What's New in Xcode

Developer

Contents

SwiftObjective-C

What’s New in Xcode

Xcode is the complete developer toolset used to create apps for Apple Watch, iPad, iPhone, and Mac. The Xcode development environment bundles the Instruments analysis tool, iOS Simulator, and the OS frameworks in the form of iOS SDKs and OS X SDKs.

Xcode 6.4 Update

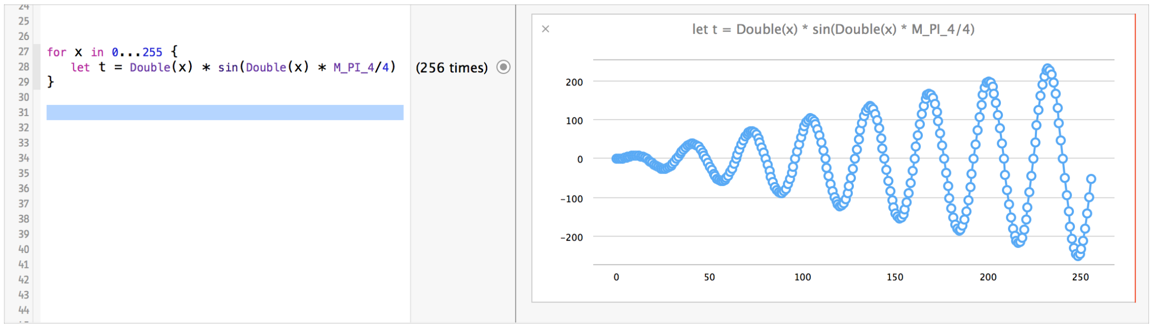
Xcode 6.4 (page 25) includes support for development on iOS 8.4.

Highlights of Xcode 6

Xcode 6 features Swift, an innovative new programming language, and extends the Xcode feature set with live visualization capabilities. Xcode 6 includes the following highlighted features:

Swift Language

- An advanced object-oriented programming language for iOS and OS X development



```
for x in 0...255 {
    let t = Double(x) * sin(Double(x) * M_PI_4/4)
}
```

```
(256 times)
```

- Xcode 6 offers full support for Swift with playgrounds, a rich documentation experience, read-eval-print loop, and other advanced features

  See more details in Swift Language (page 14).

Testing

- Testing capabilities for performance measurement
- Ability to test asynchronous code with enhanced XCTest

  See more details in Testing (page 15).

Interface Builder

- Extensive new features that support live rendering, storyboards for OS X, versatile size classes, and custom iOS fonts

  See more details in Interface Builder (page 16).

Debugger

- New view debugger, enhanced queue debugging, and new debug gauges

  See more details in Debugging (page 17).

SpriteKit and SceneKit

- Enhanced game development with new level designer and improved debugging
- Support for SpriteKit and SceneKit to work together on iOS

  See more details in SpriteKit and SceneKit (page 19).

Extensions and Frameworks

- Ability to add extensions to any app, increasing functionality
- Ability to create dynamic frameworks for iOS

  See more details in Extensions and Frameworks (page 19).

iOS Simulator

- New iOS Simulator configurations that allow saving data and settings for reuse

  See more details in iOS Simulator (page 20).

Localization

- XLIFF format support for strings localization

- Automatic base language `.strings` generation
- New Interface Builder assistant preview of alternative language UI
- Ability to run apps as they appear in other locales

  See more details in Localization (page 20).

Compiler

- Profile Guided Optimization (PGO)
- User-defined modules

  See more details in Compiler (page 20).

Instruments

- Updated user interface with new template chooser and track view
- App extension profiling support
- Profile tests to discover regression causes

  See more details in Instruments (page 21).

Xcode Server

- More complex integration scenarios with triggers
- Support for performance testing integrations
- Greater control over bot configuration and execution; bot-level statistics

  See more details in Xcode Server (page 21).

HomeKit Accessory Simulator

- Simulator for app connection with accessories in HomeKit development

  See more details in HomeKit Accessory Simulator (page 22).

For additional details on the Xcode 6 release, see *Xcode Release Notes* .

Compatibility

Xcode 6.4 requires a Mac running OS X version 10.10. It includes SDKs for OS X version 10.9, OS X version 10.10, and iOS 8.4. To develop apps targeting prior versions of OS X or iOS, see About SDKs and iOS Simulator (page 10).

Installation

You obtain Xcode 6 from the Mac App Store. It is a free download that installs directly into the Applications folder. By default, Xcode downloads developer documentation in the background for offline reading and also automatically downloads documentation updates. This behavior can be changed after installation using the Downloads preferences pane.

Developer Resources

The Apple Developer Programs provide access to the App Store and Mac App Store, additional support and documentation, and provisioning resources for testing and deployment on Apple Watch, iPad, iPhone, and iPod touch devices. For information about Apple Developer Programs, visit the Apple Developer website at http://developer.apple.com/.

For discussions about any Apple developer software, including prerelease products, visit the Apple Developer Forums at http://devforums.apple.com/.

For the latest security information, visit http://support.apple.com/kb/HT1222.

For more detailed information on a release, see the complete Xcode release notes available from the Xcode Help menu.

About SDKs and iOS Simulator

A software development kit (SDK) is a collection of frameworks (libraries, headers, and resources) that represent the API for a specific iOS or OS X version. Most of the functionality your app gets from an SDK is actually provided by the host operating system, which makes the right Base SDK and OS Deployment Target settings critical for app compatibility. Xcode automatically builds with the latest SDK and targets the latest OS.

If your app doesn't require the latest OS features, you can configure it to run on a previous version of iOS or OS X using the OS Deployment Target option in the Xcode Project settings. If your project was created in an older version of Xcode, you can let Xcode update your project. For details on this feature, see Project Modernization.

For iOS, Xcode automatically switches between the iOS Simulator SDK and the device SDK, depending on where you intend to run your app. You don't need to select these settings manually.

Important: Before submission to the App Store you must test on a device running your supported target OS version or versions.

Project Modernization

When you open a project, Xcode evaluates it to see whether any settings should be updated. This feature provides an easy way to make sure your projects conform to the latest SDKs and best practices.

Open the issue navigator to see whether anything in your project needs to be updated. You can also select the project in the project navigator and choose Editor > Validate Settings.

If the issue navigator lists modernization issues, click the issue to see a dialog that explains the updates that should be made and lets you perform any or all of them.

After you have clicked Perform Changes, regardless of whether you choose to make all the changes, Xcode does not show the warning again. To rerun the check, select your project in the project navigator and choose Editor > Validate Settings.

New Features in Xcode by Release

Chapter articles are listed by major revision. Minor update release information is appended in the chapter for the major revision.

Xcode 6 adds support for development on iOS 8 and OS X v10.10.

Relevant Chapter: New Features in Xcode 6 (page 13)

Xcode 5.1 adds support for development on iOS 7.1.

Relevant Chapter: New Features in Xcode 5.1 (page 26)

Xcode 5 adds support for development on iOS 7 and OS X v10.9.

Relevant Chapter: New Features in Xcode 5 (page 28)

Xcode 4.6 adds support for development on iOS 6.1.

Relevant Chapter: New Features in Xcode 4.6 (page 37)

Xcode 4.5 adds support for development on iOS 6.

Relevant Chapter: New Features in Xcode 4.5 (page 39)

Xcode 4.4 adds support for new Objective-C language features and supports development on OS X v10.8.

Relevant Chapter: New Features in Xcode 4.4 (page 40)

Xcode 4.3 adds enhancements to Xcode installation, improves operations and workflow, and support development on iOS 5.

Relevant Chapter: New Features in Xcode 4.3 (page 44)

Xcode 4.2 adds enhancements to the features and workflow of Xcode 4.1 to support development on iOS 5.

Relevant Chapter: New Features in Xcode 4.2 (page 47)

Xcode 4.1 adds enhancements to the features and workflow of Xcode 4.0 and, when running on OS X v10.7, implements user interface features standard in OS X 10.7 such as full-screen windows.

Relevant Chapter: New Features in Xcode 4.1 (page 51)

See Also

To learn more about Xcode, see *Xcode Overview*.

To learn more about the tasks and workflow required to develop and distribute OS X and iOS apps, see *App Distribution Guide*.

New Features in Xcode 6

- Xcode 6.0 (page 13) includes SDKs for OS X version 10.9, iOS 8, and other enhancements.
- Xcode 6.0.1 (page 22) is a maintenance update responding to developer input and Apple SQA testing.
- Xcode 6.1 (page 22) adds the SDK for OS X version 10.10 and Swift language development for OS X.
- Xcode 6.1.1 (page 23) is a maintenance update with bug fixes and performance improvements.
- Xcode 6.2 (page 23) Xcode 6.2 adds support for iOS 8.2 and WatchKit.
- Xcode 6.3 (page 23) adds support for iOS 8.3, Swift 1.2, and many other new features..
- Xcode 6.3.1 (page 25) is a maintenance update with bug fixes and performance improvements.
- Xcode 6.3.2 (page 25) fixes a Swift compilation speed regression
- Xcode 6.4 (page 25) includes support for development on iOS 8.4.

Xcode 6.0

Xcode 6 includes Swift, an innovative programming language with an interactive work area called a *playground* . Developers can directly manipulate and experiment with Swift code live—enter the code for a Bézier path in the playground and watch the path drawn directly beside the code. Perfect new code within the playground, then easily promote that code into your main project.

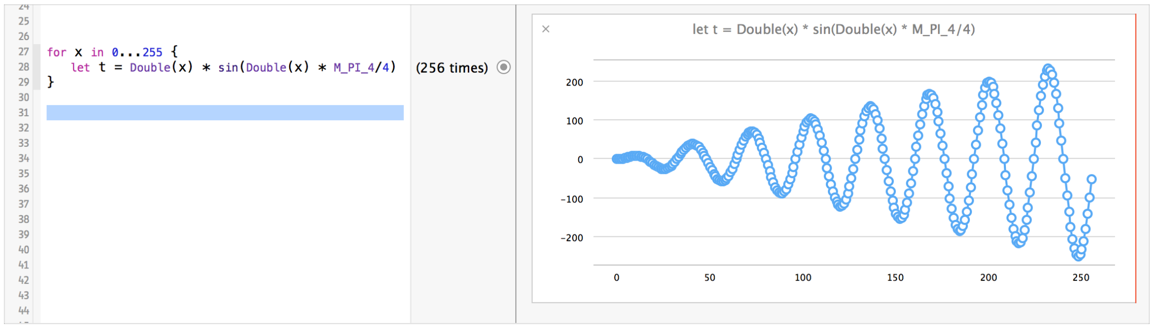
Xcode 6 extends the Xcode feature set with new live visualization capabilities. For example, view debugging pauses a running app and then explodes all the UI layers into a 3D visualization, making it easy to understand how the interface is drawn. Live rendering within Interface Builder displays your handwritten UI code as you design, so that you can edit your view's code and the IB rendering changes instantly. And the preview assistant now shows your app in different languages with only a mouse click.

Xcode 6 requires a Mac running OS X version 10.9.4 or later or 10.10. It includes SDKs for OS X version 10.9 and 10.10, and iOS 8.

Xcode 6 includes the following highlighted features.

Swift Language

Swift is a advanced object-oriented programming language for iOS development. Swift is modern, powerful, expressive, and easy to use.



```
for x in 0...255 {
    let t = Double(x) * sin(Double(x) * M_PI_4/4)
}
```

(256 times)

- **Fast.** Swift compiles and is optimized with the advanced code analysis in LLVM to create high-performance apps.
- **Complete platform.** Access all of the Cocoa Touch frameworks with Swift.
- **Safe by design.** Eliminate huge categories of bugs, crashes, and security holes.

  Swift pairs increased type safety with type inference, restricts direct access to pointers, and automatically manages memory using ARC, making it easy for you to use Swift and create secure, stable software. Other language safety related features include mandatory variables initialization, automatic bounds checking to prevent overflows, conditionals that break by default, and elimination of pointers to direct memory by default.
- **Modern.** Write, debug, and maintain less code, with an easy to write and read syntax, and no headers to maintain.

  Swift includes optionals, generics, closures, tuples, and other modern language features. Inspired by and improving upon Objective-C, Swift code feels natural to read and write.
- **Interactive.** Use Swift interactively to experiment with your ideas and see instant results.
- **Unified.** Swift is a complete replacement for both the C and Objective-C languages. It provides full object-oriented features, and includes low-level language primitives such as types, flow control, and operators.

For full information about the Swift language and to get started using it, see *The Swift Programming Language* .

Xcode 6 Features for Swift

- **Playgrounds.** Playgrounds make writing Swift code productive and easy. Enter a line of code, and the result appears immediately. If your code runs over time—for instance through a loop—you can add that line of code to Timeline Assistant to watch it progress. Display variables in a graph, inspect each step of drawing a view, or watch an animated SpriteKit scene. When you've perfected your code in the playground, simply move that code into your project. Some uses for playgrounds include:

  Designing a new algorithm, watching its results every step of the way

  Experimenting with new API or trying out new Swift syntax

  Creating new tests and then verifying that they work before promoting them into your test suite

- **Learn in a playground.** Open select documentation in a playground to learn from the tutorial in an interactive environment. The combination of richly formatted documentation and an interactive playground makes it easy to fully explore the API, changing and experimenting with the sample code.
- **Read-eval-print loop (REPL) in LLDB.** The debugging console in Xcode includes an interactive version of the Swift language called the *read-eval-print loop (REPL)* built right in. Use Swift syntax to evaluate and interact with your running app, or write new code to see how it works in a script-like environment. REPL is available from within the Xcode console or by using LLDB from within Terminal when attached to a running process.
- **Per-language documentation.** The Xcode documentation viewer shows Quick Help or reference documentation in the language of your choice—Objective-C, Swift, or both.
- **Synthesized headers.** When you need to see how the API you are using was written, Xcode shows it to you in the language you expect. For API originally written in Objective-C, Xcode shows you a version of the original header file in Swift syntax, complete with the author's comments.

Additional Feature Enhancements in Xcode 6 IDE

Bringing more live visualization to your existing projects, the following enhancements are also Swift compatible.

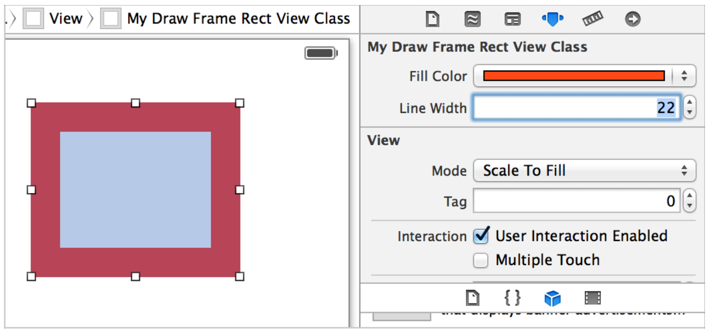
Testing

- **Performance measurement.** The enhanced XCTest framework now supports the ability to quantify the performance of each part of an app. Xcode runs your performance tests and allows you to define a baseline performance metric. Each subsequent test run compares performance, displays the change over time, and—by highlighting the problem area—alerts you to sudden regressions a code commit could introduce.
- **Asynchronous code testing.** XCTest now provides API for testing code that executes asynchronously. You can now create tests for network operations, file IO, and other system interactions that execute using asynchronous calls in a straightforward and simple manner.

For more information, see *Testing with Xcode*.

Interface Builder

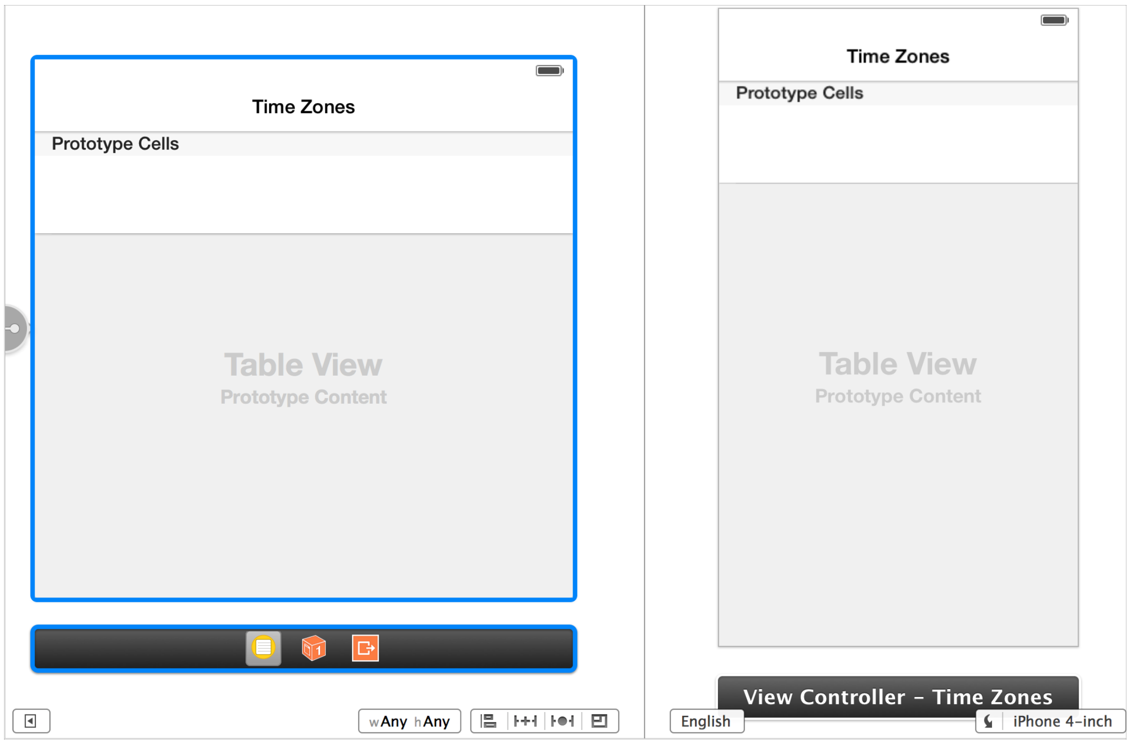
- **Live rendering.** Interface Builder displays your custom objects at design time exactly as they appear when your app is run. When you update the code for your custom view, the Interface Builder design canvas updates automatically with the new look you just entered in the source editor, with no need to build and run. You can use the Interface Builder inspector to see properties automatically as well. Take advantage of new APIs that allow you to customize the behavior of custom controls on the Interface Builder canvas—for instance, you can load sample data on the fly.



My Draw Frame Rect View Class
Fill Color
Line Width 22
View
Mode Scale To Fill
Tag 0
Interaction ✓ User Interaction Enabled
Multiple Touch

For more information on live rendering, see Creating a Custom View That Renders in Interface Builder.

- **Size classes.** Size classes for iOS 8 enable designing a single universal storyboard with customized layouts for both iPhone and iPad. With size classes you can define common views and constraints once, and then add variations for each supported form factor. iOS Simulator and asset catalogs fully support size classes as well.

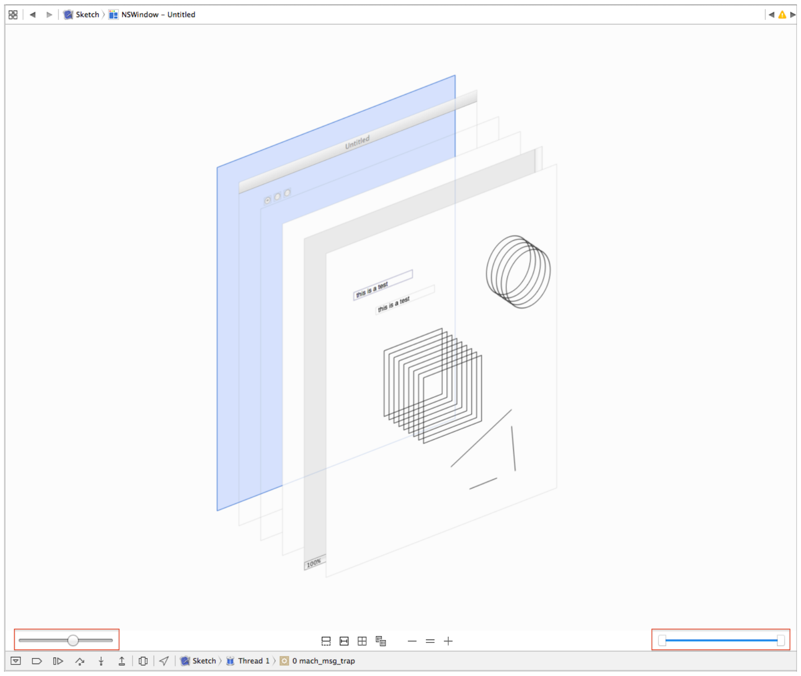


For more information on size classes, see *Size Classes Design Help* .

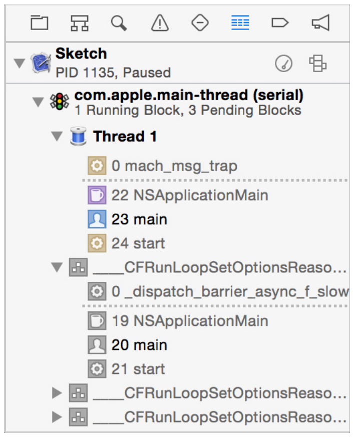
- **Custom iOS fonts.** Interface Builder renders embedded custom fonts during design time, giving a more accurate preview of how the finished app will look, with correct dimensions.

Debugger

- **View debugging.** Using the view debugger makes debugging an app's appearance as easy as debugging lines of code. A single button click pauses your running app and "explodes" the paused UI into a 3D rendering, separating each layer of a stack of views. Using the view debugger makes it immediately obvious why an image may be clipped and invisible, and the order of the graphical elements becomes clear. By selecting any view, you can inspect the details by jumping to the relevant code in the assistant editor source view. The view debugger also displays Auto Layout constraints, making it easy to see where conflicts cause problems.



- **Enhanced queue debugging.** The debug navigator records and displays recently executed blocks, as well as enqueued blocks. You can use it to see where your enqueued blocks are and to examine the details of what's been set up to execute.



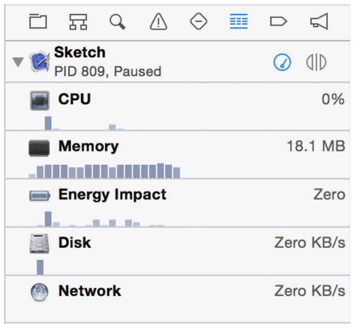
Sketch
PID 1135, Paused

com.apple.main-thread (serial)
1 Running Block, 3 Pending Blocks

Thread 1
0 mach_msg_trap
22 NSApplicationMain
23 main
24 start
____CFRunLoopSetOptionsReaso…
0 _dispatch_barrier_async_f_slow
19 NSApplicationMain
20 main
21 start
____CFRunLoopSetOptionsReaso…
____CFRunLoopSetOptionsReaso…

- **Debug gauges.** Debug gauges provide at-a-glance information about resource usage while debugging. They call your attention to previously unknown problems, especially related to areas that could create poor user experience or drain excess battery on portable Mac computers and devices.

I/O gauges. Two new debug gauges, Network Activity and File Activity, visually highlight spikes in input/output activity while your app is running.



iCloud gauge. The iCloud debug gauge includes support for the new Documents in the Cloud and CloudKit features that provide access to files outside the app-specific container.

SpriteKit and SceneKit

- **Graphics and game development.** Support for SpriteKit has been significantly enhanced with a new SpriteKit level designer and improved display of SpriteKit variables when debugging.
- **Support for iOS.** SpriteKit and SceneKit are now enhanced to work together and on iOS. Create scenes for your SpriteKit games from within Xcode. It is easier than ever to define how your characters, backgrounds, and the rest of your game comes together—it has never been easier to share code when creating great games for both iOS and OS X.

Extensions and Frameworks

- **Extensions support.** Add an extension target to any iOS or Mac app to expand your app's functionality to other apps in the OS. Xcode connects to the extension when launched, debugging the extension as it runs in the safe, embedded OS context.
- **Frameworks for iOS.** iOS developers can now create dynamic frameworks. Frameworks are a collection of code and resources to encapsulate functionality that is valuable across multiple projects. Frameworks work perfectly with extensions, sharing logic that can be used by both the main application and the bundled extensions.

For more information, see *App Extension Programming Guide* .

iOS Simulator

- **Configurations.** New iOS Simulator configurations allow you to keep data and configuration settings grouped together. Run one configuration for one version of an app, with its own data, and another configuration for a different app version.

For more information on iOS Simulator, see *iOS Simulator User Guide*.

Localization

- **XLIFF import-export.** Xcode can package your localizable strings into the industry standard XLIFF format to send off for localization. When localization is completed, it's easy to integrate the new languages back into the project.
- **Implicit .strings file.** Xcode automatically generates the base language `.strings` file directly from your source code—now you no longer need to manage this `.strings` file by hand.
- **Preview in Interface Builder.** While designing in Interface Builder, the preview assistant can show how the interface appears in other languages. You can see how your interface responds to longer or shorter languages.
- **Run in locale.** Xcode can run your app on iOS Simulator, or directly on devices, as it would appear to customers in other countries.

For more information on Xcode 6 localization, see *Internationalization and Localization Guide*.

Compiler

- **Profile Guided Optimization.** Profile Guided Optimization (PGO) works with the LLVM optimizer and your performance tests to profile the most actively used parts of your app. You can also exercise your app manually to generate a performance profile. PGO uses the profile to further optimize your app, targeting the areas that most need optimization, improving performance beyond what setting optimization options alone can achieve.

  For more information on Profile Guided Optimization, see *Xcode Profile Guided Optimization*.
- **User-defined modules.** Developers are now able to define modules for their own Objective-C code, making it easier than ever for them to share frameworks across all their projects. By combining user-defined modules with Swift's automatic creation of modules, the two languages work together seamlessly.

Instruments

- **New user interface.** The new Instruments user interface makes configuring your performance tuning session easier and improves control. The new template chooser allows you to choose your device and target as well as the starting point for your profiling session. The track view allows direct click-and-drag to set the time filter range. The toolbar takes up less space to let you focus on the task at hand.

  Instruments now looks and works more like Xcode. The tracks of recorded data are given more space, and configuration for how data is collected and viewed is managed in a unified inspector area.
- **Profile tests.** Choose any test or test suite to profile, great for analyzing memory leaks in a functional test or time profiling a performance test to see why it has regressed.
- **Support for simulator configurations.** Simulator configurations are treated like devices by Instruments, making it easy to launch or attach to processes in the simulator.
- **New Counters instrument.** Counters and Events instruments have been combined into a more powerful instrument and made easier to configure. It can track individual CPU events, and you can specify formulas to measure event aggregates, ratios, and more. iOS developers on 64-bit devices can now use Counters to fine-tune apps.
- **Swift and Extensions support.** Of course, Swift is supported—you'll see Swift symbols in stack traces and Swift types in Allocations. You can also use Instruments to profile your app extensions.

Xcode Server

- **Triggers.** Triggers allow you to make more complex integration scenarios by configuring server-side rules to launch custom scripts before or after the execution of an Xcode scheme.
- **Performance test integrations.** Xcode Server supports the new Xcode performance-testing features, making it easy for a team to share a Mac computer and a group of iOS devices for continual performance testing.
- **Delta tracking.** Issues are now tracked per integration, so you can see when an issue appeared or when it or was fixed, and by whom.
- **Greater control.** Configuration options in Xcode Server give development teams even greater control over the execution of bots. New settings for integration intervals, grouping of bots, and iOS Simulator configurations make Xcode bots more powerful than ever. The new reports UI includes bot-level statistics—for example, the number of successful integrations, commit and test addition tracking, and so forth.

For more information on Xcode Server, see *Xcode Server and Continuous Integration Guide* .

HomeKit Accessory Simulator

- **HomeKit capability.** The new HomeKit framework allows your app to communicate with and control connected accessories in a user’s home. To use HomeKit with your iOS apps, set the HomeKit switch in the project editor capabilities panel. Choose Xcode > Open Developer Tool > HomeKit Accessory Simulator to start using the simulator.

For more information on HomeKit, see *HomeKit Framework Reference* .

Xcode 6.0.1

Xcode 6.0.1 is a maintenance release responding to reported developer issues and Apple qualification testing.

Xcode 6.1

Xcode 6.1 adds development for OS X with Swift, and includes bug fixes as well as other new features.

SDK Support for OS X Yosemite

- Xcode 6.1 includes development support with SDKs for OS X version 10.9, OS X version 10.10, and iOS 8.

Swift

- Swift has access to all Cocoa frameworks for OS X development using the SDK for OS X version 10.10.
- Swift development targets can deploy on both OS X Mavericks and OS X Yosemite.

Interface Builder

- **Storyboards for OS X.** Storyboards come to OS X with Xcode 6, taking advantage of new view controller APIs in AppKit. Storyboards make it easy to wire together multiple views and define segue animations without writing code. Storyboards for OS X encourage interfaces that follow Mac standards so that your apps behave the way users expect.

  For more information on storyboards, see *Storyboard Help* .

Xcode 6.1.1

Xcode 6.1.1 includes bug fixes and performance improvements.

Xcode 6.2

Xcode 6.2 adds support for iOS 8.2. It includes the new WatchKit framework for developing Apple Watch apps.

Tools Support for WatchKit includes:

- Design tools for building Apple Watch interfaces, glances, and notifications
- Debugging and profiling support
- Apple Watch support in iOS Simulator for testing apps, glances, and notifications

Xcode 6.3

Xcode 6.3 adds support for iOS 8.3, updates to Swift, and many other new features.

Swift 1.2

- Swift 1.2 generates substantially faster performing code compared to Swift 1.1.
- Xcode now builds Swift targets incrementally for greater efficiency: a single file change no longer invokes building all of the source files in a target.

  These and other noteworthy improvements to Swift and Swift standard library are detailed in Xcode Release Notes.
- The new Swift migrator in Xcode 6.3 eases moving your Swift 1.1 code to Swift 1.2.

  In Xcode, select `Edit > Convert > To Latest Swift Syntax` to invoke the migrator.

Crashes Organizer for App Store and TestFlight Users

- A new feature that helps opted-in App Store users and TestFlight users collect and analyze crash log data for your apps.

  Crash reports gathered from opted-in App Store users and TestFlight users can be displayed in the Crashes Organizer. More details are available in Xcode Release Notes and Crashes Organizer Help in the Xcode documentation.

Xcode Playground Enhancements

- Improved documentation authoring with inline marked-up comments, inline playground results, the ability to view and edit resources embedded in playgrounds, and the ability to integrate auxiliary source files into Playgrounds. These features enable the creation of rich new experiences in playgrounds.

  For more information, see Xcode Release Notes, *Playground Reference* , and Playground Help in the Xcode documentation.

Force Touch Support

- Xcode uses Force Touch trackpad gestures for Macs that include it, and supports configuring Force Touch trackpad functionality on OS X in Interface Builder editor for `NSButton` and `NSSegmentedControl`.

> **Note:** Adopting Force Touch in Interface Builder requires running Xcode on OS X Yosemite version 10.10.3.

Objective-C Enhancements

- In Objective-C code, you can now directly express the nullability of pointers in header files, improving interoperability between Swift and Objective-C.

  See Xcode Release Notes for details on the changes and enhancements.

Debugger Enhancements

- LLDB has been enhanced to improve the support for modules in C-based languages as well as provide overall improvements in Swift debugging support.

  The LLDB Objective-C expression parser can now import modules, enabling subsequent expressions to rely on function and method prototypes defined in the module. Additional benefits of importing modules include better error messages, eliminating potentially incorrect inferred argument types, and more.

  See Xcode Release Notes for details on the enhancements.

Apple LLVM Compiler Version 6.1

- LLVM version 6.1 includes support for C++14, enhanced warning diagnostics, and new optimizations.

This updated compiler includes full support for the C++14 language standard, a wide range of enhanced warning diagnostics, and new optimizations. Support for the arm64 architecture has been significantly revised to better align with the ARM implementation; the most visible impact is that several vector intrinsics have changed to more closely match the ARM specifications.

ARM64 Intrinsics Changes

- The argument ordering for the arm64 vfma/vfms lane intrinsics has changed.

  By default, the compiler now warns about any use of the intrinsics but will retain the old behavior. To reduce risk, the transition to the new ordering is being completed in stages.

  Important: See the section *ARM64 Intrinsics Changes* in Xcode Release Notes for details on this transition.

Xcode 6.3.1

Xcode 6.3.1 is a maintenance update with bug fixes and performance improvements.

Xcode 6.3.2

Xcode 6.3.2 is a maintenance update with bug fixes and performance improvements.

- Swift projects now compile quickly, fixing a speed regression in Xcode 6.3.1.

Xcode 6.4

Xcode 6.4 includes support for development on iOS 8.4, along with bug fixes ad performance enhancements.

For additional details on the Xcode 6.4 release, see *Xcode Release Notes* .

New Features in Xcode 5.1

- Xcode 5.1 (page 26) includes SDKs for OS X v10.8, OS X v10.9, and iOS v7.1 and other enhancements.
- Xcode 5.1.1 (page 27) is a maintenance release addressing reported developer issues and Apple qualification testing.

Xcode 5.1

Xcode 5.1 requires a Mac running OS X 10.8.4 (or later), or OS X 10.9. It includes the following highlighted features:

- SDKs for OS X v10.8, OS X v10.9, and iOS v7.1
- Updates to the LLVM compiler

  Important: Xcode 5.1 no longer supports building OS X apps that use garbage collection; attempting to build such targets results in a build error. You should employ the Xcode migration tool to convert to ARC (Automatic Reference Counting) on all projects using garbage collection. For more information about transitioning to ARC, see *Transitioning to ARC Release Notes*.

- Enhancements to Auto Layout constraints and editing tools in Interface Builder

  Interface Builder supports the creation of proportional and aspect ratio constraints. Enhancements to the Attributes inspector enable the creation of cross-attribute constraints as, for example, to align the leading edge of one object with the center horizontal position of another. See Adding Layout Constraints to Objects by Control-Dragging and Editing Auto Layout Constraints.

- Variables Quick Look in the Xcode debugger supporting custom object types

  Developers can now provide Quick Look content for their classes. When an instance of a class is viewed with Quick Look using the variables view or a data tip, the debugger looks for a method named `debugQuickLookObject` in the class implementation. For more details, see *Quick Look for Custom Types in the Xcode Debugger*.

Xcode 5.1.1

- The Xcode 5.1.1 release is a maintenance release addressing reported developer issues and Apple qualification testing.

For additional details on Xcode releases, see *Xcode Release Notes* .

New Features in Xcode 5

Objective-CSwift

- Xcode 5.0 (page 28) is a major Xcode update which provides many new features along with support for development with iOS 7.0 SDK.
- Xcode 5.0.1 (page 35) adds support for development on OS X v10.9 and other feature additions.
- Xcode 5.0.2 (page 36) addresses reported developer issues and Apple qualification testing.

Xcode 5.0

Xcode 5 is the latest release of the Apple developer tools. Building on the design of Xcode 4, this release focuses on features and enhancements to improve your ability to adopt core platform features, design new interfaces, and deliver high-quality applications.

Xcode 5 adds support for developing with iOS 7.0 SDK. Xcode 5 requires a Mac running OS X v10.8.4.

User Experience Improvements

- The Xcode 5 user experience has a cleaner UI with more working space for your content. The changes are many and subtle—for instance, the toolbar has been shortened and simplified to produce more space in the editors. The new look is simpler with fewer distracting details, and the highlighting of buttons and panels is clearer and easier to see at a glance. At the same time, your familiarity with Xcode 4 works for you—you can be familiar with the new Xcode 5 UI in just a few minutes of use. You'll find all the familiar controls there for you, clearer to see and use, putting your content first and foremost.
- Open Quickly has been revamped with a streamlined input panel that is easier to use. The changes under the hood include a much stronger matching algorithm that returns highly prioritized results faster, and the results are presented with more content.
- The refined search navigator allows all current search options and settings to be seen at a glance. The options are easily manipulated by clicking directly on them in the search navigator. You can set search scopes, including selecting multiple folders in a project, and save them by name for easy re-use. The search results display wraps to allow you to see more results easily and quickly.

Automatic Configuration

- The new Accounts preferences pane allows you to manage your Apple IDs, repositories, and continuous integration servers from one place in Xcode preferences. Add and view your Developer Program Apple IDs, add source code repositories to store the location and authentication information used when accessing Subversion and Git, and add continuous integration servers to take advantage of Xcode Services on OS X Server.
- The streamlined Capabilities settings in the project editor allow you to easily configure platform features such as iCloud, Game Center, and more.
- You choose the signing identity from the target editor.
- Xcode 5 uses the Accounts preferences, the Capabilities settings, and the signing identity settings to automatically create provisioning profiles with proper settings for you. It can also identify and offer to fix issues in provisioning profiles as well.

  To learn more about the tasks and workflow required to develop and distribute OS X or iOS apps with Xcode 5, see *App Distribution Guide* .

Testing

- Xcode 5 provides a new test navigator that offers an overview of all tests in the workspace. The new test navigator has the ability to easily add new test targets and test classes, as well as the ability to run individual tests or ad hoc collections of tests. It can also show the status of the last test run for each test.
- New test categories in the assistant editor enable you to edit code and tests side by side. The source editor provides the status of the tests, and you can run individual tests from within the editor. The assistant editor's "Test Callers" and "Test Classes" categories provide access to unit tests related to the current source code in the primary editor.
- The new XCTest testing framework provides support for iOS and OS X projects. It is the default for new projects and works for iOS 7 and later, as well as all versions of OS X.

  Note: Xcode 5 offers the ability to migrate your tests from the OCUnit framework to the XCTest framework. Use the "Convert to XCTest" command located in the Refactor submenu of the Edit menu.

- The `xcodebuild` command-line tool now supports the `test` action for both iOS and OS X tests, allowing a scheme's test action to be performed from the command line or integrated into other scripts. Detailed information on using `xcodebuild` for running tests can be obtained using `man` from a Terminal window. Type: `man xcodebuild`

Continuous Integration

- Xcode 5 supports using services offered by the Xcode service included with OS X Server. You create a bot in Xcode to build, analyze, test, and archive your project on an OS X Server shared by your development team.
- Bots can be configured to launch an integration on every commit to your SVN or Git repository, or at defined intervals.
- Continuous integration allows you to see immediately when anyone on the team breaks a build or starts failing tests.
- You can view bot integration results in Xcode 5, drilling into build and test failures to find and fix the problem.

Note: Xcode bots require Xcode services, a new feature of the OS X Server product, that provide:

- Hosted bots that perform continuous integration for any Xcode project
- Archives of past integration logs available to everyone on your team
- Creation and hosting of Git repositories
- A web interface for QA teams to access recent builds and archives
- A web-based "scoreboard" to display an overview of recent bot results (useful, for example, on big-screen HDTV)

For more information about continuous integration, see *Xcode Server and Continuous Integration Guide* .

Debugger

- Debug gauges have been added to the debug navigator to show real-time memory, CPU activity, energy use, iCloud, and OpenGL data with very low overhead. This improves the visibility of program data and provides key indicators for application performance debugging.
- Debug gauges serve as a gateway to Instruments. Open Instruments templates direct from the debug gauge detail display to investigate a variety of memory, performance, and energy use situations.
- The variables view and data tips display has been refined to show a summarized value for the variable, and presents the same hierarchical display as the variables view in the Xcode debugger area for looking at child values. The data tips support presenting variable info, and can print the Objective-C `description` of the object to the console.

  Clicking on the info button (❶) next to a variable brings up a display showing the console output.

- Clicking on the Quick Look button (👁) next to a variable presents a graphical display of the variable's contents, for known graphical types.
- The debugger automatically creates a new debug session for any embedded XPC services in an application.

  Note: XPC debugging is an OS X only feature.

- The debugger now provides control options to make `NSView` objects more visible when debugging. The options include turning on frame rectangles, alignment rectangles, flashing drawing done by `NSView`, and others.

  Note: NSView debugging options are only available for OS X 10.9 and later.

- The Debug menu includes an iCloud submenu with two new commands designed to help facilitate iCloud development.

  "Trigger Sync in Simulator" provides a convenient way to force an iCloud sync from an iOS app without having to switch to the simulator. See the iOS Simulator (page 34) section for details on the Trigger iCloud Sync feature.

  "Delete Container Contents" enables you to delete all documents and data in an app's iCloud container.

  Note: Deleting contents from the app's iCloud container will affect all of your iCloud enabled devices and cannot be undone.

OpenGL ES Support

- Apps using the OpenGL ES 3.0 API can now be debugged with the OpenGL ES frame debugger.

  Note: Requires devices supporting OpenGL ES 3.0 or later, see *iOS Device Compatibility Reference* for details.

- The OpenGL ES shader profiler enables you to profile OpenGL ES shaders on compatible iOS devices.

  When inspecting a captured OpenGL ES frame, set the debug navigator to "View Frame By Program" mode. In this mode, you see timings for all programs, their constituent shaders, and the draw calls using those shaders. Select a shader to see further detail on where time is spent within the shader.

Note: Requires devices supporting OpenGL ES 3.0 or later, see *iOS Device Compatibility Reference* for details.

Shader debugging requires an ES3 capable device but not an ES3 context. That is, you can use shader debugging with an ES2 app on OpenGL ES 3.0 compatible hardware.

- The new auto variables view mode automatically shows the relevant OpenGL ES state and bound objects for the current OpenGL ES command.
- Issues found in your OpenGL ES frame capture can now also be seen in the context of the frame. Issue badges appear in the debug navigator marking commands that triggered an issue. The auto variables view in the debug area lists the issues found at the current command.
- New OpenGL ES error breakpoints add support for breaking in the debugger in response to OpenGL ES errors including multi-threading issues, shader compilation failures, and program link failures.

Interface Builder

- Interface Builder in Xcode 5 includes support for the iOS 7 user experience and user interface objects.
- The Auto Layout editor provides more flexibility when designing app interfaces. The enhanced workflow for designing interfaces with auto layout puts you in greater control of setting object relationships.

  See *Interface Builder Help* for more information on using new Interface Builder features.
- The new Preview mode of the Assistant editor can show how the iOS 7 UI you are designing would look in portrait or landscape mode, or even how it would look when viewed on a device running iOS 6.
- The asset catalog manages images and icons in multiple resolutions. An asset catalog is a new asset management file type and editor in Xcode 5. You use asset catalogs to store and manage images for different platforms, devices, and scale factors. The catalog presents the image variants required, and provides you with the ability to define slice and stretch points for images that are resized at runtime. For more information on using asset catalogs, see *Asset Catalog Help* .

Source Control and Version Editor

- The source control management workflow in Xcode 5 creates a project-centric experience by removing the Repositories organizer and moving these functions into the project window and the Source Control menu. The Source Control menu provides convenient access to many workflows including Check Out, Commit and Push changes, Update, Add, and History.
- The Xcode 5 source control management features also include the ability to check out multiple working copies and handle branch management directly. You manage all repository location and authentication information in one place using Accounts preferences.

For more information about using the new source control management workflow, see *Source Control Management Help* .

- Subversion has been upgraded to version 1.7.9.

Note: The Subversion 1.7 upgrade includes a migration option. When opening an existing project in Xcode 5, you are asked if you would like to upgrade. If you are still working with an older version of Xcode, do not update the SVN version—once you do, you cannot go back. If you choose not to update to 1.7, you can do so later at any time using the History command in the Source Control menu.

The Subversion 1.7 upgrade affects only working copies of your source and not your SVN servers.

Compiler

- The new Auto Vectorizer supports automatic optimization of computational loops for both iOS and OS X apps. To enable this option, use the `Vectorize Loops` option in the target build settings.
- Modules for system frameworks speed build time and provide an alternate means to import APIs from the SDK instead of using the C preprocessor. Modules provide many of the build-time improvements of precompiled headers with less maintenance or need for optimization. They are designed for easy adoption with little or no source changes. Beyond build-time improvements, modules provide a cleaner API model that enables many great features in the tools, such as Auto Linking.

  Note: All new projects created in Xcode 5 now build with modules enabled by default. For existing projects, you enable modules by using the project Build Settings panel. Search for "module" and set `Enable Modules (C and Objective-C)` to YES.

- Auto Linking is enabled for frameworks imported by code modules. When a source file includes a header from a framework that supports modules, the compiler generates extra information in the object file to automatically link in that framework. The result is that, in most cases, you will not need to specify a separate list of the frameworks to link with your target when you use a framework API that supports modules.
- The default C++ standard library for projects deploying to iOS 7 is now the LLVM libc++ library, which utilizes many of the benefits of C++11. Applications built using this library can deploy back to iOS 5 and OS X 10.7.
- LLVM now supports the AVX2 vector instruction extensions available in new Macs. To enable these extensions, use the Xcode build setting `Enable Additional Vector Extensions`.
- A new optimization level `-Ofast`, available in LLVM, enables aggressive optimizations. `-Ofast` relaxes some conservative restrictions, mostly for floating-point operations, that are safe for most code. It can yield significant high-performance wins from the compiler.

Note: LLVM-GCC is not included in Xcode 5.

iOS Simulator

- iOS Simulator now supports iCloud syncing of documents and KVS data within an app, enabling apps to sync between devices using iCloud. This feature is useful when testing to ensure that the app documents and data are syncing properly across multiple devices.

  Note: With the app running in the iOS Simulator, sign in to an Apple ID account using the Settings app. After signing in, use the "Trigger iCloud sync" command in the Debug menu to tell the simulator to sync with other devices.

- Chinese Sina Weibo and Tencent Weibo character systems have been added to the iOS Simulator.

  Note: With your app running in the iOS Simulator, use the Settings app to add a Chinese keyboard. Then relaunch the Settings app to see the new settings for Sina and Tencent Weibo.

See *iOS Simulator User Guide* for more information on using the new iOS Simulator features.

Instruments

- The Zombies instrument template has been enhanced in Xcode 5 and now supports use on devices. Using Zombies on devices requires iOS 7.
- The Allocations instrument now includes virtual memory mappings.
- Retain-release pairing in the Allocations instrument has been enhanced to help track down imbalanced retain counts.

Documentation

- The documentation experience for Xcode 5 has been redone. A separate window tailored to search and display provides fast access to documentation resources. The documentation window supports tabs so that you can have multiple documentation references simultaneously available.
- A dedicated table of contents display area is incorporated into the documentation window, allowing you to easily see and browse topics in open documents.
- The new documentation experience includes support for bookmarks and integrated, easy sharing via Mail, Messages, and other tools.

- Project documentation from framework API reference documentation and structured comments in your own source code are displayed in the quick help panel and in code completion popover views. Doxygen and HeaderDoc structured comments are supported formats.

Sprite Kit Support

- The Xcode 5 build system incorporates support for including Sprite Kit texture atlases as part of your project's build cycle. A texture atlas provides you with a way to improve the performance of Sprite Kit–enabled apps by combining all of an app's image assets into one or more large images. You can improve the performance of your app by drawing multiple images with a single draw call. More information about texture atlases in available in *Texture Atlas Help* and *SpriteKit Programming Guide* .
- Xcode 5 includes a new editor for Sprite Kit particle emitters. Particle emitters are a function of the Sprite Kit framework that allow you to specify a specific point in your display and create images that move and change over time. Using emitters, you can simulate rain, snow, spinning car wheels, fire, and many other effects in your game. More information on particle emitters can be found in the *Particle Emitter Editor Guide* , *SpriteKit Framework Reference* , and the *SpriteKit Programming Guide* .

Deprecation and Removal Notice

Important: Xcode 5 does not support use of the LLVM-GCC compiler and the GDB debugger. Existing projects configured to use LLVM-GCC and GDB will be reconfigured to use the LLVM compiler and LLDB debugger when opened by Xcode 5. Please file a bug using bugreporter.apple.com for any issues that prevent you from moving to Xcode 5 for this reason.

Xcode 5.0.1

Xcode 5.0.1 adds support for development on OS X v10.9 and other feature additions.

- Development with Xcode 5.0.1 is hosted on OS X 10.8.4 or later, and OS X 10.9.
- SDKs for OS X 10.8 and OS X 10.9, and iOS 7.0.3 SDK, are included.
- Xcode 5.0.1 supports continuous integration bots, hosted on OS X Server.

  Use the Add button (+) button to add OS X Servers in the Accounts preferences panel, then click the menu command Product > Create Bot.
- Includes support for OS X Server hosted repositories.
- iOS 6 (32-bit) and iOS 7 (32-bit and 64-bit) binaries build with a single build target.

Xcode 5.0.2

- The Xcode 5.0.2 release is a maintenance release responding to reported developer issues and Apple qualification testing. See *Xcode Release Notes* for more detailed information.

New Features in Xcode 4.6

- Xcode 4.6 (page 37) adds support for development with the iOS 6.1 SDK and includes new features for the compiler and debugger, along with general improvements to performance and robustness.
- Xcode 4.6.1 (page 38) supports development with OS X 10.8.3 SDK and provides compatibility for ARC in projects targeting OS X 10.6.
- Xcode 4.6.2 (page 38) is a maintenance release responding to developer reported issues and Apple QA testing input.
- Xcode 4.6.3 (page 38) fixes an issue where debugging in the iOS Simulator could hang on OS X 10.8.4.

Xcode 4.6

For the compiler:

- Compilation warnings are added which assist in finding bugs when using ARC and weak references.
- `otool` is enhanced to support disassembly of Intel AVX instructions.
- The LLVM compiler now supports C++11 “user defined literals” and “unrestricted unions” features.
- The static analyzer has enhanced cross-function analysis for C++ and Objective-C methods. With this enhancement, the static analyzer can now find deeper bugs that cross method calls in Objective-C and C++ code. This new capability extends the cross-function analysis for C functions that was introduced in Xcode 4.5.

For the debugger:

- LLDB has been enhanced to read metadata from the Objective-C runtime. This enhancement greatly reduces the need to cast arguments and the results of message sends, and makes properties more often usable (particularly from system classes).
- LLDB has improved support for stepping over inlined functions. This improved support is particularly useful for `libc++` and `always_inline` functions like `NSMakeRange`.
- LLDB now prints function argument information in backtraces by default.
- LLDB now supports “thread return,” temporary breakpoints, and a variety of aliases to add common shortcuts from GDB.

- The elements of `NSArray` and `NSDictionary` objects can now be inspected in the Xcode debugger.

Important: The LLVM GCC compiler does not include the latest Objective-C and C++11 features. Xcode 4.6 is the last major Xcode release which includes the LLVM GCC compiler and the GDB debugger. Please migrate your projects to use the LLVM compiler and LLDB debugger, and file a bug in bugreporter.apple.com for any issues that require you to use LLVM GCC or GDB.

Xcode 4.6.1

Xcode 4.6.1 is a maintenance release responding to reported developer issues and Apple qualification testing. Major highlights of this release include:

- Xcode 4.6.1 provides an update to the included OS X SDK, supporting new OS X v10.8.3 APIs.
- ARC compatibility has been ensured for projects targeting OS X 10.6

Xcode 4.6.2

- The Xcode 4.6.2 release is a maintenance release responding to reported developer issues and Apple qualification testing.

Xcode 4.6.3

- The Xcode 4.6.3 release fixes an issue where debugging in the iOS Simulator could hang on OS X 10.8.4.

Additional improvements to Xcode robustness and reliability have been incorporated. See *Xcode Release Notes* for more information.

New Features in Xcode 4.5

Xcode 4.5 adds support for development with iOS 6 SDK and includes these additional features:

- LLDB is now the default debugger.
- LLDB supports hardware watchpoints on iOS devices.
- OpenGL debugging and performance analysis for iOS apps is integrated into Xcode.
- Auto Layout is supported for iOS 6.

Xcode 4.5 also extends new features released in Xcode 4.4 as listed below:

- Improved localization workflow using base language `.xib` files now supports OS X 10.8 in addition to iOS Storyboards.
- Objective-C literals syntax in `NSNumber`, `NSArray`, and `NSDictionary` classes are supported for iOS.
- Support for subscripting using `'[ ]'` syntax with `NSDictionary` and `NSArray` are supported and deploy back to iOS 5.
- Compatibility with the C++11 standard is improved with added support for lambda expressions.

For full details of Objective-C language feature availability, tools, and deployment capability, see *Objective-C Feature Availability Index* .

Note: Check the Documentation tab of the Downloads preferences pane to be sure that Xcode always has the latest documentation.

New Features in Xcode 4.4

Xcode 4.4 adds features to support OS X v10.8 and iOS 5.1 as well as other enhancements to the toolset.

Note: Check the Documentation tab of the Downloads preferences pane to be sure that Xcode always has the latest documentation.

LLVM 4.0 Compiler

Xcode includes an updated LLVM Compiler version 4.0 with the following enhancements.

Objective-C Language Features

- Literals syntax is supported for `NSArray`, `NSDictionary`, and `NSNumber` objects, using the same '`@`' operator as for `NSString` literals.
- Subscripting is enabled for Objective-C containers, including `NSDictionary` and `NSArray`. Use the '`[ ]`' syntax convention.
- Objective-C `@properties` are synthesized by default when not explicitly implemented.

Xcode supports backward deployment of code that uses the literal syntax and object subscripting to OS X v10.7 and later, you must use the OS X v10.8 SDK to make these features available. The default `@synthesize` feature requires no special SDK or runtime support.

Note: iOS development using literal syntax and object subscripting is not yet implemented.

Improved Support for the C++11 Standard

- Lambda expressions are allowed and permit interoperability with blocks-based APIs in Objective-C++.
- Generalized initializer lists are supported.
- Generalized constant expressions (`constexpr`) are supported.

Improvements to the Static Analyzer

Xcode's built-in source code analysis tool, launched with the Analyze command in the Product menu, is enhanced for common security mistakes in API and malloc usages.

- The static analyzer engine can find complicated bugs that span function boundaries using interprocedural analysis.
- More exhaustive memory checks are made to `malloc`-related memory management and the detection of insecure API uses.

New Interface Builder Support for AppKit Features

Xcode's Interface Builder includes support for new AppKit features.

- Autolocalization
- Improved trackpad API
- Auto Layout improvements
- CoreUI-based UI customization
- `NSView` API
- Paging control
- The addition of Page Control View

Note: The Interface Builder features listed above require OS X v10.8 support.

Scene Kit Editor

Xcode 4.4 introduces a viewer and editor for 3D scene files, included in a project as DAE documents, to support use of the Scene Kit API. The Scene Kit editor allows you to preview and fine-tune the 3D scenes, and play embedded animations. You can also inspect 3D scenes for information to use in your source code. The Scene Kit editor is invoked by selecting a DAE file in the project navigator.

Note: The Scene Kit editor requires OS X v10.8 or later.

Code Completion Enhancements

Code completion now has an integrated form of QuickHelp with a short description of each item in the list based on the documentation or the specific code snippet. Integrated within the code completion window, it is displayed in a section either above or below the code completion list.

Xcode can offer symbols during code completion that haven't yet been included or imported in the current file (the framework was added to the project, but the `#import` was not included in the file). When possible, Xcode will use umbrella headers for auto import completions and will denote not-yet-linked symbols with a #error indicating which binary needs to be linked. It is a known limitation that auto import completions are available only for symbols that are already visible in at least one file in the current workspace. If necessary, this can be turned off in the Text Editing preferences.

Find and Search Additions

Find and Search have been enhanced with three new capabilities:

- The Find Bar and Search Navigator have added support for find patterns as a simpler alternative to regular expressions. In the search field, click the magnifying glass icon and select Insert Pattern.
- The Find Navigator now supports searching for references to indexed symbols. Choose Symbolic from the Find Navigator's Style pop-up menu. This form of search performs significantly faster than textual searches and excludes results from comments and nonsource files.
- Xcode can show the callers and callees of the current function or method. This function is accessed from the Show Related Items menu, or by using the Assistant editor and selecting Callers or Callees in the jump bar pop-up menu.

Source Editor Jump Bar Enhanced

The source editor jump bar pop-up menu now lists TODO and `#pragma mark` comments that are inside methods and functions.

Gesture Support Additions for Track Pad Use

Pinch-to-zoom and two-finger-double-tap change the zoom level in the following editors and viewers:

- User Interface editor
- Hex editor
- Core Data Model editor
- Quicklook editor
- OpenGL ES viewers
- Instruments track view

Three-finger-single-tap invokes QuickHelp.

Two-finger-swipe moves back and forth in the Xcode history.

Notification Enhancement

Xcode 4.4 allows the OS X v10.8 systemwide notification system to display build and warning notifications.

New Features in Xcode 4.3

- Xcode 4.3 (page 44) adds features to support iOS 5.0 and OS X v10.7, as well as other enhancements to the toolset. Minor revision updates:
- Xcode 4.3.1 and 4.3.2 (page 46) add support for the updated iOS SDK 5.1.
- Xcode 4.3.3 (page 46) provides an update to the included OS X SDK.

Xcode 4.3

The Xcode 4.3 Toolset Is Repackaged as a Single App

This version of Xcode is distributed as a single application bundle, `Xcode.app`, installed through the Mac App Store directly to the Applications folder. Installing Xcode 4.3 no longer requires the Install Xcode application. These changes make it easier for you to install and update Xcode using the standard Mac App Store mechanism.

Complementary Tools Launch from Within Xcode

The Xcode 4.3 installation reorganizes other key development tools and allows launching them using the Xcode > Open Developer Tool menu. For your convenience, you can also add these tools to the Dock to allow direct access.

Command-Line Tools Are Optional

The command-line tools are not bundled with Xcode 4.3 by default. Instead, they can be installed optionally using the Components tab of the Downloads preferences panel.

/Developer No Longer Exists

The simplification afforded by repackaging Xcode 4.3 as a single app bundle eliminates the need for the `/Developer` directory containing prior versions of Xcode. As a result, the Install Xcode application and the `uninstall-devtools` command line script are also no longer needed.

Note: The first time you run Xcode 4.3, you are prompted with a dialog to delete the Install Xcode application from the Applications directory if one is resident, and to remove any older installation of Xcode. These actions are optional, Xcode 4.3 can co-exist with older installations of Xcode on the same Mac. Some operations may require an administrator password to enable them if you switch from one version of Xcode to another.

If you choose not to delete these items on the first launch of Xcode 4.3, you can remove the `/Developer` folder and the Install Xcode application by dragging them to the Trash at any time.

Since the `/Developer` directory no longer exists with Xcode 4.3, some other parts of prior Xcode installations have moved.

Sample Code. The three sample projects previously available in `/Developer/Examples`—TextEdit, Sketch and CircleView—have been moved to the Sample Code sections of the iOS and OS X Developer Libraries at developer.apple.com.

Plug-ins, templates and other sub-components. Any path for component additions to developer tools that was previously found in a subdirectory of `/Developer` is now going to be located internal to the Xcode 4.3 application bundle. For example, Instruments templates—files such as `Automation.tracetemplate`—previously located in the `/Developer` subdirectory at `/Developer/Platforms/iPhoneOS.platform/`… is now stored in a similar path inside `/Applications/Xcode.app/Contents/Developer/`….

Other standalone utility applications and add-on technologies. Several additional tools are no longer part of the default Xcode installation, they are now downloadable as separate packages. The More Developer Tools menu command provides a direct jump to developer.apple.com/downloads in Safari where these development tools can be found.

The available downloads include:

- Audio tools: AULab, HALLab, and audio utility source code
- Accessibility tools: Accessibility Inspector, Accessibility Verifier
- Hardware IO tools: Bluetooth tools, IORegistryExplorer, USB Prober
- Graphics tools: CI Filter Browser Widget, OpenGL tools, Pixie, Quartz Debug, Quartz Composer tools
- Auxiliary tools: Clipboard Viewer, CrashReporterPrefs, Help Indexer, PackageMaker, Speech tools, SleepX
- Dashcode: Dashcode application

Auto Layout Is Now the Default for All New Cocoa Projects

New Cocoa projects created in Xcode 4.3 now use Interface Builder's Auto Layout feature by default. It can be disabled by deselecting an option in the Interface Builder design canvas:

Xcode 4.3.1 and 4.3.2

Xcode 4.3.1 and 4.3.2 add support for iOS 5.1 and include the new features of Xcode 4.3.

Note: The Xcode 4.3.2 release is a maintenance release responding to reported developer issues and Apple qualification testing.

Xcode 4.3.3

Xcode 4.3.3 provides an update to the included OS X SDK, supporting new OS X v10.7.4 APIs. Otherwise, Xcode 4.3.3 offers no new features.

Additional improvements to Xcode robustness and reliability have been incorporated. See *Xcode Release Notes* for more detailed update information.

New Features in Xcode 4.2

Xcode 4.2 adds features to support iOS 5 as well as other enhancements to the application.

Automatic Reference Counting

Xcode 4.2 includes a menu item to convert targets to use Automatic Reference Counting (ARC), which automates memory management for Objective-C objects. ARC makes memory management much easier, greatly reducing the chance that your program will have memory leaks. First, Xcode reviews your project to determine whether there are items that cannot be converted (and that you must therefore change manually). Then, Xcode rewrites your source code to use ARC.

To initiate the process, enable Continue building after errors in the General Preferences pane, then choose Edit > Refactor > Convert to Objective-C ARC. The targets that you convert are updated to build using the Apple LLVM compiler. Xcode attempts to build your target and to determine what changes must be made to use ARC. If it finds any issues that prevent conversion, Xcode displays a dialog directing you to review the errors in the Issue navigator. After you correct the errors, choose the Convert to Objective-C Automatic Reference Counting menu item again to restart the ARC-conversion workflow.

When Xcode successfully builds your application, it takes a snapshot of the current code so that you can revert later if you want to. Then Xcode displays a preview dialog showing the changes it's going to make. When you accept the changes, Xcode converts your code to use ARC.

For more information on ARC, see *Transitioning to ARC Release Notes* .

Default Compiler

The default compiler for iOS development in Xcode 4.2 is LLVM 3.0. Compared with the GCC compiler that was the default in Xcode 4.0 and the LLVM-GCC compiler in Xcode 4.1, LLVM provides better code generation and optimization than GCC, along with newer language support than LLVM-GCC, including support for ARC in Objective-C and for the new C++ standard, C++0x.

Storyboards

In Xcode 4.2, the Interface Builder user interface for iOS app UI design is based on using storyboards, that is, images of view controllers populated with user interface objects and connected together with segues. Storyboards enable you to use Interface Builder to specify all the screens in your application, including the transitions between them and the controls used to trigger the transitions. With storyboards, you can lay out every possible path through your application graphically, greatly reducing the amount of code you need to write for a complex multiscreen application.

To create a project that uses view controllers, choose File > New > New Project and select the Use Storyboard checkbox in the options dialog.

You start with a view controller object that represents your first scene (the *initial view controller*). To get view controllers for your storyboard, select Objects and Controllers from the Object library and drag the view controllers you want onto the canvas. Each view controller manages a single scene. On the iPhone, each scene represents the contents of a single screen. For iPad applications, a screen can be composed of the contents of more than one scene.

To storyboard your application, you link each object that's in a view controller and that can cause a change in the display, to another view controller that configures and implements the new scene. As you can see in the illustration below, the initial view controller has a green outline. You link the various view controllers in Interface Builder by Control-dragging between controls and view controllers. You can drag from any control that has an output to the header of any other view controller. You can add controls and views to each view controller's view just as you would add objects to a window or a view in the nib file of an Xcode 3 or Xcode 4.0 application.

The arrows between view controllers represent the segues from one scene to another. To configure a segue—for example, to specify the kind of transition to use between scenes—click the arrow and open the Attributes inspector. To define a custom transition, select Custom for the style of the segue and fill in the name of your custom segue class. Standard segue classes are in UIKit (see *UIKit Framework Reference*). For information about implementing the methods in the `UIViewController` class, see *UIViewController Class Reference*.

The result is a storyboard that graphically represents every screen of your application and the flow of control among the screens. Double-click the canvas to zoom out to see the entire storyboard.

OpenGL ES Frame Capture

The debugging experience has been updated to include a new workflow for debugging iOS OpenGL ES applications. When frame capture is enabled in the scheme for the application, the debug bar provides a new control for entering the OpenGL ES frame debugging view.

To enable this feature, you must run the application on a device and the device must be running iOS 5.0 or later. Set the destination in the scheme menu to an iOS device and choose Edit Scheme from the scheme selector in the toolbar. Select the Run action, click the Options tab, and select the OpenGL ES Enable Frame Capture checkbox.

When you build and run your OpenGL ES application, the debug bar includes a frame capture button. Click that button to capture a frame.

You can use Xcode 4.2 to:

- Inspect OpenGL ES state information.
- Introspect OpenGL ES objects such as view textures and shaders.
- Step through draw calls and watch the changes with each call.
- Step through the state calls that precede each draw call to see exactly how the image is constructed.

The debug navigator has a list of every draw call and state call associated with that frame. The buffers associated with the frame are shown in the editor pane, and the state information is shown in the debug pane.

You can step through draw calls in the debug navigator, or by using the double arrows and slider in the debug bar.

When you use the draw call arrows or slider, you can have Xcode select the stepped-to draw call in the debug navigator. To do so, Control-click below the captured frame and choose Reveal in Debug Navigator from the shortcut menu.

You can also use the shortcut menu to toggle between a standard view of the drawing and a wireframe view. The wireframe view highlights the element being drawn by the selected draw call.

Open the Assistant editor to see the objects associated with the captured frame. In the Assistant editor you can choose to see all the objects, only bound objects, or the stack. Open a second Assistant editor pane to see both the objects and the stack for the frame at the same time.

Double-click an object in the Assistant editor to see details about that object. For example, if you double-click a texture object, you can see the texture in detail.

Location Simulation

In Xcode 4.0 and 4.1, you could simulate only the current location in your application. As of Xcode 4.2, you can simulate locations other than your current location in iOS applications that use Core Location. To set a location, choose Edit Scheme from the scheme selector in the toolbar, select the Run action, and click the Options tab. You can then choose a location from the Location menu.

In addition, if you are running an application for iOS 5.0 or later that uses Core Location, the debug bar has the same location drop-down menu.

Downloading Components

To improve download time and installation efficiency with Xcode 4.2, the standard Xcode installer excludes some large tool components, such as older simulators, that are not essential to the current development toolset. Xcode presents you with a dialog when the simulator needs to be downloaded. Documentation preferences has consequently been replaced with Downloads preferences, including both documentation and components (simulators and SDKs). Using the Components tab of the Downloads preferences pane, you can view a description of each available component, download and install it.

New Features in Xcode 4.1

Xcode 4.1 adds features to support OS X v10.7 Lion as well as other enhancements to the application.

This article describes the new features in Xcode 4.1. For details on how to use these features, see *Xcode Overview* .

Project Modernization

When you open a project, Xcode 4 evaluates it to see whether there are any settings that should be updated. This feature provides an easy way to make sure your projects conform to the latest SDKs and best practices.

Open the Issue navigator to see whether anything in your project needs to be updated. You can also select the project in the project navigator and choose Editor > Validate Settings.

If the Issue navigator lists modernization issues, click the issue to see a dialog that explains the updates that should be made and lets you perform any or all of them.

After you have clicked Perform Changes, whether you choose to make all the changes or not, Xcode does not show the warning again. To rerun the check, select your project in the Project navigator and choose Validate Settings from the Editor menu.

Default Compiler

The default compiler for iOS development in Xcode 4.1 is LLVM-GCC 4.2. Compared with the GCC compiler that was the default in Xcode 4.0, LLVM-GCC provides better code generation and optimization than GCC, while being exactly source compatible with GCC 4.2.

Custom Behaviors

The Behaviors preferences pane lets you specify what should happen when a variety of events occur. Xcode 4.1 introduces new Behavior options, such as running a script, collapsing and expanding the navigator and utilities panes and the toolbar, or switching to Full Screen mode. This feature greatly expands the power of Xcode behaviors, enabling Xcode to run a script or perform a variety of actions in response to a large number of triggers.

You can also can design custom behaviors by defining behaviors that are triggered by menu items or their key equivalents. This feature allows you to create behaviors that you can invoke at any time. Once you've created a behavior, it appears in the Xcode > Behaviors menu.

To assign key equivalents to custom behaviors, in the Key Bindings preferences pane, select the Customized tab to find the behavior for which you want to assign a key equivalent.

Preprocessor or Assembly Output

Xcode 4.1 introduces commands in the Product > Generate Output menu to process source files and generate the preprocessed output or assembly output.

The preprocessor evaluates directives in your source code (instructions starting with the pound sign (#) such as includes, defines, and conditional logic) and converts them into C code to be sent to the compiler. You can examine the preprocessed output to make sure that the logic in your source code is being interpreted by the preprocessor as you expect in order to debug compilation problems.

The assembly output is the set of instructions that the compiler generated from the preprocessed output. You can study the assembly output to see how the instructions were formed or ordered, to seek out better optimization patterns, or to look for compiler bugs.

The output type is also a new category in the assistant editor for a selected primary file.

Autolayout

Interface Builder in Xcode 4.1 adds support for the new AppKit Autolayout feature. Autolayout, available starting with OS X v10.7 Lion, uses relationships called *constraints* to govern the layout of objects in the user interface. This feature is a complete replacement for the autoresizing mask. As you make changes to any view or control in Interface Builder (move it around, resize it, change an attribute, add a subview, and so forth), Interface Builder automatically adds and removes constraints based on the new layout. When you enable this feature, Interface Builder shows the constraints as you work.

Note: Autolayout is available only in OS X v10.7 Lion and later. If you are running Xcode 4 in OS X v10.6 Snow Leopard, Autolayout is not available.

Autolayout is enabled per nib. To start using Autolayout, select the Use Autolayout checkbox in the File inspector for each nib. When you enable Autolayout, Xcode updates your build settings, if necessary, to ensure your deployment target is OS X v10.7 or greater. For nib files in projects that had been targeted for OS X v10.6 or earlier, Interface Builder adds constraints to your nib files automatically when you enable Autolayout.

You can edit an automatic constraint or add your own. To add a constraint, select the view or views for which you want to add the constraint and choose a constraint from the Editor > Add Constraint menu.

You can delete a user constraint, but not an automatic constraint. Rather than trying to delete automatic constraints, create the constraints you want.

Build Setting Values in Scheme Pre- and Post-Action Scripts

There is often a need to access values from the target build settings in pre- and post-action scripts (similar to the way shell script build phases work). This enhancement allows a pre- or post-action script to define the target build settings to use via a pop-up menu in the scheme editor's pre- and post-action panes.

OS X Application Sandbox

An application sandbox enforces restrictions, known as *entitlements*, on how an application can interact with the rest of the system. A sandboxed application is harder to compromise and therefore enhances security for users. For example, if your application has no need to have access to the network, you can specify that the application's sandbox should prohibit network access. Then if a hostile hacker manages to take over control of your application on a user's computer, the hacker won't be able to use the application to send email or connect to the internet.

The iOS platform has supported entitlements for a while, and with Lion, OS X does as well. With Xcode 4.1, the project editor provides a UI for setting up entitlements for OS X applications. You can set entitlements for each target in the project editor. There is also a default code-signing entitlements file available in the file templates in the utilities pane.

Note: Although the OS X sandboxing UI is available in Xcode on v10.6 Snow Leopard, only OS X v10.7 Lion and later versions of the operating system enforce the entitlements.

When you enable application sandboxing, you can select an entitlements file if you already have one. If you do not, Xcode creates one with the name of your project and the extension `entitlements`. You can view and edit this file with the property list editor in Xcode.

Debugger Disassembly

The debugger disassembly feature provides you the ability to select what kind of content you view when debugging: source only (when available), disassembly only, or source and disassembly. The disassembly is the set of assembly-language instructions seen by the debugger while your program is running. Viewing the disassembly can give you more insight into what your code is doing when it's stopped at a breakpoint. By viewing the disassembly and source code together, you can more easily relate the instruction being executed with the code you wrote.

Choose Product > Debug Workflow to enable the disassembly-only display. The source and disassembly display is implemented as an Assistant category.

Git Remote Management

Support has been added in Xcode 4.1 for managing GIT remote repositories. You can select remote repositories from appropriate SCM workflows (such as push or pull). This feature enhances the ability of Xcode to be used for source control management.

In-Place Snapshot Restoration

With the Xcode 4.0 release, snapshots could not be restored on top of the current project content. The workflow for replacing the current version of your application with the version preserved in a snapshot involved restoring the snapshot, and then using the Finder to delete the current version and move the snapshot version into the appropriate folder. With Xcode 4.1, snapshots are automatically restored on top of the current version, unless you specify otherwise.

In Xcode 4.1, when you choose Restore Snapshot from the File menu and select the snapshot to restore, Xcode displays a preview dialog in which you can review the differences between the current version of the project and the snapshot version. When you click Restore, Xcode replaces the current version of the project with the version in the snapshot. Xcode makes a snapshot of the current version before replacing it.

To restore a snapshot in a new location instead of restoring on top of the current project, select the snapshot in the Projects pane of the Organizer window, choose the project you want to restore, and click the Export Snapshot button at the bottom of the window.

Internal Project Files in Repositories

This feature provides visibility into internal project files (schemes, user interface settings, and so forth) when looking at SCM details in the SCM commit and update workflows. You can use this facility to save and keep track of versions of project files in the same way as you save and keep track of source files.

Scheme and Run-Destination Selection

The popup menu for selecting schemes and run destinations has been changed to a path control, in order to provide you the ability to select either the scheme or run destination independently. You can still select both in a single gesture (using the submenus of the schemes). For projects or workspaces having a large number of schemes and several run destinations, this path control makes the menu much shorter and easier to deal with.

Key Bindings for Closing a Project or a Workspace

In Xcode 4.0, there was no default key binding to close a project or workspace. If you held the Option key when clicking the close box in the upper-left corner of the workspace window, all tabs, windows, and the project were closed, so that when you reopened the project, your previous set of windows and tabs did not reopen. In Xcode 4.1, the Command-Option-W key combination closes the project or workspace. Also, when you hold the Option key and click the close box, the project or workspace closes without first closing all windows and tabs. In this way, your window configuration is restored the next time you open the project.

Interface Builder Plug-in Support

If you used Interface Builder plug-ins in Xcode 3, you can continue to build and run your project in Xcode 4, and you can update your project to make your nib files editable in Xcode 4.

Xcode 4 provides limited support for Interface Builder 3 plug-ins. Specifically, you can build a project with Interface Builder plug-in dependencies, but you can’t edit the nib files. When you try to open a nib file with plug-in dependencies, Xcode 4 displays a dialog suggesting that you update the file. If you agree, Xcode converts the class of custom objects built with plug-ins to the nearest AppKit class. If the conversion isn’t possible, Xcode 4 provides a detailed error message. In that case, you must remove the plug-in dependency using Interface Builder 3 before you can edit the nib file in Xcode 4.

Document Revision History

This table describes the changes to *What's New in Xcode* .

Date	Notes
2015-06-30	Updated for Xcode 6.4.
2015-05-11	Updated for Xcode 6.3.2.
2015-04-20	Updated for Xcode 6.3.1.
2015-04-08	Updated for Xcode 6.2.
2015-03-09	Updated for Xcode 6.2.
2014-11-18	Updated for Xcode 6.1.1.
2014-10-16	Updated for Xcode 6.1 Beta.
2014-09-17	Updated for Xcode 6.
2014-04-10	Updated for Xcode 5.1.1.
2014-03-10	Updated for Xcode 5.1.
2013-11-11	Updated for Xcode 5.0.2.
2013-10-22	Updated for Xcode 5.0.1.
2013-09-18	Updated for Xcode 5.
2013-06-13	Describes the updates in Xcode 4.6.3.
2013-04-16	Describes the updates in Xcode 4.6.2.
2013-03-14	Describes the updates in Xcode 4.6.1.

Date	Notes
2013-01-28	Describes the new features in Xcode 4.6.
2012-09-19	Describes the new features in Xcode 4.5
2012-06-11	Describes changes for the Xcode 4.4 release. Describes changes for the Xcode 4.3.3 release.
2012-05-09	Updated for the Xcode 4.3.2 release.
2012-03-08	Describes the new features in Xcode 4.3.1
2012-02-16	Describes the new features in Xcode 4.3
2011-10-12	Describes the new features in Xcode 4.2.
2011-07-05	Describes how to use the features new in Xcode 4.1.