest Network NetworkInterface POP3SecureSocket POP3Socket ServerSocket ShowURL SocketCore SMTPSecureSocket SMTPSocket SSLSocket TCPSocket UDPSocket
Regular Expressions	<a href="#">RegEx</a> <a href="#">RegExException</a> <a href="#">RegExMatch</a> <a href="#">RegExOptions</a> <a href="#">RegExSearchPatternException</a>

Strings	<a href="#">Asc</a> <a href="#">AscB</a> <a href="#">Chr</a> <a href="#">ChrB</a> <a href="#">CountFields</a> <a href="#">CStr</a> <a href="#">DecodeBase64</a> <a href="#">DecodeHex</a> <a href="#">EncodeBase64</a> <a href="#">EncodeHex</a> <a href="#">Encoding</a> <a href="#">EndOfLine</a> <a href="#">Format</a> <a href="#">GetFontTextEncoding</a> <a href="#">GetInternetTextEncoding</a> <a href="#">GetTextConverter</a> <a href="#">GetTextEncoding</a> <a href="#">GuessJapaneseEncoding</a> <a href="#">InStr</a> <a href="#">Left</a> <a href="#">Len</a> <a href="#">LenB</a> <a href="#">Lowercase</a> <a href="#">LTrim</a> <a href="#">Mid</a> <a href="#">MidB</a> <a href="#">NthField</a> <a href="#">Replace</a> <a href="#">ReplaceAll</a> <a href="#">ReplaceLineEndings</a> <a href="#">Right</a> <a href="#">RTrim</a> <a href="#">Str</a> <a href="#">StrComp</a> <a href="#">String</a> <a href="#">Titlecase</a> <a href="#">Trim</a> <a href="#">Uppercase</a> <a href="#">TextConverter</a> <a href="#">TextEncoding</a>
Styled Text	<a href="#">Paragraph</a> <a href="#">Range</a> <a href="#">StyledText</a> <a href="#">StyledTextPrinter</a> <a href="#">StyleRun</a>
Threading	<a href="#">CriticalSection</a> <a href="#">Mutex</a> <a href="#">Semaphore</a> <a href="#">Thread</a>

XML	<a href="#">SOAPException</a> <a href="#">SOAPMethod</a> <a href="#">SOAPResult</a> <a href="#">XMLAttribute</a> <a href="#">XMLAttributeList</a> <a href="#">XMLCDATASection</a> <a href="#">XMLComment</a> <a href="#">XMLContentMode</a> <a href="#">XMLDocument</a> <a href="#">XMLDOMException</a> <a href="#">XMLElement</a> <a href="#">XMLNamespaces</a> <a href="#">XMLNode</a> <a href="#">XMLNodeList</a> <a href="#">XMLNodeType</a> <a href="#">XMLProcessingInstruction</a> <a href="#">XMLReader</a> <a href="#">XMLReaderException</a> <a href="#">XMLStyleSheet</a> <a href="#">XMLTextNode</a> <a href="#">XMLXsltHandler</a>
Classes	<a href="#">ConsoleApplication</a> <a href="#">Date</a> <a href="#">Dictionary</a> <a href="#">JSONItem</a> <a href="#">KeyChain</a> <a href="#">MemoryBlock</a> <a href="#">Point</a> <a href="#">PrinterSetup</a> <a href="#">Readable</a> <a href="#">Rect</a> <a href="#">RegistryItem</a> <a href="#">Serial</a> <a href="#">SerialPort</a> <a href="#">ServiceApplication</a> <a href="#">Shell</a> <a href="#">Size</a> <a href="#">Sound</a> <a href="#">StandardInputStream</a> <a href="#">StandardOutputStream</a> <a href="#">Timer</a> <a href="#">Variant</a> <a href="#">WeakRef</a> <a href="#">Writeable</a> <a href="#">XojoScript</a>

# Desktop Framework Overview (Classic)

These are the controls, classes and interfaces used by Desktop apps.

## Framework Summary

Controls	<a href="#">BevelButton</a> <a href="#">Canvas</a> <a href="#">CheckBox</a> <a href="#">ComboBox</a> <a href="#">ContainerControl</a> <a href="#">Control</a> <a href="#">DisclosureTriangle</a> <a href="#">GroupBox</a> <a href="#">HTMLViewer</a> <a href="#">ImageWell</a> <a href="#">Label</a> <a href="#">Line</a> <a href="#">ListBox</a> <a href="#">MoviePlayer</a> <a href="#">NotePlayer</a> <a href="#">Oval</a> <a href="#">PagePanel</a> <a href="#">Placard</a> <a href="#">PopupArrow</a> <a href="#">PopupMenu</a> <a href="#">OLEContainer</a> <a href="#">OpenGLSurface</a> <a href="#">ProgressBar</a> <a href="#">ProgressWheel</a> <a href="#">PushButton</a> <a href="#">RadioButton</a> <a href="#">Rectangle</a> <a href="#">RectControl</a> <a href="#">RoundRectangle</a> <a href="#">Scrollbar</a> <a href="#">SegmentedControl</a> <a href="#">Separator</a> <a href="#">Slider</a> <a href="#">TabPanel</a> <a href="#">TextArea</a> <a href="#">UpDownArrows</a>
----------	---

Classes	<a href="#">Application</a> <a href="#">Clipboard</a> <a href="#">DockItem</a> <a href="#">DragItem</a> <a href="#">MouseCursor</a> <a href="#">Movie</a> <a href="#">Screen</a> <a href="#">Toolbar</a> <a href="#">TrayItem</a> <a href="#">MDIWindow</a> <a href="#">Window</a>
Dialogs	<a href="#">FolderItemDialog</a> <a href="#">OpenDialog</a> <a href="#">SaveAsDialog</a> <a href="#">SelectFolderDialog</a> <a href="#">MessageDialog</a>
Global Methods	<a href="#">ClearFocus</a> <a href="#">EnableMenuItems</a> <a href="#">ExportPicture</a> <a href="#">GetOpenFolderItem</a> <a href="#">GetSaveFolderItem</a> <a href="#">IsContextualClick</a> <a href="#">MsgBox</a> <a href="#">OpenPrinterDialog</a> <a href="#">Screen</a> <a href="#">SelectColor</a> <a href="#">SelectFolder</a> <a href="#">UserCancelled</a> <a href="#">Window</a> <a href="#">WindowCount</a>
Menus	<a href="#">AppleMenuItem</a> <a href="#">ApplicationMenuItem</a> <a href="#">MenuBar</a> <a href="#">MenuItem</a> <a href="#">PrefsMenuItem</a> <a href="#">QuitMenuItem</a>
Modules	<a href="#">Cursors</a>
Reporting	<a href="#">Report</a> <a href="#">Reports Module</a> <a href="#">RBReportDocument</a> <a href="#">ReportField</a> <a href="#">ReportLabel</a> <a href="#">ReportLineShape</a> <a href="#">ReportPageNumberLabel</a> <a href="#">ReportPicture</a> <a href="#">ReportOvalShape</a> <a href="#">ReportRectangleShape</a> <a href="#">ReportRoundRectangleShape</a>

Project Types	Desktop
Platforms	All
Related	<a href="#">Classic Console Framework</a>

# Web Framework Overview (Classic)

These are the controls, classes and interfaces used by Web apps.

## Framework Summary

Controls	<a href="#">WebButton</a> <a href="#">WebCanvas</a> <a href="#">WebCheckBox</a> <a href="#">WebContainer</a> <a href="#">WebControl</a> <a href="#">WebControlWrapper</a> <a href="#">WebDialog</a> <a href="#">WebFileUploader</a> <a href="#">WebHTMLViewer</a> <a href="#">WebImageView</a> <a href="#">WebLabel</a> <a href="#">WebLink</a> <a href="#">WebListBox</a> <a href="#">WebMapView</a> <a href="#">WebMoviePlayer</a> <a href="#">WebPageSource</a> <a href="#">WebPopupMenu</a> <a href="#">WebProgressBar</a> <a href="#">WebProgressWheel</a> <a href="#">WebRadioGroup</a> <a href="#">WebRectangle</a> <a href="#">WebScrollbar</a> <a href="#">WebSearchField</a> <a href="#">WebSegmentedControl</a> <a href="#">WebSeparator</a> <a href="#">WebSlider</a> <a href="#">WebTextArea</a> <a href="#">WebTextField</a> <a href="#">WebToolbar</a> <a href="#">WebToolbarButton</a> <a href="#">WebToolbarContainer</a> <a href="#">WebToolbarFlexibleSpace</a> <a href="#">WebToolbarItem</a> <a href="#">WebToolbarMenu</a> <a href="#">WebToolbarSeparator</a> <a href="#">WebToolbarSpace</a> <a href="#">WebYouTubeMovie</a>
----------	--

Classes	<a href="#">MouseEvent</a> <a href="#">Session</a> <a href="#">WebAnimator</a> <a href="#">WebApplication</a> <a href="#">WebDeviceLocation</a> <a href="#">WebFile</a> <a href="#">WebGraphics</a> <a href="#">WebMapLocation</a> <a href="#">WebMenuItem</a> <a href="#">WebObject</a> <a href="#">WebPicture</a> <a href="#">WebRequest</a> <a href="#">WebSession</a> <a href="#">WebSharingSiteMovie</a> <a href="#">WebStyle</a> <a href="#">WebTextControl</a> <a href="#">WebTimer</a> <a href="#">WebUploadedFile</a>
Interfaces	<a href="#">DataCell</a>
Layout	<a href="#">WebPage</a> <a href="#">WebView</a>
Modules	<a href="#">WebCursors</a> <a href="#">XojoCloud</a>
Other	<a href="#">KeyEvent</a>
Project Types	Web
Platforms	All
Related	<a href="#">Classic Console Framework</a>



# System Requirements

[Xojo 2015 Release 3](#)

[Table of Contents](#)

- [Xojo IDE](#)
- [Desktop Apps](#)
- [Web Apps](#)
- [Console Apps](#)
- [iOS Apps](#)
- [Linux Information](#)
  - [64-bit Configuration](#)
  - [International Components for Unicode \(libicu\)](#)

## Xojo IDE

The Xojo IDE can be used on systems that meet the following requirements:

	Windows	OS X	Linux (x86, x86-64)
<b>OS</b>	<ul style="list-style-type: none"> <li>• Windows Vista (x86 or x64)</li> <li>• Windows 7 (x86 or x64)</li> <li>• Windows 8.x (x86 or x64)</li> <li>• Windows 10 (x86 or x64)</li> </ul>	<ul style="list-style-type: none"> <li>• OS X Lion 10.7.x</li> <li>• OS X Mountain Lion 10.8.x</li> <li>• OS X Mavericks 10.9.x</li> <li>• OS X Yosemite 10.10.x</li> <li>• OS X El Capitan 10.11.x</li> </ul> <p>iOS development requires 10.9.x or later and Xcode 6.x or later. See below for iOS requirements.</p>	<p>32-bit recommend (refer to <a href="#">Linux Information</a> below regarding 64-bit)</p> <ul style="list-style-type: none"> <li>• Linux Mint 16 or later (recommended)</li> <li>• Ubuntu 10.04 or later</li> <li>• Debian 6.0 or later</li> <li>• OpenSUSE 11.3 or later</li> <li>• Fedora 13 Desktop or later</li> <li>• CentOS 6.0 or later</li> </ul> <p>The Xojo IDE does not run on ARM.</p>
<b>RAM</b>	Minimum 2GB	Minimum 2GB	Minimum 2GB
<b>Other</b>	PDF viewer for documentation		PDF viewer for documentation

## Desktop Apps

Desktop apps created with Xojo have these requirements:

	Windows	OS X	Linux (x86, x86-64)	Raspberry Pi 2
<b>OS</b>	<p>32-bit and 64-bit versions of Windows are supported.</p> <ul style="list-style-type: none"> <li>• Windows Vista</li> <li>• Windows 7</li> <li>• Windows 8.x</li> <li>• Windows 10</li> </ul>	<ul style="list-style-type: none"> <li>• OS X Lion 10.7.x</li> <li>• OS X Mountain Lion 10.8.x</li> <li>• OS X Mavericks 10.9.x</li> <li>• OS X Yosemite 10.10.x</li> <li>• OS X El Capitan 10.11.x</li> </ul>	<p>32-bit and 64-bit versions of Linux are supported.</p> <ul style="list-style-type: none"> <li>• Linux Mint 16 or later</li> <li>• Ubuntu 10.04 or later</li> <li>• Debian 6.0 or later</li> <li>• OpenSUSE 11.3 or later</li> <li>• Fedora 13 Desktop or later</li> <li>• CentOS 6.0 or later (7.0 or later for x86-64)</li> </ul>	<p>32-bit ARMv7 is supported.</p> <ul style="list-style-type: none"> <li>• Raspbian Wheezy</li> <li>• Raspbian Jessie</li> </ul>

## Web Apps

Web apps consist of two parts: user interface and the app itself. Your users can use a web app in one of the following browsers for these platforms:

	Windows	OS X / OS X Server	Linux	iOS	Android	Raspberry Pi
<b>Browser</b>	<ul style="list-style-type: none"> <li>• Chrome 37+</li> <li>• Firefox 17+</li> <li>• Internet Explorer 9+</li> <li>• Edge</li> </ul>	<ul style="list-style-type: none"> <li>• Safari 6.0+</li> <li>• Chrome 37+</li> <li>• Firefox 17+</li> </ul>	<ul style="list-style-type: none"> <li>• Chrome 37+</li> <li>• Firefox 17+</li> </ul>	<ul style="list-style-type: none"> <li>• Mobile Safari</li> </ul>	<ul style="list-style-type: none"> <li>• Chrome for Android</li> </ul>	<ul style="list-style-type: none"> <li>• Raspbian Web</li> </ul>

The apps can be deployed to servers with these requirements:

	Windows	OS X / OS X Server	Linux (x86, x86-64)	Raspberry Pi
<b>OS</b>	32-bit and 64-bit servers are supported. <ul style="list-style-type: none"> <li>• Windows Server 2008</li> <li>• Windows Server 2012</li> <li>• Windows Server 2016</li> <li>• Windows Vista</li> <li>• Windows 7</li> <li>• Windows 8.x</li> <li>• Windows 10</li> </ul>	<ul style="list-style-type: none"> <li>• OS X Lion 10.7.x</li> <li>• OS X Mountain Lion 10.8.x</li> <li>• OS X Mavericks 10.9.x</li> <li>• OS X Yosemite 10.10.x</li> <li>• OS X El Capitan 10.11.x</li> </ul>	32-bit and 64-bit servers are supported. <ul style="list-style-type: none"> <li>• CentOS 6.0 or later (7.0 or later for x86-64)</li> <li>• Debian 6.0 or later</li> <li>• OpenSUSE 11.3 or later</li> <li>• Linux Mint 16 or later</li> <li>• Ubuntu 10.04 or later</li> <li>• Fedora 13 later</li> </ul>	32-bit ARMv7 is supported. <ul style="list-style-type: none"> <li>• Raspbian Wheezy</li> <li>• Raspbian Jessie</li> </ul>
<b>Deployment</b>	<ul style="list-style-type: none"> <li>• <a href="#">Standalone</a></li> <li>• <a href="#">Apache 2</a></li> <li>• <a href="#">IIS</a></li> </ul>	<ul style="list-style-type: none"> <li>• <a href="#">Standalone</a></li> <li>• <a href="#">Apache 2</a></li> </ul>	<ul style="list-style-type: none"> <li>• <a href="#">Xojo Cloud</a></li> <li>• <a href="#">Standalone</a></li> <li>• <a href="#">Apache 2</a></li> </ul>	

## Console Apps

Console apps do not have a user interface and run on systems with these requirements (or any of the server requirements listed for Web Apps):

	Windows	OS X / OS X Server	Linux (x86, x86-64)	Raspberry Pi
<b>OS</b>	<ul style="list-style-type: none"> <li>• Windows Vista</li> <li>• Windows 7</li> <li>• Windows 8.x</li> <li>• Windows 10</li> </ul>	<ul style="list-style-type: none"> <li>• OS X Lion 10.7.x</li> <li>• OS X Mountain Lion 10.8.x</li> <li>• OS X Mavericks 10.9.x</li> <li>• OS X Yosemite 10.10.x</li> <li>• OS X El Capitan 10.11.x</li> </ul>	<ul style="list-style-type: none"> <li>• Linux Mint 16 or later</li> <li>• Ubuntu 10.04 or later</li> <li>• Debian 6.0 or later</li> <li>• OpenSUSE 11.3 or later</li> <li>• Fedora 13 Desktop or later</li> <li>• CentOS 6.0 or later (7.0 or later for x86-64)</li> </ul>	32-bit ARMv7 is supported. <ul style="list-style-type: none"> <li>• Raspbian Wheezy</li> <li>• Raspbian Jessie</li> </ul>

## iOS Apps

In order to work on iOS projects, you must be using Xojo on OS X 10.9 and later with Xcode 6.x or later (required for iOS Simulator).

<b>Supported iOS Versions</b>	<ul style="list-style-type: none"> <li>• iOS 7</li> <li>• iOS 8</li> <li>• iOS 9</li> </ul>
<b>Supported iOS Devices</b>	<ul style="list-style-type: none"> <li>• iPhone 4S and newer iPhones</li> <li>• iPad mini (all models)</li> <li>• iPad 2 and newer models</li> <li>• iPod Touch (5th gen and newer)</li> </ul>
<b>iOS Developer Subscription Requirements</b>	<ul style="list-style-type: none"> <li>• Deploying to iOS devices</li> <li>• Submitting to App Store</li> <li>• Test Flight</li> </ul>

## Linux Information

Because various Linux distributions have different libraries installed by default, you may need to install additional libraries installed before your Xojo apps will run on Linux. At a minimum, Xojo requires these Linux libraries:

<b>Always Required</b>	<b>Required for Desktop</b>	<b>Optional</b>
glib 2.0	GTK+ 2.20	libwebkitgtk-1.0.0 or libgtkhtml (HTMLViewer)
glibc-2.11 (32-bit) glibc-2.14 (64-bit)		libsoup 2.4 (Xojo.Net.HTTPSocket)
libstdc++.so.6.0.13		
libc 4.2+		

If you're looking for a specific version and distribution, check out [mirrors.kernel.org](http://mirrors.kernel.org).

### 64-bit Configuration

For best results, create 64-bit Xojo apps for distribution on 64-bit Linux systems. If you have to distribute a 32-bit Xojo app on a Linux distribution, you need to ensure that the 32-bit libraries are installed. They are not installed by default, so you'll have to install them manually. Below are some command that might be helpful.

<b>Debian/Ubuntu/Mint</b>	<b>CentOS/Fedora/OpenSUSE</b>
<code>sudo apt-get install ia32-libs</code> <code>sudo apt-get install ia32-libs-multiarch</code>	<code>sudo yum install ia32-libs</code> <code>sudo yum install ia32-libs-multiarch</code>
<code>sudo dpkg --add-architecture i386</code> <code>sudo apt-get update</code> <code>sudo apt-get install ia32-libs-multiarch</code>	<code>sudo yum install glib2.i686 libgcc.i686</code> <code>libstdc++.i686</code>
<code>sudo apt-get install libc6:i386</code> <code>sudo apt-get install libc6</code> <code>sudo apt-get install libc6:i386</code>	<code>sudo yum provides missinglibrarypathorname</code> <code>sudo yum install packagenamewithlibrary</code>

Debian/Ubuntu/Mint	CentOS/Fedora/OpenSUSE
Ubuntu 14+: <pre>sudo dpkg --add-architecture i386 sudo apt-get update sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386 libglib2.0-0:i386 libsoup2.4- 1:i386 libc6:i386 libgtk2.0-0:i386</pre>	<pre>/usr/bin/yum -y install glib2.i686 libgcc.i686 libstdc++.i686</pre>
<pre>sudo apt-get install libgtk2.0-0:i386</pre>	<pre>yum install libc6.i686</pre>

If you are still having trouble identifying necessary libraries, the `ldd` command might help. You can run from Terminal in the Xojo directory to return a list of libraries required by Xojo and their status on the system:

```
ldd Xojo
```

### International Components for Unicode (libc6)

Most recent Linux distributions have `libc6` installed, but depending on the Linux distribution you may need to install it yourself. These command may be helpful:

Debian/Ubuntu/Mint	CentOS/Fedora/OpenSUSE
<pre>sudo apt-get install libc6</pre>	<pre>yum install libc6</pre>

 Ubuntu 12.04 64-bit is unable to have both 32-bit and 64-bit `libc6` installed. If you require this, use 12.10 or later or create a 64-bit build.

# System Requirements (2015r2.x)

## Table of Contents

- [Xojo IDE](#)
- [Desktop Apps](#)
- [Web Apps](#)
- [Console Apps](#)
- [iOS Apps](#)
- [Linux Information](#)
  - [64-bit Configuration](#)

## Xojo IDE

The Xojo IDE can be used on systems that meet the following requirements:

	Windows	OS X	Linux
<b>OS</b>	<ul style="list-style-type: none"> <li>• Windows Vista (x86 or x64)</li> <li>• Windows 7 (x86 or x64)</li> <li>• Windows 8.x (x86 or x64)</li> <li>• Windows 10 (x86 or x64)</li> </ul>	<ul style="list-style-type: none"> <li>• OS X Lion 10.7.x</li> <li>• OS X Mountain Lion 10.8.x</li> <li>• OS X Mavericks 10.9.x</li> <li>• OS X Yosemite 10.10.x</li> </ul> <p>Note: iOS development requires 10.9.x or later and Xcode 6.x. <a href="#">See below</a> for iOS requirements.</p>	<p>32-bit recommend (refer to <a href="#">Linux Information</a> below regarding 64-bit)</p> <ul style="list-style-type: none"> <li>• Linux Mint 16 or later (recommended)</li> <li>• Ubuntu 10.04 or later</li> <li>• Debian 6.0 or later</li> <li>• OpenSUSE 11.3 or later</li> <li>• Fedora 13 Desktop or later</li> <li>• CentOS 6.0 or later</li> </ul> <p>Note: ARM architecture is not supported</p>
<b>RAM</b>	Minimum 2GB	Minimum 2GB	Minimum 2GB
<b>Other</b>	PDF viewer for documentation		PDF viewer for documentation

## Desktop Apps

Desktop apps created with Xojo have these requirements:

	Windows	OS X	Linux
<b>OS</b>	<ul style="list-style-type: none"> <li>• Windows XP SP3</li> <li>• Windows Vista</li> <li>• Windows 7</li> <li>• Windows 8.x</li> <li>• Windows 10</li> </ul>	<ul style="list-style-type: none"> <li>• OS X Lion 10.7.x</li> <li>• OS X Mountain Lion 10.8.x</li> <li>• OS X Mavericks 10.9.x</li> <li>• OS X Yosemite 10.10.x</li> </ul>	<ul style="list-style-type: none"> <li>• Linux Mint 16 or later</li> <li>• Ubuntu 10.04 or later</li> <li>• Debian 6.0 or later</li> <li>• OpenSUSE 11.3 or later</li> <li>• Fedora 13 Desktop or later</li> <li>• CentOS 6.0 or later</li> </ul>

## Web Apps

Web apps consist of two parts: user interface and the app itself. Your users can use a web app in one of the following browsers for these platforms:

	Windows	OS X / OS X Server	Linux	iOS	Android
<b>Browser</b>	<ul style="list-style-type: none"> <li>• Chrome 37+</li> <li>• Firefox 17+</li> <li>• Internet Explorer 8+</li> </ul>	<ul style="list-style-type: none"> <li>• Safari 6.0+</li> <li>• Chrome 37+</li> <li>• Firefox 17+</li> </ul>	<ul style="list-style-type: none"> <li>• Chrome 37+</li> <li>• Firefox 17+</li> </ul>	<ul style="list-style-type: none"> <li>• Mobile Safari</li> </ul>	<ul style="list-style-type: none"> <li>• Chrome for Android</li> </ul>

The apps can be deployed to servers with these requirements:

	Windows	OS X / OS X Server	Linux
<b>OS</b>	<ul style="list-style-type: none"> <li>• Windows Server 2008</li> <li>• Windows Server 2012</li> <li>• Windows Server 2016</li> <li>• Windows Vista</li> <li>• Windows 7</li> <li>• Windows 8.x</li> <li>• Windows 10</li> </ul>	<ul style="list-style-type: none"> <li>• OS X Lion 10.7.x</li> <li>• OS X Mountain Lion 10.8.x</li> <li>• OS X Mavericks 10.9.x</li> <li>• OS X Yosemite 10.10.x</li> </ul>	<ul style="list-style-type: none"> <li>• CentOS 6.0 or later</li> <li>• Debian 6.0 or later</li> <li>• OpenSUSE 11.3 or later</li> <li>• Linux Mint 16 or later</li> <li>• Ubuntu 10.04 or later</li> <li>• Fedora 13 later</li> </ul>
<b>Deployment</b>	<ul style="list-style-type: none"> <li>• <a href="#">Standalone</a></li> <li>• <a href="#">Apache 2</a></li> <li>• <a href="#">IIS</a></li> </ul>	<ul style="list-style-type: none"> <li>• <a href="#">Standalone</a></li> <li>• <a href="#">Apache 2</a></li> </ul>	<ul style="list-style-type: none"> <li>• <a href="#">Xojo Cloud</a></li> <li>• <a href="#">Standalone</a></li> <li>• <a href="#">Apache 2</a></li> </ul>

## Console Apps

Console apps do not have a user interface and run on systems with these requirements (or any of the server requirements listed for Web Apps):

	Windows	OS X / OS X Server	Linux
<b>OS</b>	<ul style="list-style-type: none"> <li>• Windows XP SP3 or later</li> <li>• Windows Vista</li> <li>• Windows 7</li> <li>• Windows 8.x</li> <li>• Windows 10</li> </ul>	<ul style="list-style-type: none"> <li>• OS X Lion 10.7.x</li> <li>• OS X Mountain Lion 10.8.x</li> <li>• OS X Mavericks 10.9.x</li> <li>• OS X Yosemite 10.10.x</li> </ul>	<ul style="list-style-type: none"> <li>• Linux Mint 16 or later</li> <li>• Ubuntu 10.04 or later</li> <li>• Debian 6.0 or later</li> <li>• OpenSUSE 11.3 or later</li> <li>• Fedora 13 Desktop or later</li> <li>• CentOS 6.0 or later</li> </ul>

## iOS Apps

In order to work on iOS projects, you must be using Xojo on OS X 10.9 and later with Xcode 6.x (required for iOS Simulator).

<b>Supported iOS Versions</b>	<ul style="list-style-type: none"> <li>• iOS 7</li> <li>• iOS 8</li> <li>• iOS 9</li> </ul>
<b>Supported iOS Devices</b>	<ul style="list-style-type: none"> <li>• iPhone 4S and newer iPhones</li> <li>• iPad mini (all models)</li> <li>• iPad 2 and newer models</li> <li>• iPod Touch (5th gen and newer)</li> </ul>
<b>iOS Developer Subscription Requirements</b>	<ul style="list-style-type: none"> <li>• Deploying to iOS devices</li> <li>• Submitting to App Store</li> <li>• Test Flight</li> </ul>

## Linux Information

Because various Linux distributions provide different libraries installed by default, you may need to install additional libraries installed before your Xojo apps will run on Linux. At a minimum, Xojo requires these Linux libraries:

Always Required	Required for Desktop	Optional
glib 2.0	GTK+ 2.20	libwebkitgtk-1.0.0 or libgtkhtml (HTMLViewer)
glibc-2.11		libsoup 2.4 (Xojo.Net.HTTPSocket)
libstdc++.so.6.0.13		
libc 4.2+		

If you're looking for a specific version and distribution, check out [mirrors.kernel.org](http://mirrors.kernel.org).

### 64-bit Configuration

Xojo currently creates 32-bit Linux apps. For easiest installation, you should use a 32-bit Linux distribution. If you want to run your Xojo apps on a 64-bit Linux distribution, you will need to ensure that the necessary 32-bit libraries are installed. Most 64-bit Linux distributions do not install any 32-bit libraries by default, so you'll have to install them manually. Below are some commands that might help you install the necessary libraries:

Debian/Ubuntu/Mint	CentOS/Fedora/OpenSUSE
<code>sudo apt-get install ia32-libs</code> <code>sudo apt-get install ia32-libs-multiarch</code>	<code>sudo yum install ia32-libs</code> <code>sudo yum install ia32-libs-multiarch</code>
<code>sudo dpkg --add-architecture i386</code> <code>sudo apt-get update</code> <code>sudo apt-get install ia32-libs-multiarch</code>	<code>sudo yum install glib2.i686 libgcc.i686</code> <code>libstdc++.i686</code>
<code>sudo apt-get install libc6:i386</code> <code>sudo apt-get install libc6-i386</code>	<code>sudo yum provides missinglibrarypathname</code> <code>sudo yum install packagenamewithlibrary</code>
<b>Ubuntu 14+:</b> <code>sudo dpkg --add-architecture i386</code> <code>sudo apt-get update</code> <code>sudo apt-get install libc6:i386 libncurses5:i386</code> <code>libstdc++6:i386 libglib2.0-0:i386 libsoup2.4-</code> <code>1:i386 libc6:i386 libgtk2.0-0:i386</code>	<code>/usr/bin/yum -y install glib2.i686 libgcc.i686</code> <code>libstdc++.i686</code>
<code>sudo apt-get install libgtk2.0-0:i386</code>	<code>yum install libc6.i686</code>

If you are still having trouble identifying necessary libraries, the `ldd` command might help. You can run from Terminal in the Xojo directory to return a list of libraries required by Xojo and their status on the system:

```
ldd Xojo
```

# Release Notes

You can always view release notes for the specific version of Xojo you are using by selecting **New In This Release** from the Xojo Help menu.

Visit [Xojo.com](http://Xojo.com) to [download](#) the latest version of Xojo.

<b>Release</b>	<b>Release Date</b>
<a href="#">2015 Release 3.1</a>	November 10, 2015
<a href="#">2015 Release 3</a>	October 20, 2015
<a href="#">2015 Release 2.4</a>	August 18, 2015
<a href="#">2015 Release 2.3</a>	August 11, 2015
<a href="#">2015 Release 2.2</a>	May 26, 2015
<a href="#">2015 Release 2.1</a>	April 22, 2015
<a href="#">2015 Release 2</a>	April 14, 2015
<a href="#">2015 Release 1</a>	February 17, 2015

[Older Release Notes](#)

[Legacy Release Notes](#)

# Xojo 2015r3.1 Release Notes

This release fixes important issues found in Xojo 2015r3.

## Bug Fixes

<a href="#">41222</a>	Build	Fixed a bug where generating code for 64-bit or ARM would fail on Windows but think it succeeded.
<a href="#">41372</a>	Build	IDE no longer raises an exception when compiling when there is a conflict between a type that would be import by a Using clause and a type in user code.
<a href="#">41255</a>	Compiler	Fixed an assertion that would occur when a script assigned to and read from a property defined in the context object.
<a href="#">41235</a>	Framework » Web	WebRadioGroups now work in 64-bit apps.
<a href="#">41324</a>	Framework » Web	Javascript error dialog appears in front of all controls again.
<a href="#">41390</a>	Framework » Web	Fixed a NilObjectException that could occur when Picture/WebPicture was used on a subclass of WebControl.
<a href="#">40982</a>	Framework » Windows	Fixed a regression with threads that could lead to failed assertions in RuntimeThread.cpp.
<a href="#">41292</a>	Framework » Windows	Calling ShowModal on a OpenFileDialog, SaveAsDialog, or SelectFolderDialog now presents the dialog as a modal window again (regression in 2015r3).
<a href="#">41343</a>	IDE » Database Editor	Creating a new SQLite database from Insert->Database->New SQLite Database, now works again.
<a href="#">41428</a>	IDE » FileIO	File Type Set UTIs are correctly retained by the IDE when saving Text projects.
<a href="#">41009</a>	IDE » Layout Editor	Dragging any of the subclasses of SSLSocket (HTTPSecureSocket, SMTPSecureSocket) that have folder item properties onto a layout no longer causes a compilation error.
<a href="#">41420</a>	IDE » Scripting	Using a build script correctly updates the build window caption.
12 Bug Fixes		

## Changes

<a href="#">41320</a>	IDE » Code Editor	Optimized menu and toolbar processing to improve performance of typing in the Code Editor.
1 Change		

# Xojo 2015r3 Release Notes

Xojo 2015 Release 3 is one of the largest releases in Xojo history. Major changes and new features include:

- 64-bit Desktop, Web and Console apps for OS X, Linux and Windows
- Raspberry Pi 2 (32-bit ARMv7 Linux) apps for desktop, web and console apps
- Web Drag and Drop support
- The ability to create bookmarks to lines of code
- iOSLabel now supports different line breaks
- You can now “Collect” all your project items into a single bundle to make distribution easy
- All-new Uniform Type Identifier editor for managing files used by your apps
- iOSContainerControl to create reusable iOS controls

Learn more about [2015 Release 3](#).

[Download Xojo 2015 Release 3](#) to get started making apps!

 With 2015r3, Xojo no longer supports building apps for [Windows XP](#).

## Bug Fixes

<a href="#">24010</a>	Build	If an external IDE script cannot be found the IDE will, when it tries to run that step, show a warning and cancel the build.
<a href="#">40209</a>	Build	ICNS files are written using PNGS for all sizes.
<a href="#">31855</a>	Compiler	Instead of asserting, an error is now reported if an enum contains a string value.
<a href="#">38837</a>	Compiler	Compiler: Integer constants without an explicit type are now treated as Integer instead of Int32.
<a href="#">39763</a>	Compiler	Fixed a crash in the compiler when empty parenthesis were used as an expression.
<a href="#">39765</a>	Compiler	Fixed a crash with enumerations that have an invalid underlying type.
<a href="#">39855</a>	Compiler	Fixed a crash that would occur when using Mod where the second operand was zero.
<a href="#">39994</a>	Compiler	Fixed a handful of crashes that could occur if a class inherited from itself.
<a href="#">39995</a>	Compiler	Fixed a crash when an invalid value was entered into the Implements field of the method editor.
<a href="#">40001</a>	Compiler	Fixed a crash when there was a method with multiple parameters marked Extends.
<a href="#">40078</a>	Compiler	Constants referring to other constants now resolve using the proper name lookup rules.
<a href="#">40079</a>	Compiler	Fixed a crash that could occur with enumerations in rare cases.
<a href="#">40086</a>	Compiler	Fixed a crash when a class has a superclass that implements something that is not an interface.

<a href="#">40087</a>	Compiler	Fixed the class interface aggregate cycle checking to be correct in all cases.
<a href="#">40095</a>	Compiler	Fixed a crash that could occur if the underlying type of an enumeration was an array.
<a href="#">40096</a>	Compiler	Fixed a crash that could occur if a parameter was specified to be ByRef and ParamArray.
<a href="#">40128</a>	Compiler	Fixed a crash that could occur when #If blocks had a conditional value that didn't evaluate to a boolean constant and there was an #Elseif.
<a href="#">40139</a>	Compiler	Fixed a failed assertion that could occur if a structure field had an unresolved type that happened to have the same name as another non-type symbol.
<a href="#">40342</a>	Compiler	Fixed the alignment of 64-bit integers inside of naturally aligned structures.
<a href="#">38863</a>	Crashes & Assertions » Crash	Compiler: Fixed a bug where applications would crash if an exception was raised through a very large function.
<a href="#">39076</a>	Crashes & Assertions » Crash	Fixed a crash that would occur if RuntimeExcerpton.Stack was accessed from a binary lacking function names.
<a href="#">27064</a>	Database Plugins	ODBC Plugin: Binding string of type ODBC_TYPE_LONGSTRING no longer truncates string to 2 characters.
<a href="#">35603</a>	Database Plugins	The DNS chooser dialog now shows up again on OS X when the ODBCDatabase.DataSource is empty.
<a href="#">40349</a>	Database Plugins	ODBC Plugin: Fixed a host of issues related to prepared statements: - Binding ODBC_TYPE_STRING/LONGSTRING now works properly - Binding empty strings now works properly - Fixed a potential memory buffer overrun with long sql statements, binding long strings, etc. - Improved error reporting for prepared statements - Updating Date values with Null is now possible
<a href="#">27807</a>	Debugger	Can resize splitters in debugger whether the IDE is paused while debugging or not.
<a href="#">30860</a>	Debugger	When starting the debugger the bottom pane, if open, will hide the bottom pane so the debugger can properly show.
<a href="#">32726</a>	Debugger	Console Remote debugger Stub can be used from an SSH system and does not require the OS X Terminal app or xterm.
<a href="#">34893</a>	Debugger	Running a web app no longer badges the IDE dock icon.
<a href="#">38668</a>	Debugger	Debugger pane can be resized vertically as well as horizontally even when not viewing debugging data.
<a href="#">39341</a>	Debugger	Properties exposed with Inspector Behavior (esp. Text) retain their values when set in the Inspector.
<a href="#">18420</a>	Framework » All	Email attachments save as expected regardless of being base64 encoded or not. Only base64 encoded are decoded before saving.
<a href="#">21325</a>	Framework » All	Variant's Int64Value now handles exponents.

<a href="#">34709</a>	Framework » All	HTTPSocket, HTTPSecureSocket, SMTPSocket, SMTPSecureSocket now support longer username/password combinations for proxies and http authentication.
<a href="#">34980</a>	Framework » All	WebListBox.LastIndex now defaults to -1 instead of 0.
<a href="#">38643</a>	Framework » All	EmailAttachment now adds filename (Latin1 and UTF8) parameter to Content-Disposition attachment header if the attachment has its name property set.
<a href="#">39274</a>	Framework » All	Xojo.Data.GenerateJSON no longer throws an ArgumentException when passed empty text.
<a href="#">39307</a>	Framework » All	Text variant comparisons with String, Integer, Double, etc. now works instead of raising a failed assertion.
<a href="#">39319</a>	Framework » All	An off by one error in the index of method for the new MemoryBlock has been fixed.
<a href="#">39331</a>	Framework » All	Int8 and UInt8 can be used with the Str function with and without format strings.
<a href="#">39398</a>	Framework » All	GenerateJSON no longer throws an InvalidFormatException when passed a zero-length String.
<a href="#">39516</a>	Framework » All	All: SMTP connections using STARTTLS have been refactored so that an intercepting insecure server cannot accept the connection.
<a href="#">39981</a>	Framework » All	Windows/Linux: Subtracting or adding a DateInterval to a Xojo.Core.Date now properly modifies the significant fields of the date properly.
<a href="#">31664</a>	Framework » Linux	Change the pace of the indeterminate progress bar to something a little slower.
<a href="#">38795</a>	Framework » Linux	Built apps now look in the generic "Libs" folder for plugins and shared libraries if available.
<a href="#">40259</a>	Framework » Linux	Using Xojo.Net.HTTPSocket in Console apps no longer crashes in Ubuntu 12.04 or older.
<a href="#">38659</a>	Framework » Macintosh (Cocoa)	RadioButtons on TabPanels and PagePanels have the pre-2014r2 behavior again.
<a href="#">40672</a>	Framework » Macintosh (Cocoa)	Fixed a crash that would occur when closing a fullscreen or split window on OS X 10.11.
<a href="#">15116</a>	Framework » Web	WebRadioGroups are now the correct height in the browser.
<a href="#">38655</a>	Framework » Web	Fixed a WebDialog refresh bug which caused WebLabels to render incorrectly when a dialog had been closed and opened again.
<a href="#">38880</a>	Framework » Web	WebCanvas now has a DisableDiffEngine property.
<a href="#">40125</a>	Framework » Web	Fixed a bug in WebListbox which made the use of external datasources using the DataCell interface fail. Added a RefreshCell method to allow pushing DataSource changes down to the browser.
<a href="#">40188</a>	Framework » Web	WebListBox.LastIndex now gets set to -1 when DeleteAllRows is called, matching the desktop behavior.

<a href="#">40242</a>	Framework » Web	Web: Fixed a bug which prevented IE11 and IE Edge from being detected properly.
<a href="#">40262</a>	Framework » Web	Firefox and Internet Explorer 9+ will now play H.264 MP4 videos in their native players instead of using Flash.
<a href="#">40268</a>	Framework » Web	Web: Fixed a bug which prevented gzip compression from being applied to http responses.
<a href="#">40309</a>	Framework » Web	Web: WebTextField and WebTextArea now fire the TextChanged event when the text set in the inspector is cleared.
<a href="#">40511</a>	Framework » Web	Internet Explorer 11 is recognized as a supported browser again.
<a href="#">39704</a>	Framework » Web » Frame	WebHTMLViewer.URL no longer destroys the backing webfile if the encoding of the URL being set is not UTF8 and that's the only difference.
<a href="#">40060</a>	Framework » Web » Frame	Closing a WebPage, WebDialog or WebContainer no longer shows an error about closing controls.
<a href="#">40048</a>	Framework » Web » Javascript	WebSDK no longer overwrites the XojoCustom namespace if it already exists.
<a href="#">20850</a>	Framework » Windows	Modernized Open/Save/Select dialogs (i.e. using Vista+ style dialogs instead of XP style). Note: setting the Cancel caption only works on Windows 7+.
<a href="#">24826</a>	Framework » Windows	Adding a movie to a project no longer autoplays it.
<a href="#">33565</a>	Framework » Windows	WebKit based HTMLViewer no longer increments the reference count of its CefBrowser when getting the Handle property.
<a href="#">34080</a>	Framework » Windows	The GDI+ graphics functions now falls back on a generic MS Sans Serif font if an unsupported font was used (this fixes various issues like StringWidth returning 0).
<a href="#">38693</a>	Framework » iOS	When you set the Value for a segmented control on iOS that item is now marked as selected so that other changes to the control, such as changing a caption, work as expected.
<a href="#">38972</a>	Framework » iOS	You can once again use the Browse option selecting an image for an image view. On iOS it creates an image asset, not a picture, the same as if you dragged an image into the project.
<a href="#">39500</a>	Framework » iOS	A segmented control with no segments no longer results in an error when you analyze the project.
<a href="#">40599</a>	Framework » iOS	Changes to some enumerations for consistency: * UIImageView.ContentsMode ENUMERATION renamed to be plural (ContentModes) * iOSProgressWheel.Shade ENUMERATION renamed to be plural (Shades) * iOSToolbar.Context ENUMERATION renamed to be plural (Contexts) * iOSDatePicker.DatePickerMode ENUMERATION renamed to iOSDatePicker.Modes (pluralized & renamed)

<a href="#">36874</a>	IDE	Adding an event handler puts the focus on that item in the Navigator, or the last one if you add several at once, and this results in the code editor having focus so you can type immediately.
<a href="#">38771</a>	IDE	Assets no longer silently reload if they have lost track of one of the images in one of the asset types.
<a href="#">39240</a>	IDE	An error that could arise when you copied and pasted a property has been fixed.
<a href="#">39430</a>	IDE	The feedback icon in the toolbar has been fixed.
<a href="#">40402</a>	IDE	Deleting an event from a control set no longer causes issues after deleting it.
<a href="#">40891</a>	IDE	Accented characters now work fine as project item names.
<a href="#">20233</a>	IDE » Auto Complete	Toolitems on a Toolbar now autocomplete.
<a href="#">36047</a>	IDE » Auto Complete	SQLiteDatabase on iOS no longer shows methods that apply to the non-iOS SQLiteDatabase in autocomplete.
<a href="#">36417</a>	IDE » Auto Complete	Items that are not accessible for the project type are not shown by autocomplete.
<a href="#">39125</a>	IDE » Auto Complete	Text constants are now shown in autocomplete.
<a href="#">39325</a>	IDE » Auto Complete	All sortable array types show Sort and SortWith as possible autocompletion methods.
<a href="#">39383</a>	IDE » Auto Complete	Autocomplete now shows a structure's ByteValue method.
<a href="#">39406</a>	IDE » Auto Complete	Until keyword now autocompletes properly. Do loops are also closed in a more contextually sensitive way : * If you start the loop with DO it will be closed with "loop until" * If you start the loop with "do until" it will be closed with "loop"
<a href="#">37246</a>	IDE » AutoLayout	Positioning controls in IOS using cursor keys no longer hangs the IDE.
<a href="#">38744</a>	IDE » AutoLayout	Baseline constraints are not removed when you resize horizontally as they were in the example. They get revised to a top constraint if the height is changed.
<a href="#">38777</a>	IDE » AutoLayout	Controls that have constraints that depended on a control that was being deleted were not be reset and would just be removed. This fix puts constraints back on to those affected controls.
<a href="#">39004</a>	IDE » AutoLayout	Can set baseline constraints and set them relative to other controls baselines.
<a href="#">19073</a>	IDE » Code Editor	The IDE no longer reformats method parameters when loading code.
<a href="#">25588</a>	IDE » Code Editor	Code editor is active immediately after adding event handler.
<a href="#">26363</a>	IDE » Code Editor	GoTo property with arrays now works correctly.
<a href="#">26837</a>	IDE » Code Editor	Adding an event puts the focus in the event's code editor.
<a href="#">27203</a>	IDE » Code Editor	Option-return on OS X once again inserts a line continuation character.

<a href="#">28670</a>	IDE » Code Editor	Enumerations only support the various integral types (integer, UInteger, int8, uint8, etc).
<a href="#">28688</a>	IDE » Code Editor	Constant name changes are property retained.
<a href="#">28942</a>	IDE » Code Editor	Clicking elsewhere retains changes as expected.
<a href="#">35320</a>	IDE » Code Editor	When you press shift + return there is always a new blank line added and the insertion point is on that new blank line.
<a href="#">38594</a>	IDE » Code Editor	Scroll offsets are accounted for when the code editor canvas is asked for the rectangle for some text so the special characters palette and dictionary show up in the right spot.
<a href="#">38651</a>	IDE » Code Editor	Selecting text and getting the OS X dictionary definition works again.
<a href="#">38847</a>	IDE » Code Editor	Attributes and compatibility settings are NOT retained on a control instance placed on a layout. This make iOS consistent with desktop applications.
<a href="#">39878</a>	IDE » Code Editor	Bug icons are centered using the same metrics as the hit boxes and line folding mechanisms.
<a href="#">40195</a>	IDE » Code Editor	Shift-return to close a block thats opened with a select case is now correctly closed with "end select".
<a href="#">40732</a>	IDE » Code Editor	Minor typing change in the short summary for constants properties and methods so they consistently use "As".
<a href="#">41039</a>	IDE » Code Editor	Help Syntax in Code Editor displays the correct signature for the item the mouse is hovering over.
<a href="#">40036</a>	IDE » Constant Editor	Dynamic constants of Text type now work.
<a href="#">40781</a>	IDE » FileIO	Make it so when the IDE loads an old Binary or XML project the unnamed items that are in the project get loaded in a way a user can remove them. This mimics what occurs for text projects.
<a href="#">30201</a>	IDE » Find & Replace	Capitalization on help tags for search fixed. Descriptions are accurate.
<a href="#">40404</a>	IDE » Find & Replace	Find->Replace All no longer drops a default value on a property on a class/window/etc where it used to.
<a href="#">40700</a>	IDE » Find & Replace	Finds using RegEx are shown correctly in the find panel.
<a href="#">21169</a>	IDE » Inspector	Windows IDE: Tabbing through Inspector fields now shows the focus ring on PopupMenus and other controls that have such visual styles on focus.
<a href="#">34942</a>	IDE » Inspector	Changing a description marks the project dirty so that it gets properly saved.
<a href="#">35355</a>	IDE » Inspector	Fixed a bug where there were visual 'glitches' in the top 22 to 78 points of the window when the window was full screen or split screen.
<a href="#">36878</a>	IDE » Inspector	The type field for method, enums, event definitions and properties no longer persistently shows the popup every time you press tab if you select on of the items from the list. A second tab will move to the next field.

<a href="#">38093</a>	IDE » Inspector	Fixed labelling of the automatic cursor type so it's consistent across all web controls.
<a href="#">38725</a>	IDE » Inspector	Inspector editors that use popups will only commit "changes" when you actually select a different item in the pop up. Selecting the same item no longer dirties the item and therefore the project.
<a href="#">39066</a>	IDE » Inspector	Text properties, like the default value for text on an iOS Label, update when the only change is case.
<a href="#">39351</a>	IDE » Inspector	Windows IDE: Fixed the random PopupMenus/ComboBox dropdown list widths in the IDE (for example the Scope Popup in the Inspector), mostly being way too wide.
<a href="#">39360</a>	IDE » Inspector	You can now change the super class for all members of a control array.
<a href="#">39393</a>	IDE » Inspector	A pasted font name "sticks" if it is one in the list (is available).
<a href="#">39491</a>	IDE » Inspector	Text properties on control instances are set properly.
<a href="#">39566</a>	IDE » Inspector	Made it so you can expose all kinds of integer properties that you could not before. The list now includes: * Integer, UInteger, Single, Double, Int8, Int16, Int32, Int64, UInt8, UInt16, UInt32, UInt64 * String, Text * Color * Boolean
<a href="#">40610</a>	IDE » Inspector	Control names changes are remembered when you click elsewhere or leave the name field in some other way.
<a href="#">40837</a>	IDE » Inspector	IDE no longer raises an exception when field has been pasted into and loses focus.
<a href="#">39092</a>	IDE » Language Reference	Reachability of the online wiki is no longer tested if you have the local reference selected as default.
<a href="#">25476</a>	IDE » Layout Editor	Checkboxes in the IDE draw in all 3 possible states.
<a href="#">35185</a>	IDE » Layout Editor	Applying a style to the toggled appearance of a web toolbar item now draws properly in the layout editor.
<a href="#">35245</a>	IDE » Layout Editor	On Windows if you have a control array and clear the name from one member you no longer end up with an endless loop of messages saying the name cannot be blank.
<a href="#">38354</a>	IDE » Layout Editor	Redrawing the web tool bar in the IDE is significantly faster.
<a href="#">38734</a>	IDE » Layout Editor	Locking iOS controls on a layout is not an option as it completely messes up the IDE's constraint solving.
<a href="#">38898</a>	IDE » Layout Editor	Auto-Layout delta offsets were being calculated incorrectly, sometimes causing a control to move on every click.
<a href="#">39058</a>	IDE » Layout Editor	The Layout Editor no longer hangs if the width of the iOSTextArea control is narrower than the longest word.
<a href="#">39061</a>	IDE » Layout Editor	Subclasses of iOS controls in the navigator drag like the super class does and when dropped are set up comparably.

<a href="#">39096</a>	IDE » Layout Editor	Lines shows drag / grab handles all the time now including when vertical or horizontal.
<a href="#">39107</a>	IDE » Layout Editor	Subclasses of iOS controls will not permit being resized in ways the base class cannot be resized.
<a href="#">39213</a>	IDE » Layout Editor	Disabled debug logs about baselines that are no longer needed.
<a href="#">39463</a>	IDE » Layout Editor	Controls in the Layout Editor Tray area no longer show top, left, width or height properties.
<a href="#">39622</a>	IDE » Layout Editor	It is now possible to set up a split with tabs in either the main or detail area on a screen that is not large enough to see the entire iOS screen.
<a href="#">39903</a>	IDE » Layout Editor	In the iOS Layout Editor, Labels are now drawn using System Bold or System Italic when specified.
<a href="#">40243</a>	IDE » Layout Editor	Clicking from one tab to another tab in an iOS screen layout no longer drops changes made when viewing other tabs.
<a href="#">40325</a>	IDE » Layout Editor	WebLabels with MultiLine=False and content which contains multiple lines now draw properly.
<a href="#">40345</a>	IDE » Layout Editor	A group of controls can now be resized by any of the grab handles on any control in the group, rather than just the first.
<a href="#">34696</a>	IDE » Library	Missing descriptions for Class, Interface, Module, Folder and generic object have been added.
<a href="#">39345</a>	IDE » Library	Windows library description pane redraws cleaner without visual artifacts.
<a href="#">26861</a>	IDE » Menu Editor	When editing a menu the index no longer shows -2147483648 in red (it shows nothing instead).
<a href="#">39494</a>	IDE » Menu Editor	Plain text projects that have menu items subclasses in modules so they load the same as binary or xml would. This makes it not only legal to put a menu item super class in a module for vcp but now it works too.
<a href="#">40330</a>	IDE » Menu Editor	You can set an icon on a submenu.
<a href="#">14392</a>	IDE » Miscellaneous	Notes are not printed in color.
<a href="#">14427</a>	IDE » Miscellaneous	A note that has _ as the very last character no longer causes issues when loading a text project.
<a href="#">16249</a>	IDE » Miscellaneous	External items that are not missing will get examined for items to resolve and get resolved the same as internal items do.
<a href="#">25072</a>	IDE » Miscellaneous	With the Project Chooser window open most menu items are disabled.
<a href="#">25306</a>	IDE » Miscellaneous	IDE will only error if in fact it cannot remove the damaged license key file.
<a href="#">25828</a>	IDE » Miscellaneous	Language Reference window retains tool bar setting from one open instance to another and run to run.

<a href="#">26311</a>	IDE » Miscellaneous	Changes in the case of a control name are retained. If a control is a member of a control set all members are updated.
<a href="#">26894</a>	IDE » Miscellaneous	If you open the icon editor for the app icon (any actually) and do nothing but press OK the project is not marked as having changes since nothing changed.
<a href="#">27035</a>	IDE » Miscellaneous	Progress bar no longer positioned oddly when starting a debug session.
<a href="#">27847</a>	IDE » Miscellaneous	Pressing enter in the file type editor no longer causes an exception.
<a href="#">28020</a>	IDE » Miscellaneous	Can define most integer property types for use with Inspector Behaviour.
<a href="#">28113</a>	IDE » Miscellaneous	More updates to text file saving. There are still some changes that will occur like updating editor types in the behaviors sections. These are because old text files left out information and opening a project may update them. The best way to force as many of these to occur at once so your subsequent source control commits are less noisy is to: 1) Make one change (add a space and remove it to one piece of code) 2) Do an Analyze Project and save 3) Commit with all changes and subsequent commits will have a lot less noise as the project has been updated
<a href="#">28402</a>	IDE » Miscellaneous	Tray-only controls do not show size and positioning properties in the Inspector.
<a href="#">29255</a>	IDE » Miscellaneous	Removing one member of a Control Set no longer removes the event list from the Navigator.
<a href="#">29511</a>	IDE » Miscellaneous	Projects of all kinds now print all their members.
<a href="#">34666</a>	IDE » Miscellaneous	Clarified the explanation on the preference setting for asking about overwriting files when building so it's clear this setting only applies when not using the build folders.
<a href="#">35005</a>	IDE » Miscellaneous	The insert menu no longer holds the wrong items when you switch from one project type to another.
<a href="#">37515</a>	IDE » Miscellaneous	Illegal characters in the name of the app to be debugged are stripped out (so the app CAN be debugged). Illegal characters in the name of the app to be built will cause a build to fail as the name is illegal on the target platform.
<a href="#">37963</a>	IDE » Miscellaneous	Autocomplete works in the local language reference.
<a href="#">38117</a>	IDE » Miscellaneous	You can no longer create multiple event definitions with the same name.
<a href="#">38124</a>	IDE » Miscellaneous	Copy Files build step computes relative paths correctly on all platforms. It may use an absolute path on Windows if the project and file are on different volumes.
<a href="#">38233</a>	IDE » Miscellaneous	Editing a File Type/UTI now marks the project as changed.
<a href="#">38267</a>	IDE » Miscellaneous	You cannot define a property that duplicates the name of a an existing const property or event.
<a href="#">38536</a>	IDE » Miscellaneous	Switching the iOS device also sets the architecture popup accordingly.
<a href="#">38742</a>	IDE » Miscellaneous	An NilObjectException that could occur when executing a compound action has been fixed.

<a href="#">38820</a>	IDE » Miscellaneous	Event handlers are no longer silently dropped from Text project format.
<a href="#">38891</a>	IDE » Miscellaneous	IDE not handling the new targets caused it to not grab the right built application name and assert.
<a href="#">38952</a>	IDE » Miscellaneous	An exception that could occur when selecting one of the build settings and trying to print has been fixed. Doing this now will print the entire project.
<a href="#">38974</a>	IDE » Miscellaneous	Images should now save and reload from partial paths that can be preserved cross platform and even between machines.
<a href="#">39177</a>	IDE » Miscellaneous	Adding multiple using clauses to a project item will name them with unique names.
<a href="#">39242</a>	IDE » Miscellaneous	A NilObjectException that can occur using the goto location panel and navigating into a menu has been fixed.
<a href="#">39249</a>	IDE » Miscellaneous	Fixed a situation where if you used floating palettes then the IDE would be resized not based on what palettes were visible when a new project was created but based on the size the palettes would have if they were visible.
<a href="#">39263</a>	IDE » Miscellaneous	Fixed a bug that could change the type of a constant from integer to blank when you clicked on it.
<a href="#">39298</a>	IDE » Miscellaneous	The IDE no longer causes an exception when the size of columns uses measurements that do not work for a desktop listbox. For example, measurements like 60px (which work with WebListBox) do not work for the desktop.
<a href="#">39402</a>	IDE » Miscellaneous	Updated the Property/Text icon in the Navigator with a slightly thicker T.
<a href="#">39427</a>	IDE » Miscellaneous	String property editor recognizes changes that are from other means like paste and drag and drop and commits those changes properly.
<a href="#">39506</a>	IDE » Miscellaneous	Cleaned up extraneous separator lines in the View and Window menus on Windows and Linux.
<a href="#">39514</a>	IDE » Miscellaneous	An illegal cast exception when adding a computed property with a name that duplicated that of an existing pair of methods has been fixed.
<a href="#">39559</a>	IDE » Miscellaneous	Descriptions on notes in plain text format save and restore as expected.
<a href="#">39560</a>	IDE » Miscellaneous	An old oversight that would permit ',' and '=' in the name of the note has been corrected so they can be used without causing further issues.
<a href="#">39676</a>	IDE » Miscellaneous	"Text as String" in a plugin parameter list no longer gets renamed.
<a href="#">39682</a>	IDE » Miscellaneous	Closing the last altered project window on Windows when you have the LR open no longer asks you twice if the "When closing the last window quit" option is enabled. Closing the last altered project on OS X no longer closes the LR as well since it handles the app running when no documents open differently than Windows.
<a href="#">39699</a>	IDE » Miscellaneous	Trailing spaces on lines are not stripped.

<a href="#">39712</a>	IDE » Miscellaneous	Changes to web styles save properly in text projects.
<a href="#">39738</a>	IDE » Miscellaneous	Selecting Find Implementors from the contextual menu of an Interface no longer does a second search for the Interface name.
<a href="#">39784</a>	IDE » Miscellaneous	Add event window has a useful minimum size.
<a href="#">39877</a>	IDE » Miscellaneous	Added shortcut key for Analyze Item on Windows and Linux (CONTROL-SHIFT-K).
<a href="#">39883</a>	IDE » Miscellaneous	Using the IDE scripting editor no longer causes a NilObjectException.
<a href="#">39953</a>	IDE » Miscellaneous	You now get a warning when plist files in the project cannot be found.
<a href="#">40020</a>	IDE » Miscellaneous	When quitting the IDE, each open project with unsaved changes is prompted to save, rather than just the first one with unsaved changes.
<a href="#">40074</a>	IDE » Miscellaneous	Can no longer insert a container control into a console project.
<a href="#">40075</a>	IDE » Miscellaneous	iOS projects can no longer have databases inserted into them.
<a href="#">40103</a>	IDE » Miscellaneous	Remote Debug Hosts list no longer beeps when you delete a remote host using the keyboard.
<a href="#">40104</a>	IDE » Miscellaneous	Preferences for setting the Windows AutoQuit functionality is now limited to running on Windows.
<a href="#">40127</a>	IDE » Miscellaneous	iOS projects also require a bundle ID.
<a href="#">40151</a>	IDE » Miscellaneous	Adding an image to a project named "backdrop" (or any other Window property) and trying to assign it to a control no longer causes compile errors or unexpected runtime behaviour.
<a href="#">40161</a>	IDE » Miscellaneous	iOS projects no longer incorrectly show error about enum type name.
<a href="#">40184</a>	IDE » Miscellaneous	A NilObjectException while determining the folder to cache intermediate code in no longer causes the IDE to assert.
<a href="#">40197</a>	IDE » Miscellaneous	Updated menu items which point to the New & Classic Framework references.
<a href="#">40214</a>	IDE » Miscellaneous	Fixed a NilObjectException in the IDE when selecting the Constants group row in an external item.
<a href="#">40274</a>	IDE » Miscellaneous	On OS X if the volume a project is loaded from goes offline when you have the project open the IDE won't assert when you try to save it. Windows and Linux didn't experience this particular assertion.
<a href="#">40338</a>	IDE » Miscellaneous	No longer raises an OutOfBoundsException when you make control sizes really large - and it will actually clamp the size to the maximum (32767).
<a href="#">40354</a>	IDE » Miscellaneous	A missing external item no longer causes an OutOfBoundsException if it's missing when the project is reopened.
<a href="#">40369</a>	IDE » Miscellaneous	An OutOfBoundsException in the code editor has been fixed.

<a href="#">40401</a>	IDE » Miscellaneous	Restore the behaviour where the first paste of an external item would paste an external (as long as no other external item with that name already existed) and subsequent ones would bring the item in as internal copies. It does this so we avoid have many external items all pointed at the same file on disk (which would be confusing as heck and cause other problems).
<a href="#">40417</a>	IDE » Miscellaneous	Warnings in encrypted items are again ignored. This was a regression from 2015r2.2.
<a href="#">40673</a>	IDE » Miscellaneous	Enter Full Screen no longer shows up twice on OS X 10.11.
<a href="#">40674</a>	IDE » Miscellaneous	Windows in the layout editor don't draw disabled on OS X 10.11.
<a href="#">40817</a>	IDE » Miscellaneous	IDE no longer runs out of memory when there is a very long list of missing items to resolve.
<a href="#">40883</a>	IDE » Miscellaneous	Several spots in the IDE that were not looking up the dynamic constant have been fixed so they appear closer to how they will at runtime.
<a href="#">40903</a>	IDE » Miscellaneous	Make it so a declare like STATIC foo is treated similarly to DIM and doesn't strip out empty brackets.
<a href="#">40907</a>	IDE » Miscellaneous	UTI's that an app accepts are NOT marked as exported any longer (which was horribly wrong).
<a href="#">40981</a>	IDE » Miscellaneous	A NilObjectException that could occur when trying to replace a property value on an object on a layout no longer occurs. Such a change isn't actually permitted and the replace will fail . You have to change the name of the property on the base class.
<a href="#">41049</a>	IDE » Miscellaneous	WebToolbars that are smaller than the total width of all items on them no longer cause errors when trying to edit them.
<a href="#">27023</a>	IDE » Navigator	Controls in arrays sort lexically by name then by control array index. Controls that have been duplicated many times will sort similarly by the base name then the trailing numeric portion.
<a href="#">32340</a>	IDE » Navigator	The command bar + (right above the editor space) will properly update to allow or disallow the "Event handler" item in its menu when you change the super of a class selected in the Navigator.
<a href="#">37499</a>	IDE » Navigator	Cmd-double-click on an array property in the code editor will now jump to its declaration.
<a href="#">38683</a>	IDE » Navigator	Class Interfaces can no longer be dragged onto an iOS layout.
<a href="#">38792</a>	IDE » Navigator	Filtering of items now properly includes external items.
<a href="#">38793</a>	IDE » Navigator	Navigator contextual popup menu appears in the right spot.
<a href="#">39407</a>	IDE » Navigator	Updated the Pointer icon with one that's more consistent with the other property icons.
<a href="#">39565</a>	IDE » Navigator	Make it so all integer property types have integer icons (missing were the 8 and 16 bit sizes).

<a href="#">39627</a>	IDE » Navigator	When you delete an event handler from a control, searching for text contained in that event handler no longer shows that event.
<a href="#">39666</a>	IDE » Navigator	WebImageView, WebButton and WebRadioGroup icons now show properly in the Navigator.
<a href="#">40999</a>	IDE » Profiler	Profile data is no longer lost as profile editors close.
<a href="#">26415</a>	IDE » Project Editor	Items that are multiselectd have their property lists properly merged to show only those properties that are common to all selected items
<a href="#">35228</a>	IDE » Project Editor	Changes to web styles save properly with Text projects.
<a href="#">39302</a>	IDE » Rendering	Using an enum in the Inspector Behaviour for Web controls functions properly.
<a href="#">39689</a>	IDE » Rendering	No longer creates a warning for a property named NotifyControls for WebPages that do not have any WebControl subclasses on them.
<a href="#">40100</a>	IDE » Rendering	No longer get a compile error when adding a Xojo.Core.Timer to a web page.
<a href="#">39970</a>	IDE » Report Editor	Using a picture with the same name as a report segment (header, footer) or report property no longer causes an error when trying to run or compile.
<a href="#">39842</a>	IDE » Scripting	Autocomplete in an IDE script no longer causes an exception.
<a href="#">40130</a>	IDE » Scripting	IDECommunication IPC Socket now closes properly every time the IDE exits.
<a href="#">40344</a>	IDE » Scripting	Windows Copy Files Steps create any required subdirectories as expected and handle both LFS (C:\ style) and UNC paths (\\server\path style).
<a href="#">15542</a>	IDE » Style Editor	Updated the web style color picker values titles to # and % since you pick either a value (#) or a percentage (%). This replaces the "Hex" title which is misleading as you only entered hex in one portion of the editor when editing values.
<a href="#">25283</a>	IDE » Style Editor	Can edit the border size field using arrow keys, delete, backspace, etc.
<a href="#">40925</a>	IDE » Style Editor	Style Editor refreshes and doesn't wipe out colors as set.
<a href="#">39562</a>	IDE » Updater	Labels on the update window are now the same color as the background on Windows and Linux.
<a href="#">23428</a>	IDE » Web Page Editor	CueText on WebSearchField shows correct localized value in the IDE.
<a href="#">26173</a>	IDE » Web Page Editor	Changes made using the popover are properly retained so they work with Undo and mark the project as modified.
<a href="#">27153</a>	IDE » Web Page Editor	The inline editor for a listbox can scroll columns horizontally (so if you have a lot of them you can get to them all). The editor doesn't try to exactly mimic the column widths you have set up as this can result in columns that cannot be edited. If the column is > 60 pixels wide we use it as set but if < 60 we force the editor's representation to be 60 pixels so you can edit every column.

<a href="#">37783</a>	IDE » Web Page Editor	Accessing a page with a toolbar and an item on it no longer causes the project to be marked as changed despite nothing having changed.
<a href="#">24062</a>	IDE » Window Editor	If you delete an image that's used as a window background, the window redraws without the image as expected.
<a href="#">25512</a>	IDE » Window Editor	Popovers have a close icon on them.
<a href="#">25581</a>	IDE » Window Editor	Scrollers redraw themselves in the correct orientation as they are resized in the IDE.
<a href="#">27269</a>	IDE » Window Editor	If you create a popup menu and set the initial value to a bunch of #constant style values the IDE will draw the constants correctly and not show "#constant" instead.
<a href="#">29175</a>	IDE » Window Editor	When multiple items are selected the position properties (top, left, width & height) items will show as "Multiple" if several are selected. Tabbing into / out of such an item has no effect. If you select and change the value it will take effect.
<a href="#">30091</a>	IDE » Window Editor	Containers have consistent property groupings in the Inspector whether it's a subclass or instance on a window.
<a href="#">21100</a>	Plugin SDK	REALstandardGetter and REALstandardSetter work with all intrinsic types.
<a href="#">38589</a>	Plugin SDK	Fixed a bug where REALGetPropValueObject would return false even though it succeeded.
<a href="#">38840</a>	Plugin SDK	REALLockText and REALUnlockText now function properly.
<a href="#">39849</a>	Plugin SDK	Fixed the currency prop value getters and setters.
<a href="#">40351</a>	Plugin SDK	Shared Methods and Properties now work with plugin controls, instead of causing a runtime error.
<a href="#">31649</a>	Remote Debugger Stub	Console Remote debugger Stub can be used from an SSH system and does not require the OS X Terminal app or xterm.
<a href="#">33678</a>	Remote Debugger Stub	Console Remote debugger Stub can be used from an SSH system and does not require the OS X Terminal app or xterm.
<a href="#">34560</a>	Reporting	Round Rectangles on reports no longer appear as ovals.
259 Bug Fixes		

## Changes

<a href="#">6231</a>	Build	Windows/Linux: built apps that include resources now creates a Resources folder which includes the app name in its folder name, just like the Libs folder. You can also rename it to just "Resources" after the fact, and the framework will find it just like the Libs folder.
<a href="#">40144</a>	Build	Using Xcode 7 and its iOS Simulator is supported now.

<a href="#">18982</a>	Compiler	The Inline68k keyword has been removed. This has done nothing since support for compiling Mac 68k code was dropped and existed only for backwards compatibility.
<a href="#">38391</a>	Compiler	Upgraded to LLVM 3.6.
<a href="#">38731</a>	Compiler	The compiler can now build for 64-bit executables for Windows and OS X.
<a href="#">39414</a>	Compiler	The Linux IDE can now build for Linux x86-64 and Linux ARM.
<a href="#">39578</a>	Compiler	Deprecated the WindowPtr and Short data types.
<a href="#">40239</a>	Compiler	The version of LLVM used by XojoScript is now LLVM 3.7.
<a href="#">41051</a>	Compiler	The compiler no longer creates an empty libs folder if there are no plugins present.
<a href="#">41052</a>	Compiler	Declares to non-system libraries work now.
<a href="#">38781</a>	Framework » All	Removed the deprecated and non-functional OpenREALDatabaseOldFormat function.
<a href="#">39292</a>	Framework » All	The socket on Windows has been updated to use Winsock2.
<a href="#">370</a>	Framework » Linux	Reduced thread CPU usage on Linux for mostly idle threads. This difference will most likely be visible on single core processors, but this fix also benefits multi-core and other platforms.
<a href="#">39533</a>	Framework » Linux	Ticks is now guaranteed to be a monotonic clock.
<a href="#">31751</a>	Framework » Web	Converted web framework to use the new Xojo framework JSON methods.
<a href="#">39513</a>	Framework » Web	WebSDK controls can now be integrated into the Xojo Drag and Drop system.
<a href="#">39792</a>	Framework » Web	Internet Explorer 8 has been removed from the list of supported browsers as it limits our ability to provide updated capabilities and security patches.
<a href="#">39797</a>	Framework » Web	Minimum browser requirements have been updated to: Safari 6.1+ Firefox 17+ Chrome 37+ IE9+
<a href="#">40002</a>	Framework » Web	Added support for detecting the IE Edge browser using WebSession.EngineType.EdgeHTML enumeration.
<a href="#">40731</a>	Framework » Web	Updated web framework to allow the Raspberry Pi Epiphany Web Browser.
<a href="#">39838</a>	Framework » Web » Frame	The XojoCloud Firewall class is now available in Linux Console and Linux Web Apps so users can create helper apps which have the power to open/close firewall ports if necessary.
<a href="#">38265</a>	Framework » Windows	Message logging has had speed improvements: 1) a queue of messages has been added so the app being debugged can write at full speed and the IDE messages pane will be updated from that instead of trying to keep up in real time. 2) the scrollbar area only retains the last 5000 items at most 3) the queue may be flushed out if it gets more than 1000 messages behind

<a href="#">39769</a>	Framework » iOS	iOSToolButton.Type enum has been renamed to iOSToolButton.Types (note the extra 'S'). This will break existing code that uses this enum.
<a href="#">39396</a>	IDE	Currency has its own special icon in the navigator.
<a href="#">39446</a>	IDE	The Mac Creator code is not used by anything on OS X since 10.6 / 10.7. It will be preserved if set but is not used when compiling the application.
<a href="#">38553</a>	IDE » AutoLayout	It is now possible to set a control's width = its height and vice versa. Note that you can cause yourself a lot of problems should you set width = height and height = width, so don't do that.
<a href="#">29858</a>	IDE » Code Editor	You can drag a control instance into the code editor from the navigator and the name of the control will be inserted at the cursor position.
<a href="#">40017</a>	IDE » Code Editor	&u literals are now colorized similarly to string/text literals.
<a href="#">40986</a>	IDE » Code Editor	Added a preference that allows people to turn autocomplete off.
<a href="#">33475</a>	IDE » Layout Editor	The description area below the library can now be scrolled.
<a href="#">36330</a>	IDE » Layout Editor	Grab handles are now on control frames instead of outset slightly.
<a href="#">38897</a>	IDE » Licensing	Dates are shown like Aug 5, 2015 instead of the ambiguous numeric form.
<a href="#">40522</a>	IDE » Licensing	Changed caption of the download button on the license pane of the About window to Update.
<a href="#">19</a>	IDE » Miscellaneous	The replacement of the old File Type set with the UTI pane removes the long deprecated File Type from the editor. OS Type can be set (for legacy purposes) but its use is strongly discouraged by Apple.
<a href="#">19961</a>	IDE » Miscellaneous	Main toolbar now has tooltips (OS X only).
<a href="#">37596</a>	IDE » Miscellaneous	Compatibility flags allow for setting of project type and 32 / 64 bit in the Inspector.
<a href="#">38636</a>	IDE » Miscellaneous	Analyze one item now has its old Cmd+Shift+K shortcut (ctrl+shift+K on Windows and Linux).
<a href="#">38722</a>	IDE » Miscellaneous	Made it so if you have an item open as the top level item in a tab and you delete it in another tab, the open tabs showing it are closed.
<a href="#">38762</a>	IDE » Miscellaneous	Analysis warning for name lookup differences is disabled in new projects by default.
<a href="#">38802</a>	IDE » Miscellaneous	Feedback report bodies are started with a line that says what version of the IDE is in use. This is the same string used in the about window.
<a href="#">38807</a>	IDE » Miscellaneous	Web projects have architecture setting for OS X targets.
<a href="#">39022</a>	IDE » Miscellaneous	Cocoa x86_64 is now an allowed build script build target.
<a href="#">39567</a>	IDE » Miscellaneous	Byte and short now have proper icons based on their type (integer) and can be used as properties that are exposed to the inspector behavior and will be set correctly using the mass property setter.

<a href="#">39667</a>	IDE » Miscellaneous	Project Chooser Dialog caption updated with more descriptive text. Button captions reverted back to OK and Cancel
<a href="#">39679</a>	IDE » Miscellaneous	Added iOS QuickStart to the Help menu.
<a href="#">39821</a>	IDE » Miscellaneous	Project > Go To Search was never used so it has been removed.
<a href="#">40061</a>	IDE » Miscellaneous	Missing file dialog list is now multiselectable.
<a href="#">40064</a>	IDE » Miscellaneous	Binary projects now specifically use 32 bit sized values.
<a href="#">38971</a>	IDE » Navigator	Generic icon in the Navigator now has a blue cube instead of a green one.
<a href="#">39780</a>	IDE » Scripting	Added new IDE Script BuildApp constants (for iOS Universal app, 64-bit).
<a href="#">39795</a>	IDE » Scripting	If you select an external Objective-c method and use the IDE script command to report its type you will get "External Objective-C Method" returned.
<a href="#">40131</a>	IDE » Scripting	IDE: App Nap is now disabled while the IDECommunicator is connected.
<a href="#">40494</a>	IDE » Style Editor	Adding a component to a web style scrolls to the newly added item.
<a href="#">40105</a>	Miscellaneous	Windows Setup file now includes a correct File Version info, instead of 0.0.0.0.
54 Changes		

## New Items

<a href="#">37112</a>	Build	Build folder names have been updated. Existing ones have not been changed to avoid breaking existing build scripts The names are now: Linux Linux 64 bit << new Linux ARM << new Mac OS X (Intel) Mac OS X (Cocoa Intel) OS X 64 bit << new Windows Windows 64 bit << new
<a href="#">39526</a>	Compiler	The compiler now issues an error when calling Objective-C declares on non-iOS or OS X platforms.
<a href="#">40490</a>	Compiler	The desktop framework is now supported for ARM.
<a href="#">2211</a>	Framework » All	All non-structure single dimension arrays now have an overload of the Sort function that can be used to customize the sort order.
<a href="#">40724</a>	Framework » Linux	Can now build desktop app for Raspberry Pi 2 (Linux ARMv7).
<a href="#">20384</a>	Framework » Web	Web Framework now supports drag and drop.
<a href="#">32859</a>	Framework » Web	Web: WebCanvas now has a DisableDiffEngine property.
<a href="#">37892</a>	Framework » Web	Added new property AcceptingConnections as Boolean which allows you to Start/Stop a web app's ability to accept connections. Setting this property to False will stop the server(s) and disconnect all active sockets immediately.
<a href="#">39440</a>	Framework » Web	WebCanvas now has a DisableDiffEngine property.
<a href="#">40269</a>	Framework » Web	Web: Re-enabled gzip compression on images for browsers that support it.

<a href="#">39791</a>	Framework » Web » Frame	Standalone SSL web apps now use TLSv1.2.
<a href="#">36581</a>	Framework » iOS	iOSLabel now has a variety of clipping modes that can be used for times when you want clipping instead of wrapping.
<a href="#">38901</a>	Framework » iOS	Added iOS Container Controls (iOSContainerControl).
<a href="#">14820</a>	IDE » Code Editor	The external method editor can now define an external Objective-C method and its associated selector.
<a href="#">33898</a>	IDE » Code Editor	Added bookmarking with project-specific bookmarks that work similarly to breakpoints.
<a href="#">28875</a>	IDE » Find & Replace	Added a View->Hide Find and shortcut for toggling the Find/Errors/Messages panel (Shift-Command-F, Shift-Control-F).
<a href="#">3499</a>	IDE » Miscellaneous	Collect Project Items - will collect the items next to a SAVED project - if it's not saved it will do nothing (wont be enabled) - items remain external - items will be grouped (pictures, data, scripts etc) - the collection of item will NOT rename items on disk so duplicates will cause collection to report errors that are visible in the errors pane IF you do NOT want it to modify the project you have open do a "Save As" first then Collect them into the new copy. Collection DOES dirty the project but does NOT automatically save it.
<a href="#">30538</a>	IDE » Miscellaneous	When you run your project, the bottom pane (Find, Errors, Messages) will close if it is open so that the debugger has more room for its display.
<a href="#">38348</a>	IDE » Miscellaneous	For iOS projects dragging a picture into the project will by default create an image asset with the picture set as the 1x version. Holding the option key down will still import the picture as a Picture project item.
<a href="#">38593</a>	IDE » Miscellaneous	File Types Set/UTI editor permits specifying importable and exportable.
<a href="#">38737</a>	IDE » Miscellaneous	iOS applications now have a full set of capabilities editors to make it easy to add GameCenter, etc to the entitlements needed (in the Advanced Inspector tab for the iOS Build Settings).
<a href="#">39476</a>	IDE » Miscellaneous	ICNS files are not written out for non-OS X targets.
<a href="#">11196</a>	IDE » Project Editor	The File Types Editor has undergone an enormous overhaul and now lets users on all platforms set up and manage UTI's (which also happens to work for windows file types and Linux since they require a subset of the data a UTI provides).
<a href="#">39488</a>	IDE » Scripting	Added docs for new IDE Build Targets for 64-bit and Pi.
<a href="#">40129</a>	IDE » Scripting	Added an EnvironmentVariable method to the IDE scripting context for getting those values from the environment in which the IDE was launched.
<a href="#">40704</a>	IDE » Scripting	IDE Script supports NewIOSProject command.
<a href="#">41036</a>	IDE » Warnings Panel	For certain kinds of errors (right now it's ONLY linker errors) you can right click and copy the error message since it MAY be many lines long

<a href="#">39408</a>	Miscellaneous	Added Project-Deploy Application to the menu for deploying Xojo Cloud apps (Command-Shift-B, Control-Shift-B).
<a href="#">38599</a>	Plugin SDK	Plugins SDK: The size of plugin data structures is no longer required to be the same across all platforms. Also, the offsets of members in structures that are used with REALStandardGetter or REALStandardSetter can now vary across platforms.
29 New Items		

## Docs and Examples

<a href="#">39596</a>	Documentation	Added “See Also” sections to better relate WebApplication.Timeout, WebApplication.SessionTimeout, WebSession.Timeout and WebSession.TimedOut.
<a href="#">40174</a>	Documentation	Xojo.Introspection classes doc pages now show correct full name prefix of “Xojo.Introspection”.
<a href="#">40228</a>	Documentation	Updated ISO links on Xojo.Core.Locale page.
<a href="#">40254</a>	Documentation	Added Return (keyed &h24) to Keyboard page.
<a href="#">40273</a>	Documentation	Added BottomLayoutGuide and TopLayoutGuide to iOSView properties.
<a href="#">40280</a>	Documentation	Removed incorrect note on SQLiteDatabase.CreateBlob about read-only blobs.
<a href="#">40514</a>	Documentation	Xojo.Introspection.ConstructorInfo page now includes Parameters method.
<a href="#">39195</a>	Documentation » User's Guide	Profiling information is only gathered when an app quits normally. Added additional information regarding this for web apps to User Guide Book 3: Framework, Section 8.8: Debugging and Profiling.
<a href="#">39254</a>	Documentation » User's Guide	Corrected SQL typo in Framework Guide, section 10.1 (Retrieving Data from a Database).
<a href="#">38269</a>	Examples	Examples/Xojo Framework/UsingExample works again.
<a href="#">38909</a>	Examples	For iOS AlertSheet example, changed Libs from “Cocoa” to “Foundation.Framework”.
<a href="#">38978</a>	Examples	Examples/iOS/Speak updated to use AVFoundation library in Declare. And speech now works in iOS 8.3 Simulator.
<a href="#">39420</a>	Examples	In EEWeb example, added back missing DB file to Copy Files Step for Xojo Cloud.
<a href="#">39591</a>	Examples	Added Examples/Communication/Serial/Serial Bar Code Reader Example that shows how to read values from a barcode scanner using the Serial class.
<a href="#">39592</a>	Examples	Added Space Rocks game example for desktop and iOS (Examples/Games/SpaceRocks).

<a href="#">15856</a>	Language Reference » Documentation	Updated System.NetworkInterfaceCount to note that it only counts connected interfaces.
<a href="#">16717</a>	Language Reference » Documentation	RegexMatch.SubexpressionString and SubexpressionStringB were incorrectly listed as properties instead of methods.
<a href="#">25657</a>	Language Reference » Documentation	Added docs for DataSource and DataCell interfaces used by WebListBox.
<a href="#">30858</a>	Language Reference » Documentation	Searching for items with special URLs (like &b) results in a search page where the links actually work.
<a href="#">38833</a>	Language Reference » Documentation	Int64 number range is now listed as -9,223,372,036,854,775,808 to +9,223,372,036,854,775,807.
<a href="#">38860</a>	Language Reference » Documentation	Xojo.Core.Date.Nanosecond property is no longer listed in docs as Nanoseconds.
<a href="#">38884</a>	Language Reference » Documentation	Fixed typo in SMTPSocket example code.
<a href="#">38893</a>	Language Reference » Documentation	Added missing "Using" clause to HTTPSocket.SetRequestContent sample code.
<a href="#">38959</a>	Language Reference » Documentation	Corrected typos on Xojo.Math page.
<a href="#">38984</a>	Language Reference » Documentation	BinaryStream sample code no longer uses ReadShort.
<a href="#">39005</a>	Language Reference » Documentation	Added Constructor to Xojo.IO.FolderItem page.
<a href="#">39042</a>	Language Reference » Documentation	Corrected text formatting for iOSView.Active/Deactivate events.
<a href="#">39262</a>	Language Reference » Documentation	Added Text data type to VarType page.
<a href="#">39278</a>	Language Reference » Documentation	Updated links to Microsoft Office Developer docs on ExcelApplication, WordApplication and PowerPointApplication pages.
<a href="#">39481</a>	Language Reference » Documentation	Removed references to OS X Carbon on SegmentedControlItem.Title and SegmentedControlItem.Icon.
<a href="#">39507</a>	Language Reference » Documentation	Updated IDEScripting PDF with latest info from 2015r2.2.
<a href="#">39654</a>	Language Reference » Documentation	WebApplication.SSLPort page is a read-only property, not a design-time property.
<a href="#">39773</a>	Language Reference » Documentation	Added additional note to Timer to indicate that it runs in the main thread.

<a href="#">39774</a>	Language Reference » Documentation	Added WebThread page.
<a href="#">39836</a>	Language Reference » Documentation	Pages for Const command also show ability to specify the type.
<a href="#">39908</a>	Language Reference » Documentation	Updated Currency data type pages to indicate precise range of values it supports.
<a href="#">40021</a>	Language Reference » Documentation	Corrected sample code for Xojo.Crypto.PBKDF2.
<a href="#">40116</a>	Language Reference » Documentation	Added a page to link to dialog-related pages when searching for “dialog” in wiki.
<a href="#">40334</a>	Language Reference » Documentation	Added missing IndexOf method to Xojo.Core.MemoryBlock page.
<a href="#">40450</a>	Language Reference » Documentation	Added additional information about Priority to Thread page.
<a href="#">40454</a>	Language Reference » Documentation	Corrected type in SQLiteDatabase.LastRowID example to Int64.
<a href="#">40460</a>	Language Reference » Documentation	Declare page in wiki now has correct bold/italics for keywords and parameters.
<a href="#">40692</a>	Language Reference » Documentation	Added a warning to IPCSocket and Mutex pages about possible name conflicts when same temp folder is used.
43 Docs and Examples		

# Xojo 2015r2.4 Release Notes

This release fixes an important issue found in 2015r2.3.

## Bug Fixes

<a href="#">40053</a>	Framework » Linux	LibICU now properly loads for apps run on some distros with a minor version number attached (examples include: OpenSuse, Ubuntu 64-bit, CentOS 7).
1 Bug Fixes		
1 total items		

# Xojo 2015r2.3 Release Notes

This release fixes a few important issues found in 2015r2.

## Bug Fixes

<a href="#">39215</a>	Framework » All	DataControl no longer raises a failed assertion when the control is destructed.
<a href="#">39767</a>	Framework » All	Fixed Auto's handling of opaque colors so that it no longer corrupts the information.
<a href="#">40034</a>	Framework » All	Xojo.Introspection.PropertyInfo.Value's getter now works correctly on iOS.
<a href="#">40053</a>	Framework » Linux	No longer asserts when app is run on a distro that includes an ICU library with a minor revision number attached (like OpenSUSE 13.1).
<a href="#">28667</a>	Framework » Macintosh (Cocoa)	Fixed a crash that could occur if a control's width or height managed to become less than or equal to zero.
<a href="#">39818</a>	IDE » Xojo Cloud Uploader	Fixed a bug in the Xojo Cloud uploader which caused uploads to hang at the last moment.
6 Bug Fixes		
6 total items		

# Xojo 2015r2.2 Release Notes

This release fixes a few important issues found in 2015r2.

## Bug Fixes

<a href="#">39201</a>	Compiler	A mixture of UI and non-UI controls no longer causes a situation where an additional UI control would result in some non-UI controls being rendered to the compiler.
<a href="#">39220</a>	Database Engine	SQLiteDatabase: Binding Text values to PreparedStatement no longer inserts incorrect/random data.
<a href="#">39232</a>	Debugger	Fixed object IDs in the debugger being identical.
<a href="#">39147</a>	IDE » Miscellaneous	Fixed an error where it was possible to have a change to a text (VCP) project not saved when editing an event handler.
<a href="#">39185</a>	IDE » Miscellaneous	Fixed an IDE crash that could sometimes occur when deleting a ContainerControl from the project.
<a href="#">39237</a>	Web » Frame	Web: Fixed a regression where App.Port and App.SecurePort returned the IDE values when queried in the App.Open event of a Standalone app that had the --port or --secureport property set on the command line.
6 Bug Fixes		
6 total items		

# Xojo 2015r2.1 Release Notes

This release fixes a few important issues found in 2015r2.

## Bug Fixes

<a href="#">38979</a>	Compiler	Error dialog no longer blinks into existence and then out on Windows.
<a href="#">38928</a>	Compiler	Fixed a failed assertion that occurred when building a project that had a database in it.
<a href="#">39020</a>	Framework » iOS	All: Fixed a bug where comparing an empty Text against an empty Text literal would return an incorrect result.
<a href="#">38576</a>	Framework » Macintosh	File types are no longer written with everything as “exported types” which causes some file types to not get recognized by OS X.
<a href="#">38973</a>	Framework » Windows	Non-WAV based sound files dragged into the project now Play properly again.
<a href="#">38929</a>	IDE » Layout Editor	Image sets now properly reload into the TabView editor.
<a href="#">38982</a>	IDE » Layout Editor	TabView icons which use Image Sets now render at the correct size.
<a href="#">38921</a>	IDE » Navigator	Deleting an event implementation from a control on iOS actually removes it so its no longer rendered out
<a href="#">38977</a>	IDE » Navigator	Image sets now load properly from projects in version control format.
9 Bug Fixes		
9 total items		

# Xojo 2015r2 Release Notes

Xojo 2015 Release 2 contains about 100 changes. Some of the more notable new features and changes include:

- iOS Icons and Images can now have multiple sizes associated with them.
- Expanded availability of parts of the New Xojo Framework to Desktop, Web and Console projects:
  - [Xojo.Data](#)
  - [Xojo.Net.HTTPSocket](#)
  - [Xojo.IO](#)
  - [Xojo.Crypto](#)
  - [Xojo.Math](#)
  - [Xojo.Introspection](#)
- Added [iOSDatePicker](#) control
- Reduced size of Windows and Linux built apps
- Language improvements
  - Added [PBKDF2](#) and [BER/DER](#) methods to Xojo.Crypto
  - XojoScript LLVM updated to 3.5.1

## Bug Fixes

<a href="#">38520</a>	Build	An error generating code for compiling a multiline label on a report has been fixed.
<a href="#">24019</a>	Build	Apple Scripts, Movies, Pictures, Sounds and RawData can no longer erroneously be selected to be encrypted as the data in the item was never touched or encrypted and being able to select them in the IDE lead people to incorrectly believe the data was being encrypted.
<a href="#">38286</a>	Compiler	Compiler: Fixed code generation for 2D arrays on 64-bit iOS.
<a href="#">38291</a>	Compiler	Compiler: Fixed compilation of Objective-C declares that returned large structures.
<a href="#">38140</a>	Compiler	Holding the option key down no longer generates an app that does not run. This used to switch between the Carbon and Cocoa frameworks but that is no longer supported (see case 19377).
<a href="#">38521</a>	Compiler	iOS applications now build when Xcode 6.2 or Xcode 6.3 is installed.
<a href="#">38020</a>	Compiler	Removed IsGlobal on MemberInfo since there is nothing in Introspection that will ever have it set to True.
<a href="#">38656</a>	Compiler	The library field of Objective-C declares is now respected and the application will link against those libraries.
<a href="#">38554</a>	Crashes & Assertions » Crash	A NilObjectException that could occur on OS X 10.10.3 has been fixed.
<a href="#">38715</a>	Crashes & Assertions » Crash	Fixed an out of bounds memory write that happened when converting a String to a PString.
<a href="#">37971</a>	Crashes & Assertions » Crash	License agreement window now has a minimum size of 320 x 320.

<a href="#">38385</a>	Debugger	Debugger: Fixed a bug that led to corrupting the last character when editing a string in the debugger.
<a href="#">29166</a>	Framework » All	Removed deprecated BinaryStream functions: ReadLong, ReadByte and ReadShort WriteLong, WriteByte and WriteShort
<a href="#">38475</a>	Framework » iOS	iOS SQLiteDatabase no longer crashes when getting a NULL value from a column, also added a IsNull property to SQLiteDatabaseField.
<a href="#">37939</a>	Framework » iOS	iOSSegmentedControl now properly handles adding and removing segments at run-time.
<a href="#">37934</a>	Framework » iOS	The Enabled property now works with iOSToolButton.
<a href="#">38407</a>	Framework » iOS	The UnhandledException event is now called when the app is run on iOS devices.
<a href="#">10325</a>	Framework » Macintosh	Serial.XmitWait no longer raises an Error code of 19.
<a href="#">38695</a>	Framework » Macintosh (Cocoa)	Accessing screen indexes out of bounds no longer causes a crash. It will throw an out of bounds exception.
<a href="#">38808</a>	Framework » Macintosh (Cocoa)	Macintosh: Xojo.Core.Timer now works on 10.7 and 10.8.
<a href="#">38217</a>	Framework » Macintosh (Cocoa)	User selected font and size for the ListBox is honored by the listbox header again.
<a href="#">21576</a>	Framework » Web	Web: Cookies whose names contain spaces now load properly when a session first starts up.
<a href="#">38511</a>	Framework » Web	Web: WebContainer ContentsScrolled event now fires when scrolling in only one direction.
<a href="#">38284</a>	Framework » Web	WebSDK: Fixed a bug which caused the javascript namespace creation method to be called once per control instance instead of once per unique namespace.
<a href="#">38459</a>	Framework » Windows	Adding or subtracting a DateInterval now correctly adds/subtracts months (also fixed for Linux).
<a href="#">34157</a>	Framework » Windows	MediaPlayer.Looping now works properly with the native Windows Media Player.
<a href="#">37324</a>	Framework » Windows	Setting Clipboard.Picture, with a Picture that has an alpha channel, no longer crashes.
<a href="#">38703</a>	IDE	Removing a tab from an iOSScreen in a text project works as expected.
<a href="#">37857</a>	IDE » Auto Complete	Removed hidden Text Iterator classes from auto-complete.
<a href="#">38015</a>	IDE » AutoLayout	IDE no longer hangs when auto-layout causes a TextArea size to get very small.

<a href="#">11465</a>	IDE » Code Editor	Items that have been deleted are no longer still showing when you search, delete them then click the search result again.
<a href="#">38208</a>	IDE » Code Editor	TargetARM now autocompletes.
<a href="#">38296</a>	IDE » Code Editor	The Text type highlights like other intrinsic types.
<a href="#">38492</a>	IDE » Debugger	Structure values display correctly.
<a href="#">38123</a>	IDE » Find & Replace	Searching using "this item and subclass" option with a module selected now also searches any classes contained within the module.
<a href="#">38443</a>	IDE » Inspector	Changing the name of a WebToolbarItem and leaving the item by clicking elsewhere no longer causes an endless stack of dialogs.
<a href="#">37301</a>	IDE » Inspector	Duplicating a string property no longer doubles up quotes in the duplicate.
<a href="#">38090</a>	IDE » Inspector	Font selector in the Inspector now shows in alphabetical order (ignoring odd characters like @ that occur in some font names).
<a href="#">38021</a>	IDE » Inspector	The Method Editor Combobox now splits up array types as parameters properly.
<a href="#">38142</a>	IDE » Inspector	The Value property no longer appears (then disappears) on the Inspector for a CheckBox control. Instead, use the Initial State property in the Inspector.
<a href="#">38357</a>	IDE » Inspector	You can't use a SplitView for the default iPhone screen (iOS does not support this).
<a href="#">38087</a>	IDE » Menu Editor	Menu Editor (and all other command bar icons) draw more clearly since they are no longer scaled.
<a href="#">38560</a>	IDE » Miscellaneous	A NilObjectException that could occur due to a bug in OS X 10.10.3 is now fixed.
<a href="#">38545</a>	IDE » Miscellaneous	An NilObjectException that occurred when the Debug Plugins directory exists next to the IDE has been fixed.
<a href="#">37880</a>	IDE » Miscellaneous	Compiling a project where there are files that were missing that were not resolved no longer causes an exception in the IDE.
<a href="#">14078</a>	IDE » Miscellaneous	Dropping an invalid "cur" file on the IDE no longer causes a NilObjectException.
<a href="#">37970</a>	IDE » Miscellaneous	Fixed a memory leak related to the Window Layout Editor.
<a href="#">38281</a>	IDE » Miscellaneous	Fixed a NilObjectException that can occur when adding an overloaded method.
<a href="#">36697</a>	IDE » Miscellaneous	Fixed a NilObjectException that could occur when using a Filter for the Navigator.
<a href="#">38409</a>	IDE » Miscellaneous	Fixed an edge case where closing the Xojo Cloud upload dialog/sheet could cause a crash.

<a href="#">19377</a>	IDE » Miscellaneous	Holding the Option key down when pressing Run no longer produces an error.
<a href="#">24321</a>	IDE » Miscellaneous	IDE: Pasting string properties no longer doubles up the surrounding and contained quotes.
<a href="#">38344</a>	IDE » Miscellaneous	Identifiers are created as you type in a name and company in the Project Chooser.
<a href="#">33331</a>	IDE » Miscellaneous	Invalid names for WebToolBarItems no longer cause compile issues.
<a href="#">38570</a>	IDE » Miscellaneous	No longer generate an OutOfBoundsException with an oddly formed list of parameters for an extends method.
<a href="#">37977</a>	IDE » Miscellaneous	Selected items in the Errors pane are now readable on Windows and Linux.
<a href="#">36765</a>	IDE » Miscellaneous	Switching from tab to tab no longer requires clicking in the code editor to make it so you can start/continue editing.
<a href="#">38129</a>	IDE » Miscellaneous	The deprecated CFBundleGetInfoString is no longer written to Info.plist for OS X apps.
<a href="#">38241</a>	IDE » Miscellaneous	The Resume button on the toolbar on Windows now enables when expected during debugging.
<a href="#">37964</a>	IDE » Miscellaneous	Undoing and redoing the addition of a project item (like a container control) properly restores the Navigator, avoiding a NilObjectException.
<a href="#">38145</a>	IDE » Miscellaneous	Web apps no longer report NaN as the bytes per transaction under some circumstances in the log.
<a href="#">38175</a>	IDE » Navigator	A cosmetic issue where multiple entries for "Layout" could arise has been fixed.
<a href="#">38667</a>	IDE » Navigator	Clicking from one item to another in the Navigator no longer dirties the project.
<a href="#">38368</a>	IDE » Style Editor	Linux IDE: deleting a property from a WebStyle no longer crashes
64 Bug Fixes		

## Changes

<a href="#">35574</a>	Framework » All	Xojo.Crypto now works on Desktop, Web and Console projects for Windows, Linux and OS X.
<a href="#">35218</a>	Framework » All	Xojo.IO.FolderItem now works for all platforms.
<a href="#">38311</a>	Framework » All	Xojo.Net.HTTPSocket now works on all platforms (note this requires libsoup-2.4 on Linux).
<a href="#">37645</a>	Framework » Linux	The Linux framework now uses the system provided ICU libraries instead of statically linking them, which reduces the size of built apps on Linux. This means libicu 4.2+ is required to be installed on your Linux distros.

<a href="#">35284</a>	Framework » Web	Standalone web app startup parameters can now be manipulated in App.Open by changing the values in the args array.
<a href="#">37646</a>	Framework » Windows	Reduced size of Windows ICU DLLs by statically linking them in and removing unused features from the library.
<a href="#">36303</a>	IDE » AutoLayout	Added restrictions on what constraints you can add if a control can't be resized in various directions.
<a href="#">38382</a>	IDE » Debugger	The modal dialog which asks whether you want to continue waiting for a web app to launch now has a Hide button so the dialog will not continue to interrupt the debugging process.
<a href="#">38519</a>	IDE » Inspector	Constant editor inspector pane behaves like the property editor panes and selects the name when you create a new constant.
<a href="#">38602</a>	IDE » Inspector	When you add an even definition its name is now selected in the Inspector.
<a href="#">38039</a>	IDE » Library	Long descriptions in the library can now be scrolled.
<a href="#">38318</a>	IDE » Miscellaneous	New iOS projects have an icon asset and launch image asset instead of the App Icon and Launch Images folders.
<a href="#">38401</a>	IDE » Miscellaneous	The template window split is more proportional so it looks nicer.
<a href="#">25359</a>	IDE » Miscellaneous	Unknown type errors now display the name of the unknown type.
<a href="#">38495</a>	IDE » Navigator	Focus is set on the name property in the Inspector for items that only have a simple set of ID group properties. Basically those that only have a name property like folders and menus.
<a href="#">25640</a>	IDE » Window Editor	Can paste methods, consts, enums and structures when they are copied from full text.
<a href="#">38134</a>	Plugin SDK	Plugins SDK: Removed the obsolete plugin converter tool.
<a href="#">38132</a>	XojoScript	XojoScript: The version of LLVM used by XojoScript has been upgraded to 3.5.1.
18 Changes		

## New Items

<a href="#">38298</a>	Compiler	Compiler: Added an optimization pass for Objective-C method calls that removes excess calls to sel_registerName.
<a href="#">37660</a>	Framework » All	On Windows/Linux, external resource items (i.e. Pictures, Sounds, Movies) are no longer compiled into the built executable, they are now copied to a Resources folder next to the built app. The benefit is that the resource is not loaded into memory until accessed, and Pictures are no longer converted into BMPs.
<a href="#">38009</a>	Framework » All	Xojo.Data namespace is now available for all targets/platforms.
<a href="#">38341</a>	Framework » iOS	Added iOSDatePicker control.

<a href="#">38371</a>	Framework » iOS	Added missing Crypto functions for Xojo.Crypto: PBKDF2 BERDecodePrivateKey BERDecodePublicKey DEREncodePrivateKey DEREncodePublicKey Hash
<a href="#">38174</a>	Framework » Web	Added ability to set the HttpOnly attribute in the Session.cookie.set method. Removed the sessionid cookie from the javascript framework.
<a href="#">38422</a>	Framework » Web	The WebRequest object passed to HandleSpecialURL and HandleURL now has a Secure property which tells you if the request came in over a secure channel.
<a href="#">38188</a>	Framework » Web	Web: The SSL Certificate location can now be specified on the command line using --certificate=/full/path/to/file
8 New Items		

## Docs and Examples

<a href="#">38505</a>	Documentation » Language Reference	Changed links to TargetHasGUI to instead link to TargetDesktop. Removed links to TargetCarbon.
<a href="#">38060</a>	Documentation » Language Reference	Clarified Notes for Encoding method to make it clear it does not “guess” the encoding of the String and only returns what the encoding has been set as.
<a href="#">38297</a>	Documentation » Language Reference	Corrected typo in sample code for Xojo.Core.DictionaryEntry.
<a href="#">38025</a>	Documentation » Language Reference	Links to MemberInfo.Parameters now correctly say “Parameters” instead of “GetParameters”.
<a href="#">38046</a>	Documentation » Language Reference	Notes for Declare now indicate the Library must be either a String literal or a Constant.
<a href="#">38429</a>	Documentation » Language Reference	Redirected Session.UnhandledException to WebSession.UnhandledException.
<a href="#">38496</a>	Documentation » Language Reference	ScrollBar.LiveScroll no longer indicates it is read-only.
<a href="#">38182</a>	Documentation » Language Reference	ServiceApplication now shows the command to manually start a Windows service.
<a href="#">38428</a>	Documentation » Language Reference	SQLiteDatabase now shows that SQLite 3.8.8 is used.
<a href="#">38400</a>	Documentation » Language Reference	Updated Graphics description to indicate you can draw to Graphics from a Picture.
<a href="#">38398</a>	Documentation » User's Guide	Updated all download links to User Guide to point to the correct version.
<a href="#">38609</a>	Examples	iOS UIDatePicker Declare example now correctly gets date selections for 1947 and earlier.
<a href="#">38448</a>	Examples	YUI Text Editor example (using WebSDK) now uses HTMLHeader shared method on control rather than App.HTMLHeader to make it easier to reuse the control in other projects.

13 Docs and Examples
103 total items

# Xojo 2015r1 Release Notes

Xojo 2015 Release 1 contains about 150 changes. Some of the more notable new features and changes include:

- 64-bit iOS Builds
  - iOS builds are now created as Universal Binaries containing both 32-bit and 64-bit parts as required to meet Apple's new App Store submission requirements
- A new TargetARM constant for conditional compilation
- IDE improvements, including
  - Preference for Searching
  - 1024x1024 icon sizes
  - Better display of debugger values for pictures and arrays
- Language improvements
  - Added Parse method to Integer, Double and Single data types
  - SQLite updated to 3.8.8

## Bug Fixes

<a href="#">19863</a>	All: Fixed a crash when passing a structure array into SortWith.
<a href="#">37833</a>	All: Fixed a memory leak with Xojo.Core.MemoryBlock.
<a href="#">37835</a>	All: Fixed a regression where assigning empty strings to a MemoryBlock would crash.
<a href="#">37674</a>	All: Fixed Array.IndexOf's handling of Auto arrays.
<a href="#">37664</a>	All: Fixed Array.IndexOf's handling of delegate objects.
<a href="#">37093</a>	All: Fixed RuntimeException's CallStack property having an extra Nil element at the end of the array.
<a href="#">32325</a>	All: Fixed shuffling of structure arrays.
<a href="#">33905</a>	All: Optimized sorting an array that is already mostly sorted.
<a href="#">37363</a>	All: String.ToText no longer raises an exception on Windows/Linux if the string is empty.
<a href="#">37811</a>	All: Text.Split no longer raises an assertion if the Text to split is empty. It now returns an empty Array instead.
<a href="#">17059</a>	All: Using IndexOf on a string array where a string starts with NUL byte now works correctly.
<a href="#">37611</a>	All: Xojo.Core.Timer.CancelCall no longer raises an IteratorException.
<a href="#">37871</a>	Cocoa: Fixed a crash with ComboBox.DeleteAllRows.
<a href="#">37201</a>	Cocoa: The ArrowAllDirections mouse cursor now resembles the old Carbon appearance instead of a simple crosshair.
<a href="#">37351</a>	Compiler: Fixed a bug that made ByRef arguments count for overload resolution, which incorrectly marked function calls as ambiguous.
<a href="#">37629</a>	Compiler: Fixed a bug with incremental compilation that resulted in global functions or variables not being counted as a dependency that should force a recompile if they change.

<a href="#">37981</a>	Compiler: Fixed a bug with unsigned integer comparison on iOS and XojoScript.
<a href="#">37616</a>	Compiler: Fixed a crash when trying to get or set a structure property's value via introspection.
<a href="#">37663</a>	Compiler: Fixed a crash with exception handlers when building for iOS devices.
<a href="#">37508</a>	Compiler: Fixed a failed assertion triggered by calling an invalid function.
<a href="#">37567</a>	Compiler: Fixed a failed assertion when compiling a class that has a property that is a large structure.
<a href="#">37500</a>	Debugger: Date values stored in a Variant now show correctly in the debugger.
<a href="#">34555</a>	IDE: A change to a project item name is validated before the contextual menu is shown to you cannot create a class with no name.
<a href="#">28114</a>	IDE: Added missing Help tags for the file types command bar items.
<a href="#">37576</a>	IDE: Added Variant.TypeText to autocomplete.
<a href="#">37041</a>	IDE: Adding a new sqlite database to a project no longer permits using the name "Database".
<a href="#">13259</a>	IDE: Alt-Shift-Click inserts the name of the item selected in the navigator to the current insertion point in the active code editor.
<a href="#">37581</a>	IDE: Altered reloading an iOSControl so it reloads all saved properties.
<a href="#">37406</a>	IDE: An event that has been implemented can be removed and re-added to a control.
<a href="#">36823</a>	IDE: Auto-layout baselines guides now result in baseline constraints.
<a href="#">21281</a>	IDE: Changed from using JPEG 2000 to PNG for the elements of the ICNS file so that OS X builds done on Windows and Linux properly create the ICNS file.
<a href="#">37564</a>	IDE: Clicking the help item in the toolbar no longer results in a help window with no toolbar configured.
<a href="#">25894</a>	IDE: Closing the last project window will, if set, close the IDE down or simply leave the tray item (as set in preferences).
<a href="#">37570</a>	IDE: COM help reference entries wont be shown on platforms that don't support COM controls.
<a href="#">29017</a>	IDE: Commands like Stop Debugging work even when the Debugger is not the current active tab.
<a href="#">17852</a>	IDE: Copy file steps now directly supports the use of partial paths for the copy file steps subdirectory property.  Note : Partial paths still behave platform specific in some respects. On OS X if you use the colon (:) it will be treated as a /. On Windows if you use / it will be treated as a \.
<a href="#">31436</a>	IDE: Copy paste and drag drop of methods, constants, etc. into external classes now works properly.
<a href="#">37454</a>	IDE: Copy/copy/paste now work with the IDE Script Editor window.
<a href="#">37601</a>	IDE: Copying a group of lines no longer causes an exception when trying to create the backing image.
<a href="#">29272</a>	IDE: Creating a new control set from a selected group of similar controls now works properly.

<a href="#">37613</a>	IDE: Custom controls names in the Library update when changed.
<a href="#">19639</a>	IDE: Custom supers for reports saved in plain text format are restored properly.
<a href="#">14434</a>	IDE: Desktop containers draw better in the IDE (back color and backdrop) in their editor and in a disabled form on a window layout.
<a href="#">37194</a>	IDE: Don't raise a NilObjectException if you start a search in the LangRef window before the response has been received from one of the online sites.
<a href="#">37507</a>	IDE: Dropping a method on a class interface behaves like a Copy not a Move.
<a href="#">21258</a>	IDE: Ellipsis to indicate further interaction required have been added to menus.
<a href="#">37188</a>	IDE: Event definition editor pane parses a single entry in the name into component parts as the method editor pane does.
<a href="#">37037</a>	IDE: Event handlers that are added to a control instance but have no code in them are now correctly hooked up so add handler will properly report an error if you try to add a handler to it again.
<a href="#">9808</a>	IDE: External project items no longer show property editors differently than internal project items.
<a href="#">33217</a>	IDE: External script references no longer get lost.
<a href="#">19099</a>	IDE: Extract Interface and Extract Superclass both order their lists alphabetically.
<a href="#">37232</a>	IDE: Fixed a NilObjectException that could occur when trying to add attributes to a NON-UI class in an iOS project.
<a href="#">37684</a>	IDE: Fixed issue where right click would incorrectly position the insertion point in the Code Editor.
<a href="#">37577</a>	IDE: Fixed the displayed type of Variant.TextValue in autocomplete.
<a href="#">32736</a>	IDE: Floating properties palette has the advanced icon and panels.
<a href="#">37687</a>	IDE: Grab handles for lines appear regardless of setting of X1, X2, Y1, and Y2.
<a href="#">23591</a>	IDE: ICNS files are now written out with the Apple recommended set of images as PNG which do NOT include any 48 x 48 images.
<a href="#">38035</a>	IDE: Icon editor no longer beeps when you press the delete key.
<a href="#">36968</a>	IDE: Icon editor no longer behaves like its compositing icons together. It now replaces the existing icon with a new one when dropped into the editor.
<a href="#">37727</a>	IDE: If a plugin is misconfigured and claims to have properties, events, methods, etc. but really doesn't then the error message in Feedback will identify this better.
<a href="#">37699</a>	IDE: If there is no default value for a constant or property no equal sign is shown in the Navigator.
<a href="#">36800</a>	IDE: In Auto-layout, controls constrained to the baselines of other controls move as expected.
<a href="#">17739</a>	IDE: In the Code Editor, AddHandler and RemoveHandler now display syntax help and are highlighted as keywords.

<a href="#">37072</a>	IDE: In the Code Editor, clicking in the gutter below the last visible line no longer toggles the last line's break point.
<a href="#">36980</a>	IDE: In the File Type Set Editor, pressing forward delete when a field is being edited simply edits the field. When no field is being edited and a row is selected pressing forward delete removes the row.
<a href="#">15132</a>	IDE: In the Style Editor, typing in a value in the text field for colors now works.
<a href="#">25704</a>	IDE: Inspector Behavior sheet is resizable. Lists resize to fit width and columns are resizable.
<a href="#">21393</a>	IDE: Make it so when a web segment has an icon assigned that it actually draws it whether its been copied and pasted or saved and the project reloaded.
<a href="#">22116</a>	IDE: Menu handlers now get numeric suffixes like enums, properties, etc. do when you duplicate them.
<a href="#">37099</a>	IDE: Method editor now says "Delegate name" instead of "method name" when you edit a delegate.
<a href="#">37403</a>	IDE: No longer causes an exception when using a script command to build a console or web app.
<a href="#">37401</a>	IDE: No longer causes an exception when using a script command to build a console or web app.
<a href="#">19115</a>	IDE: No longer get an error when a toolbar button icon image has the same name as the button.
<a href="#">19340</a>	IDE: No longer repeatedly ask if a person wants to update a projects minimum load version as they alter it and save.
<a href="#">37060</a>	IDE: Opening new tabs uses the current Find panel size (if open).
<a href="#">37700</a>	IDE: Plugin super classes are not shown in the super class browser for iOS projects.
<a href="#">14913</a>	IDE: Pressing the Enter key completes inline editing same as the return key does.
<a href="#">28654</a>	IDE: Project chooser panel no longer gets spurious line artifacts when resizing.
<a href="#">37656</a>	IDE: Removed extra space in build dialog.
<a href="#">25046</a>	IDE: Renaming Report controls in the Inspector also changes the name in the Navigator.
<a href="#">37285</a>	IDE: Resizing a control by the corner drag handles snaps to guides just like dragging any of the side handles does.
<a href="#">17862</a>	IDE: Segmented control with no segments no longer generates an error when you analyze a project.
<a href="#">37381</a>	IDE: The Export dialog when making a folder external has a more sensible caption. Folders properly check all contents recursively for whether they can be exported.
<a href="#">35005</a>	IDE: The Insert menu rebuilds properly depending on what project you have front most.
<a href="#">25580</a>	IDE: The Library no longer appears to be a drop target when you are dragging a control out of the library.
<a href="#">33190</a>	IDE: The ListBox popover editor permits shift tab to go backwards (just as it uses tab to go forwards).
<a href="#">27967</a>	IDE: The small progress wheel control in web projects no longer slowly creeps across the layout as you save and reopen the project.
<a href="#">26354</a>	IDE: The web style border editor can tab from the Size field to the container with the color selector fields.

<a href="#">35367</a>	IDE: The “Define missing method” function now works even when you select a text fragment that starts with “me.”.
<a href="#">28092</a>	IDE: Using Cut, Copy, Paste, Delete in the Font name combobox in the Inspector doesn’t cause the control to be deleted.
<a href="#">37615</a>	IDE: When a base class name for an iOS custom control changes, any views with instances of that control also get the super updated for those instances.
<a href="#">30622</a>	IDE: When using floating palettes and the last workspace for a project is closed the floating properties palette also clears so that you cannot modify something that no longer exists.
<a href="#">37692</a>	IDE: When you change the default value of a constant or property value they change in the Navigator right away so you no longer need to close then open the group of constants or properties to see the change.
<a href="#">23263</a>	IDE: You can now do a discontinuous select of project items in the Navigator on Windows and Linux (this already worked on OS X).
<a href="#">37326</a>	iOS: An empty iOS project no longer generates a warning about an unused variable.
<a href="#">37008</a>	iOS: Controls that are visually parented in the IDE are now properly parented at run time.
<a href="#">35555</a>	iOS: Dynamic constants are now working.
<a href="#">37919</a>	iOS: Fixed a crash with invalid JSON.
<a href="#">37449</a>	iOS: Fixed a failed assertion triggered by dereferencing a pointer with no offset.
<a href="#">37669</a>	iOS: Removed File Types from the Build Settings.
<a href="#">37071</a>	iOS: The Sign project step name (in Build Settings) is no longer editable.
<a href="#">37678</a>	Linux: IDE no longer crashes at launch on Ubuntu 14.10.
<a href="#">34626</a>	Linux: The IDE no longer crashes in certain instances when clicking on the button to change a class' super.
<a href="#">20370</a>	OS X: Because of a bug in Launch Services names with special characters don’t work for the icon file name. Now always using App.icns as the name.
<a href="#">37777</a>	OS X: Fixed a crash with Xojo.Core.Date when used as a property.
<a href="#">37672</a>	OS X: Fixed Xojo.Math.RandomInt crashing.
<a href="#">15019</a>	Plugins: REALLoadObjectMethod no longer fails with DrawPicture in Console apps.
<a href="#">33896</a>	Remote Debugger: Changing the “launch after receiving” check box no longer takes several seconds for the UI to update.
<a href="#">17882</a>	Serial: Retrieving Bits, Stop, Parity, and Port properties now work properly instead of incorrectly retrieving its high word (i.e upper 16-bits did not contain the correct data). This was seen mainly on Windows.
<a href="#">37123</a>	Web: A WebControlWrapper Style property no longer appears if the developer has indicated that it shouldn't.

<a href="#">37085</a>	Web: Fixed a bug which caused WebLabels to be invisible when included in a dynamically created WebContainer.
<a href="#">37748</a>	Web: Fixed a regression in the WebSDK which caused ControlAvailableInBrowser to return True too early. Increased WebControlWrapper.APIVersion to 5 so the only version that has this issue will be version 4.
<a href="#">37670</a>	Web: Removed unused ChromeFrame code from the framework.
<a href="#">37164</a>	Web: WebCheckboxes again respond to touches when their MouseDown event is implemented.
<a href="#">37435</a>	Web: WebContainer mouse event handlers no longer interfere with scrolling.
<a href="#">37205</a>	Web: WebLabels in a WebContainer in a Control Array draw properly again.
<a href="#">37760</a>	Web: WebListbox no longer offsets the selection if placed inside a WebContainer and accessed from a touch device.
<a href="#">37691</a>	Web: WebMoviePlayer and WebMapView no longer incorrectly send Mouse movement events to the server.
<a href="#">17126</a>	Web: WebStyle gradients now render for IE. Note: IE only supports 2 point gradients.
<a href="#">34341</a>	Web: Webstyles with borders no longer cause controls to change size at runtime.
<a href="#">37745</a>	WebSDK: Updated documentation to reflect the APIVersion 4 and 5.
<a href="#">37562</a>	Windows: Deleting a menu in the Menu Editor on Windows no longer causes an exception.
<a href="#">21685</a>	Windows: Graphics.DrawPolygon now draws smoother edges when using GDI+
<a href="#">37413</a>	Windows: Xojo.Core.Date subtraction now works properly.
<a href="#">37390</a>	Windows: Xojo.Core.Timer now works for Desktop apps.
<a href="#">37756</a>	XojoScript: Fixed a bug that prevented CompilerWarning from firing when there were warnings on consecutive lines.
<a href="#">31641</a>	XojoScript: The compiler now gives an error when a script declares its own RuntimeException class.

Total: 127

## New Items

<a href="#">37510</a>	Compiler: Added a TargetARM constant that is set to true when building for ARM devices.
<a href="#">37930</a>	Compiler: The 'StructureAlignment' attribute on a structure can now be set to 0, which indicates that the compiler should perform natural alignment. Natural alignment ensures that the structure will be laid out correctly for a given platform's ABI rules.
<a href="#">28073</a>	IDE: Added a new preference for controlling whether searches and filtering happen immediately or wait until enter/return are pressed.
<a href="#">21947</a>	IDE: Added support for 1024 x 1024 icon size.

Total: 4	
<b>Changes</b>	
<a href="#">37439</a>	All: Added Parse function to Integer, Double, and Single. This function acts more like Val/CDbl did, in that it's more lenient at parsing and will not raise exceptions.
<a href="#">37757</a>	Compiler: TargetHasGUI, TargetPPC, TargetPowerPC, TargetMacOSClassic, and Target68K have formally be marked as deprecated.
<a href="#">37156</a>	Compiler: The compiler now preserves type information better when dealing with "Integer" and "UInteger". As such, error messages will reflect what the user types instead of "Int32" or "UInt32".
<a href="#">37709</a>	IDE: Added Icons for autocomplete purposes for Auto, Text and Ptr.
<a href="#">35791</a>	IDE: Auto-layout baselines guides now result in baseline constraints.
<a href="#">35950</a>	IDE: Auto-layout baselines guides now result in baseline constraints.
<a href="#">36318</a>	IDE: Certain deprecated items have been removed from file formats and print outs of projects.
<a href="#">36644</a>	IDE: Debugger Viewer now is labelled with "View as " instead of "view".
<a href="#">17995</a>	IDE: It is possible to paste multiline text into the constant editor listbox on Windows and Linux (it already worked on OS X).
<a href="#">16569</a>	IDE: Revised project printing to not repeat the note name, group things better and add in delegates and using clauses.
<a href="#">12874</a>	IDE: The Debugger shows headers for 1D arrays as Row and Value. For 2D arrays it shows row, column and value. For a 2D array sorting by rows puts data in Row / Column order.
<a href="#">17234</a>	IDE: The picture in the debugger view now scrolls as you move the mouse wheel.
<a href="#">37614</a>	IDE: There is now a Project Controls section in the Library for iOS projects.
<a href="#">12574</a>	IDE: Trying to place an instance of a Container Control or EmbeddedWindowControl directly on a window is not permitted. You can no longer drag out a control, like a canvas, change its super to Container Control or EmbeddedWindowControl.
<a href="#">37929</a>	IDE: Updated structures to support natural alignment. This means now you can set the StructureAlignment attribute to 0 to use natural alignment.
<a href="#">36648</a>	IDE: Viewing a window in the debugger shows "Controls" instead of "Contents" in the debugger.
<a href="#">8287</a>	IDE: When you use Run Paused, the Resume toolbar item, resume menu item and pause items will be disabled until such time the app to be debugged has been started and communicated with the IDE.
<a href="#">29995</a>	OS X: ICNS files written use up to date internal image formats (PNGs instead of JPEG 2000) and only use currently documented formats.
<a href="#">37329</a>	PluginSDK: Updated Plugins SDK with new function to handle the Text datatype. Please review the Plugins SDK for details.

<a href="#">36218</a>	SQLiteDatabase: Updated to SQLite 3.8.8.
<a href="#">37681</a>	Web: Removed the "Xojo HTTP Server" header from the list of headers returned for a standalone web app.
<a href="#">37741</a>	Web: Updated standalone HTTP response headers to more closely match existing behavior.
<a href="#">26807</a>	Web: Web Apps no longer request that ChromeFrame be activated if it's installed on the user's computer.
Total: 23	

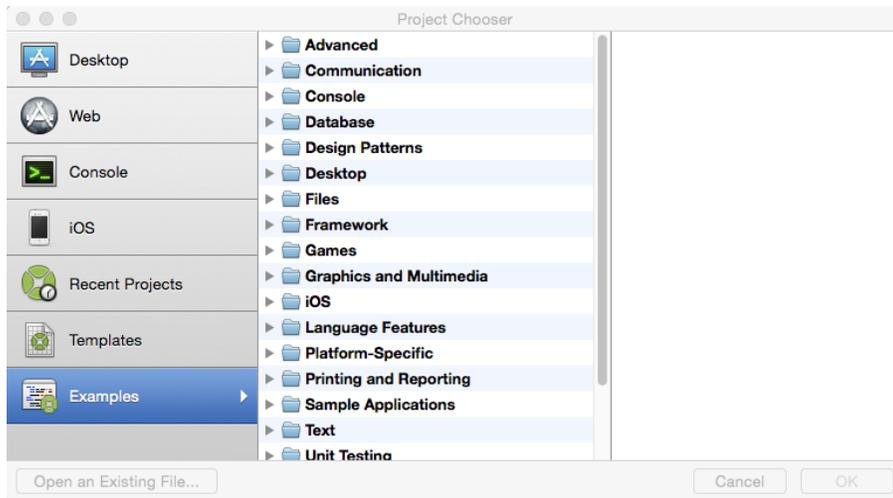
## Docs and Examples

<a href="#">37947</a>	DocLangRef: Added Crypto and Introspection to list of Themes in Language Reference.
<a href="#">37274</a>	DocLangRef: Added information for WebApplication.Security.
<a href="#">30413</a>	DocLangRef: Added note to array.Sort to indicate how items are sorted.
<a href="#">37374</a>	DocLangRef: Added page for REALbasic module as an alternate way to access some global items.
<a href="#">37813</a>	DocLangRef: Corrected "REALSQLDatabase" to "SQLiteDatabase" in Record Navigation section of RecordSet page.
<a href="#">36666</a>	DocLangRef: Removed System.KeyScript, which is no longer an available method.
<a href="#">36308</a>	DocLangRef: Simplified wording for description of FolderItem.GetSaveInfo.
<a href="#">37517</a>	DocRefGuide: Added exceptions raised by Xojo.IO.SpecialFolder.GetResource.
<a href="#">37738</a>	DocRefGuide: Fixed sample code for UIImage.FromFile.
<a href="#">37884</a>	DocRefGuide: Removed incorrect information from UIView.Activate and Deactivate.
<a href="#">37148</a>	DocRefGuide: Removed SelChange from the events for UITextField.
<a href="#">37151</a>	DocRefGuide: Removed usage of Math.Pi from sample code.
<a href="#">37779</a>	Examples: Added iOS/Declares/Base64Encoding project.
<a href="#">37780</a>	Examples: Added iOS/Declares/Pasteboard to show how to add text to the pasteboard.
<a href="#">37932</a>	Examples: Added XojoDoodle example for iOS.
<a href="#">37647</a>	Examples: Fixed iOS/Database/SQLiteInMemory example so that it can be properly code-signed when built.
<a href="#">37778</a>	Examples: In iOSAlerts, changed declare lib to /usr/lib/libobjc.dylib so that it builds for device.
<a href="#">37737</a>	Examples: Updated IDECommunicator example to work with current version of Xojo.
<a href="#">37931</a>	Examples: Updated XojoNotes to use SplitView on iPad.
<a href="#">37159</a>	Examples: Updates to error messages for XojoScript example project.

Total: 20
Release Total: 174

# Examples

There are over 320 example projects included in the [Xojo Download](#), which you can access directly from the Project Chooser.



Here is a searchable list of what is you'll find in the Examples folder when you download Xojo. You can also [download the examples separately](#).

## Advanced

### AddHandler

- AddHandler

### Build Automation

- AutoSaveScript

### Declares

- DeclareGetPID
- SoftDeclares
- WindowOpacity

### Delegates

- DelegateExample

### IDE Scripting

### IDECommunication

- IDECommunicator

### Shell

- AsyncShellExample
- GetDirectory
- Interactive Shell

- ShellBackendTest

## Thread

- Semaphore Example

## XojoScript

- Evaluator

## Imaging

- ScriptImaging

## Scripts

- XojoScript

## Communication

### Internet

- AutoDiscovery
- Email Example
- EmailSSLExample
- GoogleMaps
- HTTP Example
- HTTPDownloader
- HTTPSocketPostExample
- JSON Example
- NetworkInterface
- ServerSocketClientTest
- ServerSocketServerTest
- SimpleChat
- SOAPConversionRate
- SOAPTempConv
- SynchronousTCPSocketClass
- SynchronousTCPSocketExample
- TwilioAnswerCall
- TwilioSMS
- UDPEXample

### Web Server

- WebServer
- WebServerWindowsService
- YahooWeather
- YouTube
- IPCSocket

## Serial

- Line State Change Tester
- SerialLineIndicator

---

## Console

- ArraysConsoleAndUserInput

## Multiprocessing

- WordCounter
- WordCounterGUI
- signals

## Database

- DatabasePictureTest

## DataControl

- DataControlExample

## MSSQLServer

- MSSQLServerExample

## MySQL

- MySQLExample

## ODBC

- GetDataSourceNames
- ODBCExample

## Oracle

- OracleExample
- OraclePluginTest

## PostgreSQL

## LargeObjects

- LargeObjects

## ListenAndNotify

- Listen
- Notify
- PostgreSQLExample

## PreparedStatement

- SQLitePreparedStatement

## SQLite

- SQLiteBackup
- SQLiteBlob
- SQLiteExample

---

## Design Patterns

- Decorator
- Factory
- Interpreter
- LazyInitialization
- Observer
- Singleton

## Desktop

### ContainerControls

- ContainerControlExample
- DownloadContainer

### TabbedWebBrowser

- TabbedWebBrowser

### Controls

- ControlSets
- GroupBox

### ListBox

- FileBrowser
- IconGrid

### Icons

- ListBoxDragBetweenCells
- ListBoxExample
- ListBoxExportExample
- ListBoxGridExample
- PropertyListExample
- SourceListExample
- ProgressBar
- TextFieldResizing

### Toolbar

- ToolbarExample

### Custom Controls

- CalendarWindow
- CanvasButton
- LinkLabel
- OKCancelContainer

---

## Dialog Boxes

- DialogsExample

## DragAndDrop

- ListBoxDragAndDrop
- ListBoxesDragAndDrop

## Menus

- ConstructContextualMenu
- FontMenu
- OpenRecentMenu
- WindowMenu

## Threading

- ThreadingExample

## TrayItem

- TrayExample

## UpdatingUIFromThread

- UIThreadingWithTask
- UIThreadingWithTimer

## Files

### BinaryStream

- BinaryStream
- Copy and Move Example
- DragToFinder
- FileBrowser
- FileSearch
- FolderInfo
- FolderItem Dialogs
- MacOSXFSRefs
- SpecialFolderPaths

## Framework

- CryptoExample
- CryptoRSAExample

## Games

### 2K

- 2K-Desktop

- 2k-Web

## Combat

- Combat

## Sounds

- GameInputExample

# Graphics and Multimedia

## Animation

- Animation

## Block Attack

- BlockAttack
- CanvasAreas
- CanvasDragRect
- CanvasDrawDrag
- CanvasDrawString
- CanvasScrolling
- CanvasZoom
- Dice
- DoReMi

## DragPicture

- DragPicture
- DrawingWithThreads
- FallingBlocks
- Fire
- FlowChart
- FolderMoviePlayer
- GradientExample
- GraphicalClock
- GraphTest
- GridLockExample

## Jewel Game

- JewelGame
- Obj2DTextRotation

## ObjectsInCanvas

### Graphics

- ObjectsInCanvas

---

## OpenGL

- FluidsDemo
- Pinball

## Sound

- Sound Player
- Sparkler
- System Colors
- VectorGraphics
- WaveLetter

## iOS

### Apps

- TicTacToe
- TipCalculator

### EddiesElectronics

- EEiOS

### XojoClicker

- XojoClicker
- XojoDoodle

### XojoNotes

- XojoNotes

## Auto-Layout

- LayoutConstraintExample
- NoCodeProportionalSpaced

## Controls

- DatePickerExample
- DynamicControls
- FontExample
- HTMLViewerExample
- MultipleViews
- ProgressExample
- SegmentedControlExample
- SplitViewExample
- SwipeExample
- SwitchExample

### TabBarExample

- TabBarExample

---

## Table

- GroupTableExample
- SectionTableExample
- SimpleTableExample
- TableCheckmark
- TableDataSource
- TableDetail
- TableDisclosure
- ToolbarExample
- TouchCanvas

## Database

- SQLiteInMemory
- SQLiteVersion

## Declares

- AlertSheet
- Base64Encoding

## CoreLocation

- CoreLocationLib
- IconBadgeNumber
- iOSAlerts
- Pasteboard
- SetFocus
- ShowURL
- Speak
- TableSelectAndScroll
- UIButtonDeclares
- UIDatePicker
- UIDevice
- UIScrollView

## Framework

- AdvancedThreadingExample
- CryptoExample
- DateExample
- IntrospectionExample
- ThreadExample
- TicksExample

## Graphics

- AnalogClock
- ClipExample
- DiceRoller
- PathExample
- Psychedelic
- RotationExample

- SineWaves
- SparkleTouch

## Networking

### EasyTCPSocket

- EasyTCPSocket Server
- EasyTCPSocket-iOS

### HTTPSocket

- iOSHTTPSocketsExample

### TCP

- tcpsocket\_ios
- tcpsocket\_server

## Sound

### GameBuzzer

- GameBuzzer

### NatureSounds

- NatureSounds

## Language Features

- ActionSourceInterfaceExample
- BuiltInAttributes
- Dictionary
- Enumerations
- ExtensionMethods
- MethodAssignment
- OperatorLookup
- OperatorOverload
- ParameterOptions

## Platform-Specific

### Linux

- GetDefaultControlSize
- OverlayScrollbars

### WindowShape

- WindowShape

---

## OS X

- AddressBook

## AppleScript

- AppleScriptTest
- Cocoa Declares
- DragPromises
- FastCocoaTextAreaUpdates
- FileSystem
- KeyChainExample

## PictureEffects

- CopyOSHandle

## Plist

- EnableRetina

## RetinaDisplay

- RetinaDisplay

## Windows

- AeroTest

## COM

- OLE Automation Tests
- CommandLinkExample
- CPUUsage

## CustomWindowShape

- WindowShape
- DeclareDrawing
- GDIPlusTest

## LoadMSAccessUsingADO

- LoadMSAccessUsingADO
- MutexExample

## Office Automation

- Excel Automation
- PowerPoint Automation
- Word Automation
- SetWindowIcon

## Printing and Reporting

---

## Printing

- GraphicsPrintingExample
- Labels
- Printing To Screen or Printer

## PrintingGraphics

- PrintingPicture
- StyledTextPrinterExample

## Reporting

- ListBoxReport

## GasReport

- GasReport

## Orders1

- List Of Orders

## Orders2

- Breaking List Of Orders

## Products

- List Of Products Preview
- List Of Products

## Sample Applications

- BookmarkManager
- Chat
- SlidersWeb
- TaskManager
- ToDo
- URLManager
- XojoText

## 8Queens

- 8Queens

## EddiesElectronics

### Database

- DemoDataGenerator

### Desktop

- EEDesktop

## Graphics

### Web

- EEWeb
- MapLocation
- SimpleBrowser

## Sliders

- Sliders

## Slideshow

- SlideShow

## XojoNotes

- XojoNotesDesktop
- XojoNotesWeb

## Text

- ClipBoard
- DragText
- SearchAndScroll
- StyledText
- XMLExample
- XMLReader

## Unit Testing

### XojoUnit

#### Graphics

- XojoUnitConsole
- XojoUnitDesktop
- XojoUnitWeb

## Web

- HashTagExample
- HelpDeskWithChat
- LoginExample
- PushExample
- QuoteService-SpecialURL
- ScrollingWebPage
- SecureLoginExample
- SessionTimeout
- WebSessionContext

---

## Containers

- ContainerControlDynamicExample
- ContainerControlExample
- Navigator
- ScrollingContainer
- TabPanelExample
- WebGridContainer

## Controls

- AudioPlayer
- ControlWithName Example
- LargeData
- ListBoxExample
- ProgressBar
- Scrollbar
- SortableListBox
- ThreadProgress
- TimerExample
- WebComboBox
- WebFileUploader

## CustomControls

- Gravatar
- HTMLArea
- jQuery Calendar

## ThreeStateButton

- ImageButton
- YUI Rich Text Editor

## iOS7Controls

- iOS7SimulatedControls

## MapView

- MapViewer

## Toolbar

- ToolbarExample

## Dialog Boxes

- DialogsExample
- DynamicDialogsExample

## Downloading

- Downloading

## Graphics

### Animator

- Animator
- CanvasBoxes

### CanvasChart

- EECanvasChart
- CanvasClock
- WebGridExample

## Printing

- HTMLViewerPrinting

## SSL

- SSLTest

## Styles

- StylesExample

## XojoCloudFileManager

- XojoCloudFileManager

## Xojo Framework

- IntrospectionExample
- UsingExample

# Webinar: Xojo Wars 2015 Community Battle

It's time for Xojo Wars. All submitted entries will be run in a live battle to determine the winner. Watch as it all unfolded live!

- [Xojo Wars on GitHub](#)

## XDC 2015: Introduction to Xojo iOS

In this session from the Xojo Developer Conference you'll learn about Xojo newest build platform: iOS. Xojo Developer Evangelist Paul Lefebvre covers general iOS concepts, the iOS project structure and iOS project items.

You can [order videos for all the XDC 2015 sessions from the Xojo Store](#).

# Creating an LED Circuit

In this tutorial, you will create a simple circuit with an LED and will then make the LED blink using a simple Xojo app.

For step-by-step instructions, refer to:

- [Blinking LED Tutorial](#)

# Button LED Circuit

In this tutorial you will extend the [Blinking LED Tutorial](#) to add a button. With Xojo code you will turn on the LED when the button is pressed and turn off the LED when the button is not pressed.

For step-by-step instructions refer to:

- [Button LED Tutorial](#)

# Xojo Community

Xojo has a vibrant community with lots of sources for information.

- [Blogs](#)
- [Books, Magazines, Training and Tutorials](#)
- [Discussion](#)
- [Open-Source Projects](#)
- [Free Source Code and Tools](#)
- [XojoTalk Podcast](#)

# Blogs

The following blogs have regular Xojo content. If you have or know of a Xojo blog that is not on this list, [please let us know](#) so we can get it added.

<b>Blog</b>	<b>Author</b>	<b>Description</b>
<a href="#">Official Xojo Blog</a>	Xojo	The official Xojo blog with lots of tips, news and other information about Xojo.
<a href="#">BKeeney Briefs</a>	Bob Keeney	A great source of news, opinion, reviews about Xojo.
<a href="#">MonkeyBread Software Blog</a>	Christian Schmitz	Tips, new and Xojo information.
<a href="#">My Blog: Xojo, Filemaker, iOS</a>	Tim Dietrich	Opinion, tips about Xojo, iOS and other tools.
<a href="#">Views of a Coder</a>	Thomas Tempelmann	Technical information about Xojo and programming.
<a href="#">Alwaysbusy's Corner</a>	Alain Bailleul	Strong focus on graphics and games.
<a href="#">Cully Technologies Blog</a>	Kevin Cully	Xojo tips, Linux tips and more.
<a href="#">Interpreted Keystrokes</a>	Obleo Beck	Technology news, Xojo tips.
<a href="#">Real Software Blog</a>	Xojo	The old Real Software blog still has lots of useful information about Xojo.
<a href="#">Za'atar Digital</a>	Eric de La Rochette	Xojo tips and techniques.
<a href="#">Camp Software Blog</a>	Hal Gumbert	Xojo tips for developers migrating from FileMaker.
<a href="#">Xojo-Blog</a>	Ulrich Bogun	Tagebucheinträge aus der Programmierwet. Deutsch Xojo blog.
<a href="#">Aprende Xojo</a>	Javier Rodríguez	El blog realbasic español.
<a href="#">Papayatech</a>	Adelar Duarte	Português blogue Xojo.

# Books, Magazines, Training and Tutorials

This is a list of valuable resources that can also help you learn Xojo.

<a href="#">XDev Magazine</a>	The Xojo magazine has been publishing 6 issues a year since 2002. A great source for Xojo information, tips and long-form articles on a wide variety of Xojo topics.
<a href="#">Xojo YouTube Channel</a>	A wide variety of Xojo videos and webinars (including international videos) are hosted on YouTube. You can also find and search many of the videos here in the Xojo Dev Center <a href="#">Videos section</a> .
<a href="#">BKeeney Software Training</a>	Access BKeeney Software's Xojo Training anytime. Create an account to get your free Guest Pass and start watching over 8 hours of Xojo video in just minutes. Subscribers have access to over 50 hours of video!
<a href="#">MonkeyBread Software Tutorial Videos</a>	Demos on how to use MBS plugins with Xojo.
<a href="#">Xojo 3D Tutorials</a>	Xojo3D.com provides Xojo software engineers with 3D programming resources.
<a href="#">RSLibrary</a>	A library of eBooks for Xojo.
<a href="#">VTC Real Studio Video Course</a>	An 8-hour video course showing you how to get started with Real Studio and create an actual app. Most of the concepts still apply to Xojo.
<a href="#">Real OOP with REALbasic book</a>	A book about the object-oriented capabilities of Xojo. Although the book refers to REALbasic, the OOP model works the same as Xojo.
<a href="#">Introduction to Programming with Xojo</a>	A free book that's a great introduction to the fundamentals of programming for just about anyone.

# Community Discussion

<a href="#">Xojo Forum</a>	Unlike other forums you may have used, the Xojo forum is a fun and inviting place where the users are friendly and helpful. This is a great place to ask questions, find answers and help others.
Twitter	For news, tips and Xojo information be sure to follow us on Twitter and take part in the conversation: <ul style="list-style-type: none"><li>• <a href="#">@Xojo</a></li><li>• <a href="#">@Lefebvre</a></li></ul>
<a href="#">Facebook</a>	The Xojo page on Facebook is a great place for Facebook users to stay on top of Xojo news and information.
LinkedIn	Check out the <a href="#">Xojo Developers group</a> on Linked In for news, information and to connect with other professional Xojo users.
<a href="#">Reddit</a>	The Xojo subreddit is a fun place to find and share links and information about Xojo. Be sure to subscribe if you're a Reddit user.
<a href="#">IRC</a>	If you're into IRC, check out the <a href="#">##xojo</a> channel.

# Free Source Code and Tools

The following sites all have a variety of free Xojo projects and examples for you to use. If you know of other sites with free Xojo projects and examples, please [let us know](#) so we can add them.

Site	Description
<a href="#">MacTechnologies Consulting</a>	A variety of free classes and tools (and some shareware).
<a href="#">Thomas Tempelmann's Code Samples</a>	Here you'll find some tips, examples, reusable classes and plugins for Xojo. All code and information, unless specifically noted, is free for your unlimited use.
<a href="#">Charcoal Design</a>	Free projects for use with Xojo.
<a href="#">Just Add Software</a>	Contains some of our publicly available classes for Xojo for the benefit of the user community.
<a href="#">Xojo Developer's Spot</a>	A large collection of Xojo projects and tools.
<a href="#">Eugene Dakin Xojo Examples</a>	The code on this site is free to download and contains examples that I was using on various projects.
<a href="#">Xojo Tools</a>	While working in Xojo I've made a few reusable items that you're welcome to use as well.
<a href="#">The Xojo Files</a>	On this site you will find some examples to get you started in Xojo.
<a href="#">BKeeney Software</a>	Xojo classes and tools to help with your projects.
<a href="#">DeclareSub</a>	Useful Xojo projects.
<a href="#">Ben's Software</a>	Printing modules and a math control.
<a href="#">Simcar Software</a>	Lots of useful UI controls.

# Open-Source Projects

There are a wide variety of open-source Xojo projects, many of which are listed below. If you create an open-source project or find one not listed here, please [let us know](#) so we can add it.

- [iOS Projects](#)
- [Desktop Projects](#)
- [Web Projects](#)
- [Tools](#)
- [General Libraries](#)

## iOS Projects

Project	Owner	Description
<a href="#">iOSKit</a>	Jason King	A declares implementation of many different classes for Xojo iOS apps.
<a href="#">iOSWrapper</a>	Michel Bujardet	Module that brings legacy and additional functions to Xojo iOS.
<a href="#">xojoGestures</a>	Stephen Beardslee	Xojo module for attaching UIGestureRecognizer to Views.
<a href="#">iOSLib</a>	Ulrich Bogun	A library extending Xojo's iOS feature by adding extension modules and classes.

## Desktop Projects

Project Name	Owner	Description
<a href="#">TextInputCanvas</a>	Xojo	Source for a plugin that allows developers to implement custom text input controls with international input support.
<a href="#">MacOSLib</a>	DeclareSub	Contains lots of OS X-specific functionality, including support for Mac App Store validation, X-platform preferences and many Cocoa APIs.
<a href="#">Windows Functionality Suite</a>		A collection of Windows-specific functionality.
<a href="#">XojoHTTPServer</a>	Brandon Skrtich	A http server class for Xojo / RealBasic licensed under the MIT License.
<a href="#">XojoControls</a>	Brandon Skrtich	A set of custom controls for Xojo / RealBasic licensed under the MIT License.
<a href="#">YubiKey</a>	Brandon Skrtich	A YubiKey Authentication Class for Xojo / RealBasic.
<a href="#">CustomEditField</a>	Thomas Tempelmann	Powerful text editor class made in Xojo.
<a href="#">FGSourceList</a>	Garry Pettet	A SourceList control.
<a href="#">FGScopeBar</a>	Garry Pettet	A ScopeBar control.
<a href="#">FGThumbnailCanvas</a>	Garry Pettet	A Canvas subclass for displaying image thumbnails.

Project Name	Owner	Description
<a href="#">rhCharts</a>	Rich Hatfield	Xojo Charting Class
<a href="#">Calendar Time Chooser</a>	Intelligent Visibility	Xojo Calendar and Time Chooser Picklist.
<a href="#">Logging Class</a>	Intelligent Visibility	Xojo Class for Syslog Style File Logging.
<a href="#">TELNET Class</a>	Intelligent Visibility	This is the repository for the TELNET Class for Xojo.
<a href="#">AutoCompleteTextField</a>	Intelligent Visibility	A custom Textfield that offers Auto Complete capabilities.
<a href="#">Kaju</a>	Kem Tekinay	Xojo code for implementing self-updating apps.
<a href="#">Xojo Colors Module</a>	Nocturnal Coding Monkeys	This is a Xojo Module for Colors. This makes it easy to assign colors to Windows, Container Controls, and other Controls without having to memorizing the RGB codes for various colors.
<a href="#">Cool Loading Wheel Control</a>	Intelligent Visibility	This is a "loading wheel" that is very cool looking and has a twitter bootstrap feel.
<a href="#">Mojo</a>	DeclareSub	Mojo is a library of items missing from the Xojo framework.
<a href="#">Xojo-EasyData</a>	Camp Software	Xojo EasyData for FileMaker Developers - Making Xojo Database Apps Easier.
<a href="#">iChart123</a>		Create professional looking charts using 100% Xojo code.
<a href="#">ChartPart</a>	Kevin Cully	ChartPart is an open-source native Xojo class that you drop into your projects for when you need elementary charting capabilities. ChartPart works on Linux, Mac, and Windows. Chart types available are bar charts, stacked bar charts, pie charts, and line charts.

## Web Projects

Project	Owner	Description
<a href="#">KLTableViewControl</a>	Francisco Lobo	This class is a view controller to manage cells (table). It was designed from the ground up for speed and performance. It provides several options to help reduce the loading time (like lazy loading of cells).
<a href="#">KLNotificationView</a>	Francisco Lobo	A simple to implement and use notification view for web applications.

## Tools

Project	Owner	Description
<a href="#">Xojo Format Code</a>	Jeremy Cowgar	Code formatter written in XojoScript for Xojo.

Project	Owner	Description
<a href="#">Profile-Reader</a>	Kem Tekinay	Displays Xojo Profile Code reports.
<a href="#">OpenLingua</a>	Thomas Tempelmann	A clone of Xojo's Lingua application for localizing your apps.
<a href="#">XsEdit</a>	Kem Tekinay	A XojoScript editor.
<a href="#">XPT</a>	Jeremy Cowgar	Xojo Project Tool - tool to manipulate the manifest files for Xojo.
<a href="#">pygments-xojo</a>	Charles Yeomans	Pygments-Xojo adds support for the Xojo language to the Pygments syntax highlighting package.

## General Libraries

Project	Owner	Description
<a href="#">XojoUnit</a>	Xojo	Unit Testing framework for Xojo.
<a href="#">Storm</a>	Paul Lefebvre	Object-relational manager (ORM) for SQLite and Xojo.
<a href="#">XojoSimpleSockets</a>	Brandon Skrtich	A Open Source Replacement for EasyTCPSocket and EasyUDPSocket.
<a href="#">XojoUtilities</a>	Brandon Skrtich	A set of utility classes for Xojo / RealBasic licensed under MIT License.
<a href="#">XoDrill</a>	Nocturnal Coding Monkeys	This a a Xojo class to use the Mandrill APIs from MailChimp. You must open an account with Mandrill ( <a href="https://mandrillapp.com">https://mandrillapp.com</a> ) and get an API key.
<a href="#">DropBoxAPI</a>	Intelligent Visibility	Provides programatic integration with Dropbox directly (not through sync services). This project is a work in progress as I am not implemented every API call in the Dropbox Core API to start.
<a href="#">Xojo-OpenLib</a>	Camp Software	An open library of methods and functions for Xojo.
<a href="#">SQLdeLite</a>	1701 Software	SQL components for accelerating your application development with Xojo.
<a href="#">TT's Persistent Objects</a>	Thomas Tempelmann	These classes make it possible to easily store the properties of select classes in a database, without having to deal with RecordSets and all the other DB specifica such as SQL commands for creating tables or deleting rows.
<a href="#">Data Serialization</a>	Kem Tekinay	Xojo class to serialize and deserialize classes via JSON.
<a href="#">MongoDB Xojo Driver</a>	Alwyn Bester	This is the MongoDB driver for the Xojo programming language. The current version of the driver only supports connections to a single server. Support for replica sets and sharding will be added in future versions of the driver.

---

<b>Project</b>	<b>Owner</b>	<b>Description</b>
<a href="#"><u>XojoMongo</u></a>	Patrick Perroud	XojoMongo is a Xojo middleware module that connects Xojo applications to MongoDB instances providing basic CRUD operations to NoSQL databases.
<a href="#"><u>Xojo Option Parser</u></a>	Jeremy Cowgar	Parses command line parameters to your Xojo app.
<a href="#"><u>JSONItem MTC</u></a>	Kem Tekinay	A drop-in, faster replacement for the native Xojo JSONItem class.
<a href="#"><u>OpenCV</u></a>	François Jouen	Use OpenCV (open-source Computer Vision) with Xojo.
<a href="#"><u>classPreferences</u></a>	Mike Charlesworth	A cross platform preferences class for Xojo using SQLite.
<a href="#"><u>Slack</u></a>	Xojo	A Xojo library for communicating with Slack

# XojoTalk



Join Developer Evangelist Paul Lefebvre and special Xojo community guests as they talk about Xojo, technology and more.

Subscribe to XojoTalk in your favorite podcast app, iTunes or via email.

- [Podcast App link](#)
- [RSS Reader link](#)
- [iTunes link](#)
- [Email Notification](#)

## Episodes

#	Title	Release Date	Description
015	<a href="#">Type my password like an animal</a>	July 1, 2015	Paul talks with Justin Elliott, IT and Development Manager at Penn State University.
014	<a href="#">Software Maniacs</a>	June 16, 2015	Paul talks with Ken Whitaker, the Managing Director of Leading Software Maniacs, an expert in leadership, project management and Agile development.
013	<a href="#">Hockey Programmer</a>	June 2, 2015	Paul talks with Mike Cotrone, owner of Intelligent Visibility, makers of some powerful networking apps that are all created with Xojo.
012	<a href="#">Shipping is a feature</a>	May 26, 2015	Paul talks with Paul Levine, maker of EverWeb, the Mac (and soon Windows) Web site builder.
011	<a href="#">The forum is my personal training ground</a>	April 9, 2015	Paul talks with Tim Hare of Telios Systems Company.
010	<a href="#">Strap a Mac Pro to my wrist</a>	March 24, 2015	Paul talks with Tim Dietrich a newcomer to the Xojo development world.
009	<a href="#">It's my dog</a>	March 11, 2015	Paul talks with Xojo Senior Engineer Norman Palardy.

#	Title	Release Date	Description
008	<a href="#">Really Wicked</a>	February 25, 2015	Paul talks with Hal Gumbert, Xojo developer and owner of Camp Software.
007	<a href="#">Snowgun</a>	February 10, 2015	Paul talks with Xojo developer and Linux aficionado Kevin Cully.
006	<a href="#">I have no idea what you're talking about</a>	January 27, 2015	Paul talks with long-time Xojo developer Kem Tekinay.
005	<a href="#">Where's My Column?</a>	Jan 15, 2015	Paul talks with Marc Zeedar, publisher of xDev, the Xojo Developer Magazine. Topics include the magazine, publishing formats, iOS, pricing apps, iOS, the App Store and more.
004	<a href="#">Grizzled Veterans</a>	December 17, 2014	Paul talks with Xojo engineer Travis Hill.
003	<a href="#">Developer of Long Standing</a>	November 11, 2014	Paul talks with Eric Gibbon, a long-time Xojo developer and one of the organizers of the Xojo Users' Group UK Conferences.
002	<a href="#">That's A Lot Of Tens</a>	October 7, 2014	Paul talks with Bob Keeney, a long-time member of the Xojo community and founder of BKeeney Software.
001	<a href="#">Shoving Phones In Their Pockets</a>	September 23, 2014	In this initial episode of the XojoTalk podcast, Xojo Developer Evangelist Paul Lefebvre is joined by Xojo founder and CEO Geoff Perlman. Topics include Xojo for iOS, Microsoft buying Mojang (Minecraft), the 2015 Xojo Developer Conference and the Apple iPhone, Pay and Watch event.

# International Resources

Xojo resources and documentation are available in a variety of languages. Though the [Xojo IDE is English-only](#) (with the exception of Japanese & Chinese), we strive to provide support solutions in many languages. We have Xojo Evangelists for Spanish, French, German and Italian and you can find their contact information in the designated language pages in this section. Japanese support is provided by Grape City.

We are currently looking for an experienced Xojo developer who speaks both Portuguese and English for an exciting opportunity as the Xojo Evangelist for Portuguese speakers. If you are interested in this position please email Xojo at [hello@xojo.com](mailto:hello@xojo.com).

- [Español](#)
- [Deutsche](#)
- [Italiano](#)
- [Français](#)
- [Japanese](#)
- [Português](#)
- [Nederlands](#)

# Recursos En Español

## Xojo Spanish Evangelist: Javier Menendez



Javier@xojo.com

Desarrollador, consultor y formador con más de 25 años de experiencia en el mundo de las TI. Uso Xojo desde 1999 y he creado y publicado con este lenguaje todo tipo de utilidades y aplicaciones tanto comerciales como shareware para OS X, Windows y Linux, así como plug-ins. Al mando de [AprendeXojo.com](http://AprendeXojo.com).

## Guía del Usuario Español

- Guía del Usuario Libro 1: Fundamentos, [[PDF](#), [iBooks](#)]
- Guía del Usuario Libro 3: Framework, [[PDF](#), [iBooks](#)]
- Guía del Usuario Libro 4: Despliegue, [[PDF](#), [iBooks](#)]

## Inicio Rápido Español

- Desktop QuickStart [[PDF](#), [iBooks](#)]
- Web QuickStart [[PDF](#), [iBooks](#)]
- iOS QuickStart [[PDF](#)]

## Tutoriales Español

Los tutoriales te guían en la creación de una aplicación de escritorio o web.

- Desktop Tutorial [[PDF](#), [iBooks](#)]
- Web Tutorial [[PDF](#), [iBooks](#)]

## Recursos Adicionales

### Videos Español

### Xojo Foro Español

## Blog

- [Spanish: AprendeXojo](#)

## Podcast

- [ApredeXojo Podcast Español](#)

## Libros

- [Programación multiplataforma Xojo](#)
- [OOP Xojo: Lector RSS \(HTTPSocket, XMLDocument, XPath, ExReg\)](#)

## Tutoriales Adicionales

- [Sobrecarga de Operadores en Xojo \(iBooks Store\)](#)
- [Clases e Interfaces de Clase en Xojo \(iBooks Store\)](#)

Twitter: [@XojoES](#)

[Facebook](#)

## Deutsche Ressourcen

### Xojo-Evangelist Deutschland: Ulrich Bogun



Ulrich@xojo.com

Im Hauptberuf Mediengestalter, wurden meine brachliegenden Informatik-Kenntnisse dank Xojo wieder zum Leben gebracht. Meine berufsbedingte Mac-Affinität hat dann auch dafür gesorgt, dass meine Lieblings-Zielplattformen Mac OS X und iOS wurden, was zwangsläufig zur Erstellung eigener, frei verfügbarer Xojo-Erweiterungsbibliotheken für diese Systeme führte. Gerne würde ich Xojo auch verstärkt als ernsthafte Spielentwicklungs-IDE eingesetzt sehen, wo es m.E. noch etwas unterrepräsentiert ist. Als Xojo-Entwicklerevangelist für den deutschsprachigen Raum gilt mein besonderes Interesse auch der Einführung von Xojo als Entwicklungsplattform für den Schul-Informatikunterricht. Wann immer Sie Fragen haben, die dank Sprachbarriere oder lokaler Gepflogenheiten schwer im internationalen Forum zu klären sind: Ich bin gerne für Sie da.

## Deutsche Schnellstartanleitungen

- Desktop QuickStart [[PDF](#), [iBooks](#)]
- Web QuickStart [[PDF](#), [iBooks](#)]
- iOS QuickStart [[PDF](#), [ePub](#)]

## Deutsche Tutorials

In den Tutorials werden Sie Schritt für Schritt durch die Erstellung einer anspruchsvolleren Desktop- oder Web-Applikation geführt.

- Desktop Tutorial [[PDF](#), [iBooks](#)]

- Web Tutorial [[PDE](#), [iBooks](#)]

## Zusätzliche Ressourcen

[Xojo-Videos auf Deutsch](#)

[Xojo Forum - Deutsch](#)

### Blogs

- [xojoblog.me](#)

Twitter: [@XojoDeutsch](#)

# Italian Resources

## Xojo Italian Evangelist: Antonio Rinaldi



Antonio@xojo.com

Sono un ingegnere che lavora da oltre 20 anni nel settore IT come progettista, sviluppatore, consulente e formatore. Utilizzo Xojo da moltissimi anni per creare soluzioni di ogni tipo per tutte le piattaforme possibili.

## Guide rapide in Italiano

- Desktop QuickStart [[PDF](#), [iBooks](#)]
- Web QuickStart [[PDF](#), [iBooks](#)]
- iOS QuickStart [[PDF](#)]

## Tutorial in Italiano

I tutorial ti guidano nella creazione di un'applicazione più avanzata per il desktop o il web

- Desktop Tutorial [[PDF](#), [iBooks](#)]
- Web Tutorial [[PDF](#), [iBooks](#)]

## Risorse Aggiuntive

[Video in italiano](#)

[Forum Xojo in Italiano](#)

[Linee guida per i 64 bit](#)

Twitter: [@XojoIT](#)

[Facebook](#)

## Ressources en français

### Xojo French Evangelist: Stephane Pinel

Développeur, consultant et formateur depuis près de vingt ans, Il est passionné par Xojo depuis sa toute première version. Stéphane intervient dans la traduction de ressources Xojo en français, la relation avec les développeurs Xojo francophones et effectue des présentations en France et ailleurs en Europe. Il anime également des communautés Xojo francophones sur [Twitter](#) et [Facebook](#).



[Stephane@xojo.com](mailto:Stephane@xojo.com)

### Guides de démarrage en français

- Guide de démarrage Desktop [[PDF](#), [iBooks](#)]
- Guide de démarrage Web [[PDF](#), [iBooks](#)]
- Guide de démarrage iOS [[PDF](#), [iBooks](#)]

### Ressources additionnelles

Vidéos en français

Forum Xojo en français

Twitter: [@XojoFR](#)

[Facebook](#)

Please visit [Xojo's Japanese Developer Center](#).

# Recursos Portugueses

## QuickStarts Brazilian Portuguese

- Desktop QuickStart [[PDF](#), [iBooks](#)]
- Web QuickStart [[PDF](#), [iBooks](#)]

## Additional Resources

### Videos

[Xojo Forum - Português](#)

### Blogs

- [Papayatech Blog](#)

Twitter: [@XojoPT](#)

## Resources - Nederlands

[Xojo Forum - Nederlands](#)

[Facebook Community](#)

[Xojo Nederland Blog](#)

# Deprecations and Removals

## Table of Contents

- [Overview](#)
- [Operating Systems](#)
- [Deprecated Features](#)

## Overview

As time marches on, some older features of Xojo may no longer be relevant, may be unsupported by the OS or may be replaced with a newer feature. In such cases, the Xojo feature will be marked as De-emphasized, Deprecated and then finally removed.

- **De-emphasized** means that the item's use is no longer encouraged and alternatives should be investigated. Bugs will not be fixed unless they cause fatal errors. At some point after an item is de-emphasized, it will be deprecated.
- **Deprecated** means that an item is no longer supported, although the item still works and remains available for use. If your projects require this item, you should consider finding another solution. Bugs are not typically fixed for deprecated items.
- Deprecated item will eventually be **removed** from Xojo, typically about a year after they were deprecated (although this may be sooner or later if the situation warrants it).

## Operating Systems

Some older operation systems are no longer supported by Xojo. This table shows you the most recent Xojo release that works with various older operating systems.

Type	OS	Last Supported Release
<b>IDE</b>		
	Windows XP	2013r4.1
	OS X 10.6	2013r3.3
	PowerPC	2010r3.2
<b>OS X Builds</b>		
	Carbon	2014r2
	OS X 10.6	2013r3.3
	OS X 10.5	2012r2.1
	OS X 10.4	2011r4.3
	OS X 10.3	2009r4

Type	OS	Last Supported Release
	OS X on PowerPC	2011r3
	OS 9	2007r3
<b>Windows Builds</b>		
	Windows XP	2015r2.4
	Windows XP SP2	2013r4.1
	Windows 2000	2010r3.2
	Windows 98	2007r3
	Single-File EXE	2008r1
<b>Linux Builds</b>		
	CentOS 5.x	2013r3.3

## Deprecated Features

### 2016 Release 1

#### Deprecations

- OS X 10.7
- OS X 10.8

#### Removals

### 2015 Release 4

#### Deprecations

- Graphics.Pixel

#### Removals

### 2015 Release 3

#### Deprecations

- WindowPtr (Use Window.Handle)
- Short (use Int16)
- Support for iOS 7 apps
- Windows Vista

#### Removals

- Inline68K constant
- Xojo no longer supports compiling for Windows XP.

## 2015 Release 2

### Deprecations

### Removals

- BinaryStream.ReadLong, ReadShort, ReadByte, WriteLong, WriteShort and WriteByte methods (deprecated in 2006r1). Use data-type specific methods instead.

## 2015 Release 1

### Deprecations

- TargetHasGUI, TargetPPC, TargetPowerPC, TargetMacOSClassic, Target68K

### Removals

- Xojo IDE no longer supported on Windows XP.

## 2014 Release 1

### Deprecations

- Xojo ID no longer supported on Windows XP.

### Removals

- EditableMovie
- QT3DAudio, QTEffect, QTEffectSequence, QTGraphicsExporter, QTSoundTrack, QTTrack, QTUserData
- System.QuickTime
- MoviePlayer.EditingEnabled, MoviePlayer.PlayerType, MoviePlayer.PlaySelection, MoviePlayer.QTVRNode, MoviePlayer.QTVRPan, MoviePlayer.QTVRPanMix, MoviePlayer.QTVBPanTiltSpeed, MoviePlayer.QTVRTilt, MoviePlayer.QTVRTiltMax, MoviePlayer.QTVRTiltMin, MoviePlayer.QTVRZoom, MoviePlayer.QTVRZoomMax, MoviePlayer.QTVRZoomMin, MoviePlayer.QTVRZoomSpeed, MoviePlayer.Rate, MoviePlayer.SelLength, MoviePlayer.SelStart, MoviePlayer.Clear, MoviePlayer.Copy, MoviePlayer.Cut, MoviePlayer.Paste, MoviePlayer.QTVRHotSpotCount, MoviePlayer.QTVRHotSpotID, MoviePlayer.QTVRNodeTypeObject, MoviePlayer.QTVRNodeTypePanorama, MoviePlayer.QTVRTriggerHotSpot, MoviePlayer.QTVRTriggerHotSpotNames, MoviePlayer.Undo
- FolderItem.CreateMovie, FolderItem.OpenEditableMovie
- GetQTCrossFadeEffect, GetQTGraphicsExporter, GetQTSMPTEEffect
- Plugins SDK: Functions and types relating to

QuickTime have been removed from the SDK.

## 2013 Release 4

### Deprecations

- `HTTPSecureSocket.DefaultPort`
- `System.QuickTime`
- All QuickTime classes and related methods/properties

### Removals

## 2013 Release 1

### Deprecations

- OS X Carbon builds
- `SpecialFolder.AppleMenu`
- `DisableAutoWaitCursor` pragma
- `TextArea.Open`
- `TextArea.Save`
- `TextArea.SetTextAndStyle`
- `TextArea.TextStyleData`
- `WebFile.FileDownloadDelegate`
- `FolderItem.AbsolutePath` (use `FolderItem.NativePath` instead)
- `REALSQLdatabase` (use `SQLiteDatabase`)

### Removals

- `BooleanProvider`
- `ListInterface`
- `StringInterface`
- `StringProvider`
- `DataAvailableProvider`
- `BindingInterface`
- `BindPartInterface`
- `ListDataProvider`
- `ListDataNotifier`
- `ListDataNotificationReceiver`
- `TupleInterface`
- `EnablingBinder`
- `ActionBinder`
- `StringBinder`

- ListBinder

## 2012 Release 2

### Deprecations

#### Removals

- RB3D
- ToolbarItem
- Sprite Surface
- System.Pixel

## 2012 Release 1

### Deprecations

- System.Keyscript
- System.Pixel
- Inline68K

#### Removals

- EditField (use TextField and TextArea)
- Constructors named after classes (use Constructor)
- Destructors named after classes (use Destructor)
- DatabaseRecord.GetIndString (use DatabaseRecord.Column)
- FolderItem.SaveAsMactintoshPICT [sic] (use Picture.Save, with Format = Picture.SaveAsMacintoshPICT)
- FolderItem.SaveAsMactintoshRasterPICT [sic] (use Picture.Save, with Format = SaveAsMacintoshRasterPICT)
- FolderItem.CreateBinaryFile (use BinaryStream.Create)
- FolderItem.OpenAsBinaryFile (use BinaryStream.Open)
- FolderItem.OpenBinaryStream (use BinaryStream.Open)
- FolderItem.CreateBinaryStream (use BinaryStream.Create)
- FolderItem.OpenAsTextFile (use TextInputStream.Open)
- FolderItem.CreateTextFile (use TextOutputStream.Create)
- FolderItem.AppendToTextFile (use TextOutputStream.Append)
- BinaryStream.HandleTypeMacFileRefNum
- BinaryStream.HandleTypeMacFileSpecPointer
- ChasingArrows
- LittleArrows
- ContextualMenu
- NewAppleEvent
- Serial.Port (use Serial.SerialPort)

---

## 2011 Release 4

### Deprecations

- Window.Graphics and Canvas.Graphics
- System.Pixel

### Removals

- Compiling for PowerPC & Universal Binary

## 2011 Release 3

### Deprecations

- GetQTCrossFadeEffect
- GetQTGraphicsExporter
- GetQTSMPTEEffect
- QT3DAudio
- QTEffect
- QTEffectSequence
- QTGraphicsExporter
- QTSoundTrack
- QTTrack
- QTUserData
- QTVideoTrack

### Removals

## 2011 Release 2

### Deprecations

- EditableMovie

### Removals

## 2011 Release 1

### Deprecations

- FolderItem.DesktopFolder (use SpecialFolder.Desktop)
- FolderItem.TrashFolder (use SpecialFolder.Trash)
- FolderItem.SharedTrashFolder (use SpecialFolder.Trash)
- FolderItem.TemporaryFolder (use SpecialFolder.Temporary)

### Removals

---

## 2010 Release 5

### Deprecations

- FolderItem.MacDirID
- FolderItem.MacVRefNum

### Removals

## 2010 Release 4

### Deprecations

- StaticText class (use Label class)

### Removals

## 2010 Release 3

### Deprecations

- FolderItem.OpenAsPicture (use Picture.Open)
- FolderItem.SaveAsPicture (use Picture.Save)
- NewAppleEvent (use AppleEvent constructor)
- NewAppleEventTarget (use AppleEventTarget.Create)
- NewMemoryBlock (use MemoryBlock constructor)
- NewPicture (use Picture constructor)

### Removals

## 2010 Release 1

### Deprecations

- RectControl.NewDragItem (use DragItem constructor)
- Window.NewDragItem (use DragItem constructor)

### Removals

---

## 2009 Release 5

### Deprecations

- FolderItem.AppendToTextFile (use TextOutputStream.Append)

### Removals

- Minimum Mac OS is now OS X 10.4

## 2009 Release 4

### Deprecations

- FolderItem.OpenBinaryStream (use BinaryStream.Open)
- FolderItem.CreateBinaryStream (use BinaryStream.Create)
- FolderItem.OpenAsTextFile (use TextInputStream.Open)
- FolderItem.CreateTextFile (use TextOutputStream.Create)

### Removals

## 2009 Release 3

### Deprecations

- EditField (use TextField and TextArea)
- FolderItem.CreateBinaryFile (use BinaryStream.Create)
- FolderItem.OpenAsBinaryFile (use BinaryStream.Open)

### Removals

## 2008 Release 5

### Deprecations

- DatabaseRecord.GetIndString (use DatabaseRecord.Column)
- ResourceFork

### Removals

- ApplicationSupportFolder (use SpecialFolder.ApplicationData)
- DesktopFolder (use SpecialFolder.Desktop)
- DocumentsFolder (use SpecialFolder.Documents)
- PreferencesFolder (use SpecialFolder.Preferences)
- SystemFolder (use SpecialFolder.System)

- TemporaryFolder (use SpecialFolder.Temporary)
- TrashFolder (use SpecialFolder.Trash)
- SpecialFolder.ShutdownItems (no replacement)
- SpecialFolder.ControlPanels (no replacement)

## 2008 Release 4

### Deprecations

- FolderItem.SaveAsMactintoshPICT [sic] (use Picture.Save, with Format = Picture.SaveAsMacintoshPICT)
- FolderItem.SaveAsMactintoshRasterPICT [sic] (use Picture.Save, with Format = Picture.SaveAsMacintoshRasterPICT)

### Removals

- NewREALDatabaseOldFormat
- TabPanel.Facing property and associated constants
- DebugDumpObjects
- Window.BalloonHelp, App.BalloonHelpVisible, RectControl.BalloonHelp, RectControl.DisabledBalloonHelp, MenuItem.BalloonHelp, MenuItem.DisabledBalloonHelp
- MenuItem.Bold, MenuItem.Italic, MenuItem.Underline
- Window.FloaterProcess

## 2008 Release 3

### Deprecations

- Serial.Port (use Serial.SerialPort)

### Removals

## 2008 Release 2

### Deprecations

- NewREALDatabaseOldFormat
- TabPanel.Facing property and associated constants
- DebugDumpObjects
- ChasingArrows
- LittleArrows
- Constructors named after classes (use Constructor)
- Destructors named after classes (use Destructor)

### De-Emphasized

- RB3D

- NotePlayer

### Removals

- AppleMenuFolder (use SpecialFolder.AppleMenu)
- ExtensionsFolder (use SpecialFolder.Extensions)
- StartupItemsFolder (use SpecialFolder.StartupItems)
- FontsFolder (use SpecialFolder.Fonts)
- ShutdownItemsFolder (use SpecialFolder.ShutdownItems, through 2008r4.2)
- ControlPanelsFolder (use SpecialFolder.ControlPanels, through 2008r4.2)
- CSV Plugin (use third-party replacements)

## 2008 Release 1

### Deprecations

- SpriteSurface (try SuperSpriteSurface from TinRocket instead)
- NewAppleEvent (use AppleEvent constructor)

### Removals

- PPPSocket
- CSV Plugin

## Earlier Releases

### Deprecations

- DBF plugin
- DTF plugin

### Removals

# Send Us Your Feedback

## Reporting Bugs and Making Feature Requests

If you think you have found a bug in Xojo or have a feature request, please let us know about it. The way to report bugs or make feature requests is with the Feedback application. Feedback is designed to gather all the necessary information that helps us track down bugs and implement feature requests.

For each bug or feature request reported (called a case), you will receive a confirmation message via email with the case number. When cases are updated, you get an email telling you about the change.

You can launch the Feedback application directly from within Xojo. Just choose Help → Xojo Feedback or click the Feedback button on the toolbar. If you have already installed Feedback this will open the Feedback application. If you have not yet installed Feedback, your default web browser will open the [Xojo download page](#) where you can download Feedback for your OS.

## Using the Feedback App

The Feedback app has three main areas: the search field, the sidebar and the case area.

## Searching and Creating New Cases

**i** You have to search before you can create a new case.

The search field is where you search for cases. **You have to search for cases before you can create a new case.** This is to help people perhaps find an existing case on the same topic and help prevent duplicates. If you want to create a new case, just enter the title for the case as the search criteria to see a list of similar cases. After you search, you will have the ability to save your search (into the sidebar Folders section) or to create a new case. You can also just double-click any case in the search results to view the case.

### Search Tips

- When searching, "AND" is always implied between keywords
- You can use the "-" character to exclude specific words from the search: toolbar -linux
- The "|" is used to mean "OR": Str | StrB
- You can use parenthesis to group items: listbox -(hierarchical | checkbox)
- Use quotes to search for phrases: "windows build"
- The index skips common words and abbreviates common suffixes

## Sidebar

### My Feedback

The My Feedback section of the navigator gives you a quick overview of cases that are relevant to you.

- Dashboard: The Dashboard gives you an overview of your cases and their ranks.

- **My Top Cases:** Use this screen to rank cases. Licensed users of Xojo have their cases ranked more heavily. Drag open cases from the sidebar into the appropriate position to rank a case.
- **My Favorites:** These are cases that you have created or cases that you have marked as a favorite by clicking the "star" button on the case action bar.
- **My Cases:** These are the cases you created.
- **Participating In:** These are the cases you created and any cases that you have commented on.

## Shared Feedback

- **Search Results:** The results from the most recent search.
- **Top Cases:** Displays the top ranked cases based on everyone's feedback rankings.
- **Release Notes:** Displays release notes for Xojo.

## Folders

You can create your own folders and then add cases to them, which might be useful for tracking. Use the action bar below the sidebar to add a folder. You can drag open cases from the sidebar to a folder for quick access. Folders can be local or shared. If you create a shared folder, you have the option of providing a password. Right-click on a shared folder to get its "sharing link" to send to others that you want to be able to view the folder. Any search results that you choose to save also appear in the Folders section.

## Case Area

The case area can show information about a case or cases. You can also click on the Dashboard in the sidebar to show an overview of cases you have created or ranked.

When a list of cases is displayed, such as for search results, you can customize the visible columns by using View->Customize List Columns feature. You can click on any of the headers to sort the list. If you want an avatar to appear next to your name on your cases and comments, create a Gravatar account and associate it with the email used by your Xojo account.

When viewing a case, you can see its original title and description along with notes added to the case. You will also be able to see the overall status of the case.

## Formatting

In the case description, you can format code snippets by using the [code] tag:

```
[code]
Dim i As Integer
i = 10
[/code]
```

## Case Status

As cases progress through the system, their status changes. Here is a list of statuses and what they mean.

- **Needs Review:** The case is in queue to be reviewed.
- **Reviewed:** The requested bug/feature is has been reviewed and is pending judgement.

- Verified: The case has been verified and has been assigned to an engineer.
- Scheduled: The case has been scheduled for development.
- Fixed: A fix for this case has been checked in and is pending verification.
- Fixed and Verified: The fix for this case has been verified and is available in the noted IDE version.
- Implemented: The feature has been completed, checked in and is pending verification.
- Implemented and Verified: The feature has been verified and is available in the noted IDE version.
- Closed: The case has been closed for the specified reason.
- Information Required: The case is waiting for additional information from the author.

## Ranking Your Cases

Use the My Top Cases selection in the Sidebar to rank cases. Drag open cases from the sidebar onto one of the rank areas to rank the case. You can rank your top 5 cases. The ranking works as follows:

- Each active license you have for Xojo counts as 1 point
- A Xojo Pro license counts for 12 points total (4 points for each license multiplied by 3)
- Assigning a case to a higher ranking multiplies your points. If you have a single license, then a 5th place rank for a case gives it 1 point. A 1st place rank for a case gives it 5 points.

Under each ranking position, you will see how many points are applied to cases.

## Creating Good Feedback Cases

It is easy to create Feedback cases, but it is not so easy to create good cases. There are four types of cases you can create: Beta Bug, Bug, and Feature Request.

- Beta Bug: A bug that was found in a current Xojo beta.
- Bug: A bug that was found in the currently shipping version of Xojo.
- Feature Request: A change or suggestion for new functionality.

What makes a good case? A good case is a case that is well defined, with reproducible steps and usually an associated project that demonstrates the issue.

It starts with a good summary. Feedback uses intelligent phonetic searching on the summary of each case. It does not search the details of the cases. A good case summary should be a few words which best describe the nature of the case.

An example of a good case summary is "HTMLViewer should use WebKit on all platforms". Summaries like "HTMLViewer doesn't work" or "WebKit" are not actionable, too short and not specific, making them difficult to find when searching.

For bugs, the details should describe the issue more clearly and then include steps to reproduce the it. Ideally you will want to also attach a sample project that demonstrates the issue. **Our statistics show that cases with sample projects are fixed faster than cases without sample projects.** If a case is created that cannot be reproduced it's status may be changed to "Closed" or "Information Required". In both those situation, you will want to provide additional information if you want the case to remain open.

For feature requests, the details should describe the problem you need to solve. Try to limit specific implementation suggestions.

## Posting Feedback Links in Forum

Sometimes it is helpful to post links to your Feedback cases for others to see so that they can mark it as a favorite or add more information. To do, so click the "gear" icon at the bottom of the Feedback case screen and choose "Copy Sharing Link" from the menu. You can then paste this link to your forum post (do not use an additional formatting with it) and it will automatically appear in your post as a link that will open Feedback and display the case.

## Troubleshooting

If you are having trouble accessing Feedback, here are some things to check:

- Download and install the latest version version.
- Verify you have correctly entered your Xojo account information.
- If you are using a Proxy, ensure the proxy information has been entered.

# How do I work with Files?

These questions are related to file handling, which make use of the [FolderItem](#) class.

- [How do I load and save data to a file?](#)
  - [How do I list files in a folder?](#)
  - [How do I upload and download files in a web app?](#)
  - [How do I create a folder?](#)
- 

## How do I load and save data to a file?

There are several classes to help you load and save your data. If you are working with text files, you use the [TextInputStream](#) class to load data and the [TextOutputStream](#) class to save data. If you are working with binary data (or data with a fixed-size file format) you can use the [BinaryStream](#) class.

For more information:

- Refer to the [TextInputStream.Open](#) method for sample code that shows you how to read a text file.
  - Refer to the [TextOutputStream.Create](#) method for sample code that shows you how to create a text file.
  - Watch the [Fun with Files](#) webinar.
- 

## How do I list files in a folder?

The [FolderItem](#) class is used to access the file system and its [Children](#) method provides a way for you to loop through all the files contained within a [FolderItem](#) that refers to a folder.

For more information:

- Refer to the [FolderItem.Children](#) method for sample code that shows you how to add the names of all the files in the Documents folder to an array.
- 

## How do I upload and download files in a web app?

To allow users to upload files to your web app, you use the [WebFileUploader](#) control. This control provides a user interface for the user to select and add files to a list to be uploaded. You then add a button to the page that calls the [Upload](#) method of the control to actually upload the files to your web app on the server. When the upload has finished, the [UploadComplete](#) event handler is called where you can get a list of the files that were uploaded so that you can process them as necessary.

To make a file available for download, you first have to create a [WebFile](#) that refers to the file. This [WebFile](#) should be a Property of the page so that it does not go out of scope while the file is downloading, which would cause an incomplete download. You then want to set the [ForceDownload](#) property to True so that the file is downloaded rather than displayed in the browser. To start the download, you call [ShowURL](#) and pass in the URL of the [WebFile](#) that you created.

For more information:

- Examine the [WebFileUploader](#) example project (Examples/Web/Controls) that shows you how to upload files to a web app.
  - Examine the [Downloading](#) example project (Examples/Web/Downloading) that shows you how to download files from a web app.
  - Watch the [Uploading and Downloading Files with a Web App](#) webinar.
- 

## How do I create a folder?

To create a folder, you first create a [FolderItem](#) that points to where you want the folder. Then you call the [CreateAsFolder](#) method.

For more information:

- Refer to [FolderItem.CreateAsFolder](#)

## How do I use databases?

These questions are related to working with databases.

- [How do I get the number of rows in a RecordSet?](#)
- [How do I connect to a database server from an iOS app?](#)

## How do I get the number of rows in a RecordSet?

The `RecordSet.RecordCount` property can tell you the number of rows in the RecordSet for these databases:

- [SQLiteDatabase](#)
- [PostgreSQLDatabase](#)
- [OracleDatabase](#)
- [ODBCDatabase](#) (depends on the database driver you are using)

This property does not work for other database types.

Alternatively, you can create a SQL SELECT statement to return the number of rows in a query. This is done using the COUNT method. The exact syntax varies by database, but is usually something like this:

```
SELECT COUNT(*) As RowCount FROM TableName WHERE value1 = x AND value2 = y
```

With this SELECT statement, you can then check the result in RecordSet results:

```
Dim sql As String = "SELECT COUNT(*) AS RowCount FROM Table1 WHERE ID = 1"  
Dim rs As RecordSet = myDatabase.SQLSelect(sql)
```

```
If rs <> Nil And Not myDatabase.Error Then  
    Dim rowCount As Integer = rs.IdxField(1).IntegerValue  
End If
```

## How do I connect to a database server from an iOS app?

iOS apps should not directly connect to database servers like desktop apps do. There are several reasons for this:

- iOS devices may not have a reliable or fast network connection, making DB server connections unreliable.
- Maintaining a constant connection to a database server would dramatically reduce battery life.
- Apple does not include native drivers for database servers with iOS and it is not practical to embed your own.

However, you can still connect to a database server! You just have to do it with a web service. The general concept is that you would create a web service and companion API (using Xojo or another tool) that connects to the database server. The iOS apps connects to the web service (using `HTTPSocket`).

For more information, watch the 2-part video on this topic:

- [REST Web Services](#)