iOS Simulator User Guide

Developer

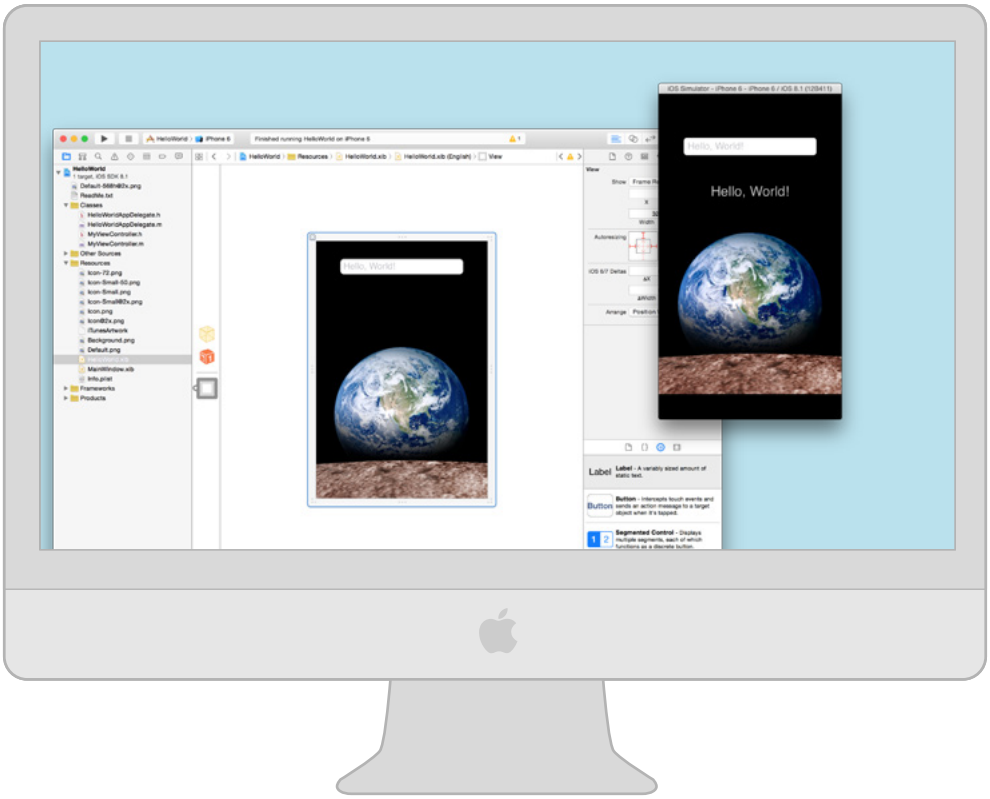
Contents

Figures and Tables

About iOS Simulator

iOS Simulator allows you to rapidly prototype and test builds of your app during the development process. Installed as part of the Xcode tools along with the iOS SDK, iOS Simulator runs on your Mac and behaves like a standard Mac app while simulating an iPhone, iPad, or Apple Watch environment. Think of the simulator as a preliminary testing tool to use before testing your app on an actual device.

iOS Simulator enables you to simulate different iOS and Apple Watch devices and several versions of the iOS operating system. Each combination of a simulated device and software version is considered its own simulation environment, independent of the others, with its own settings and files. These settings and files exist on every device you test within a simulation environment.



At a Glance

By simulating the operation of your app in iOS Simulator, you can:

- Find major problems in your app during design and early testing
- Test your app using developer tools that are available only for iOS Simulator
- Learn about the Xcode development experience and the iOS development environment before becoming a member of the iOS Developer Program

This guide walks you through iOS Simulator, starting with the basics of how to use it and moving on to the tools found within iOS Simulator that can assist you in testing and debugging your apps.

Organization of This Document

Read the following chapters to learn how to use iOS Simulator:

- Getting Started in iOS Simulator (page 7), to understand the functionality of iOS Simulator, and gain a working knowledge of the various ways to launch it
- Interacting with iOS Simulator (page 23), to learn about the various ways of interacting with iOS Simulator, including gestures and hardware manipulation
- Testing and Debugging in iOS Simulator (page 39), to understand the tools available within iOS Simulator to assist you with testing and debugging your apps
- Customizing Your iOS Simulator Experience with Xcode Schemes (page 50), to learn about additional ways to customize your iOS Simulator experience through Xcode schemes

See Also

Apple provides these related documents that you may find helpful:

- To learn the basics of developing iOS apps, see *Start Developing iOS Apps Today*.
- To learn more about how you can customize your development experience within Xcode, see *Xcode Overview*.
- To learn about the process of testing your app on a device, submitting it to the App Store, and distributing it, see *App Distribution Quick Start*.

Getting Started in iOS Simulator

The iOS Simulator app, available within Xcode, presents the iPhone, iPad, or Apple Watch user interface in a window on your Mac computer. You interact with iOS Simulator by using the keyboard and the mouse to emulate taps, device rotation, and other user actions.

The chapter presents the basics of using iOS Simulator. You can perform these steps using your own iOS app or, if you do not have an app to use, with the *HelloWorld* sample code. For more detailed information on interacting with iOS Simulator and using it to test and debug your apps, refer to the later chapters in this guide.

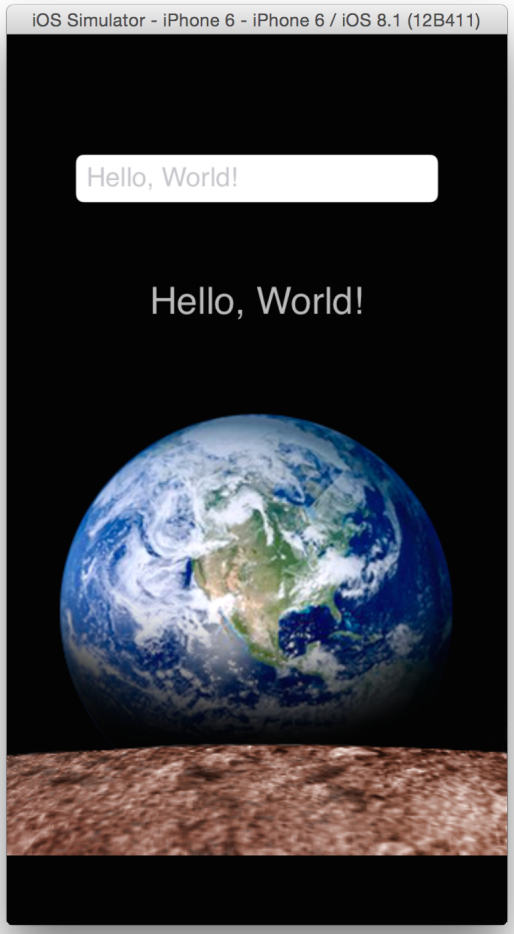
Access iOS Simulator from Xcode

There are two different ways to access iOS Simulator through Xcode. The first way is to run your app in iOS Simulator, and the second way is to launch iOS Simulator without running an app.

Running Your App in iOS Simulator

When testing an app in iOS Simulator, it is easiest to launch and run your app in iOS Simulator directly from your Xcode project. To run your app in iOS Simulator, choose an iOS simulator—for example, iPhone 6 or iPad Air—from the Xcode scheme pop-up menu and click Run. Xcode builds your project and then launches the most recent version of your app running in iOS Simulator on your Mac screen, as shown in Figure 1-1.

Figure 1-1 Simulated iPhone running the HelloWorld app



> **Note:** If you are testing an app with a deployment target of iPad, you can test only on a simulated iPad. If you are testing an app with a deployment target of iPhone or universal, you can test on either a simulated iPhone or a simulated iPad.

Launching iOS Simulator Without Running an App

At times, you may want to launch iOS Simulator without running an app. This approach is helpful if you want to test how your app launches from the Home screen of a device or if you want to test a web app in Safari in iOS Simulator.

To launch iOS Simulator without running an app

1. Launch Xcode.
2. Do one of the following:
    - Choose Xcode > Open Developer Tool > iOS Simulator.
    - Control-click the Xcode icon in the Dock, and choose Open Developer Tool > iOS Simulator from the shortcut menu.

iOS Simulator opens and displays the Home screen of whichever simulated device was last used.

View the Installed Apps

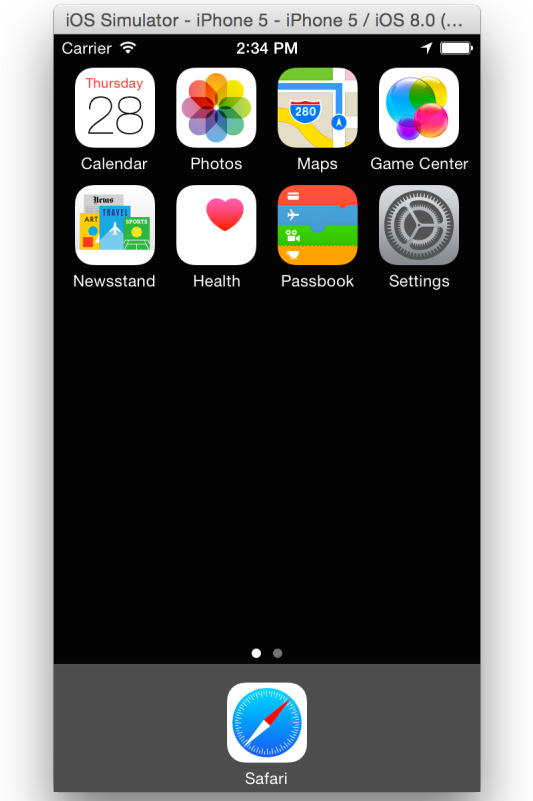
From the Home screen, you have access to all of the apps that are installed in the iOS simulation environment. There are two ways to access the Home screen in iOS Simulator from your app:

- Press Command-Shift-H.
- Choose Hardware > Home.

Much like the Home screen on an iOS device, the simulator's Home screen has multiple pages. After clicking the Home button (or accessing the Home screen through the Hardware menu), you arrive at the second page of the Home screen. To get to the first page, where all of the preinstalled apps are found, swipe to the first Home screen by dragging to the right on the simulator screen.

On the Home screen, you see that all of the apps that have been preloaded into iOS Simulator. See Figure 1-2.

Figure 1-2 Home screen in iOS Simulator



The apps that you see on the Home screen are specific to the iOS device simulation environment. Because Passbook and the Health app are available only for the iPhone, these apps don't appear if you are simulating a legacy device or an unsupported device type.

Use the installed apps to test your app's interaction with them. For example, if you are testing a game, you can use iOS Simulator to ensure that the game is using Game Center correctly.

Use Safari to Test Web Apps

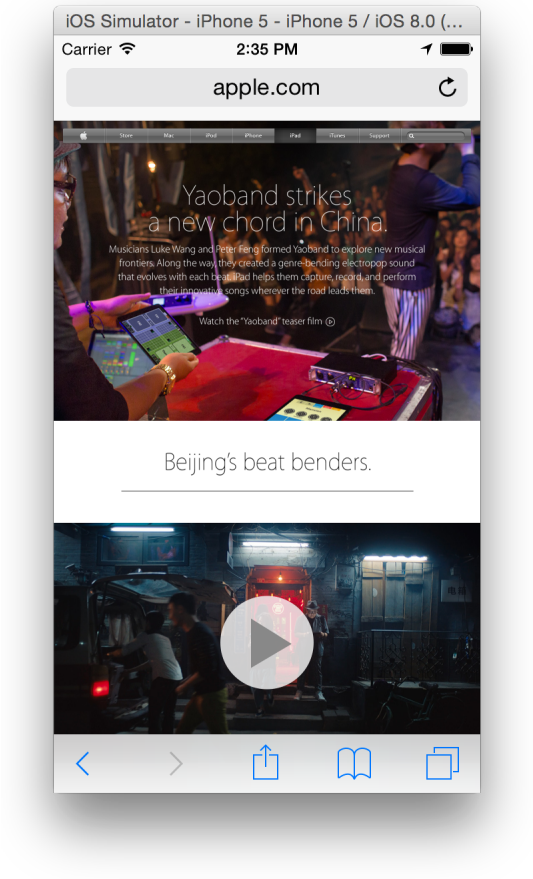
From the Home screen, you can access Safari within iOS Simulator. Use Safari to test your iOS web apps directly on your Mac.

1. From the Home screen, click Safari.

2. In the address field in Safari, type the URL of your web app and press the Return key.

If your Mac is connected to the Internet, it displays the mobile version of the URL you specified. For example, type `apple.com` into the address field and press Return. Safari displays the Apple website. See Figure 1-3.

Figure 1-3 The Apple website running in Safari in iOS Simulator



Use Maps to Simulate Location Awareness

iOS Simulator provides tools to assist you in debugging your iOS app. One of the many features you can debug in iOS Simulator is location awareness within your app. Set a location by choosing Debug > Location > *location of choice*. The menu has items to simulate a static location or following a route.

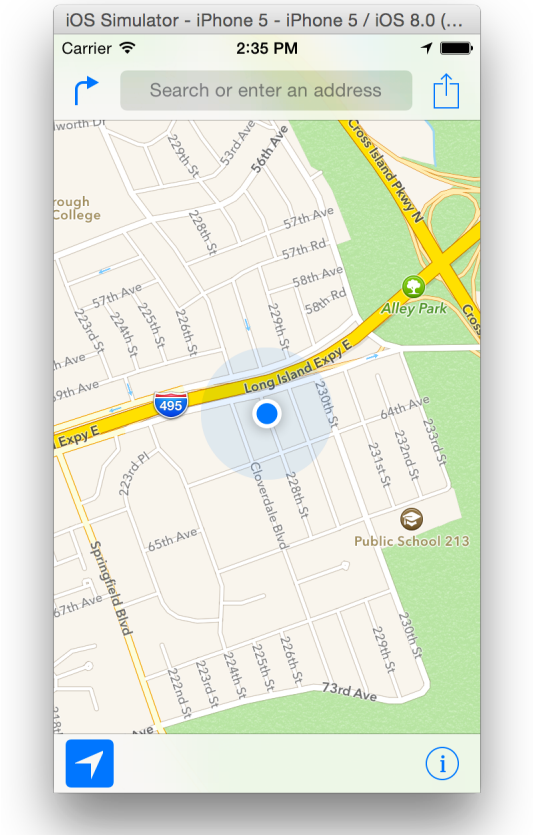
You can specify your own location, which can be seen in the Maps app.

1. From the Home screen, click Maps.
2. Choose Debug > Location > Custom Location.

3. In the window that appears, type the number `40.75` in the latitude field and the number `–73.75` in the longitude field.
4. Click OK.
5. Click the Current Location button in the bottom-left corner of the simulated device screen.

After completing this task, notice that the blue dot representing your location is in New York, NY, near the Long Island Expressway, as shown in Figure 1-4.

Figure 1-4 Running Maps and simulating a latitude and longitude in iOS Simulator



Change the Simulated Device and iOS Version

iOS Simulator provides the ability to simulate many different combinations of device type and iOS version. A *device type* is a model of iPhone or iPad, including the special *Resizable* type. Each device-iOS combination has its own simulation environment with its own settings and apps. iOS Simulator provides simulators for common device-iOS combinations. You can also add simulators for a specific combination you want to test. However, not all device type and iOS version combinations are available.

For information on using the resizable type, see `Resize the Simulated iPhone or iPad` (page 17).

> **Note:** To test apps for the iPad mini, use a simulated iPad with the same pixel resolution as the iPad mini.

You can switch between different device-iOS combinations. Switching closes the window for the existing device and then opens a new window with the selected device. The existing device goes through a normal iOS shut down sequence, though the timeout might be longer than the one on a real device. The new device goes through a normal iOS start up sequence.

To change the simulated device

1. Choose a Hardware > Device > *device of choice*.

   The iOS Simulator closes the active device window and opens a new window with the selected device.

If the device type and iOS version combination you want to use is not in the Device submenu, create a simulator for it.

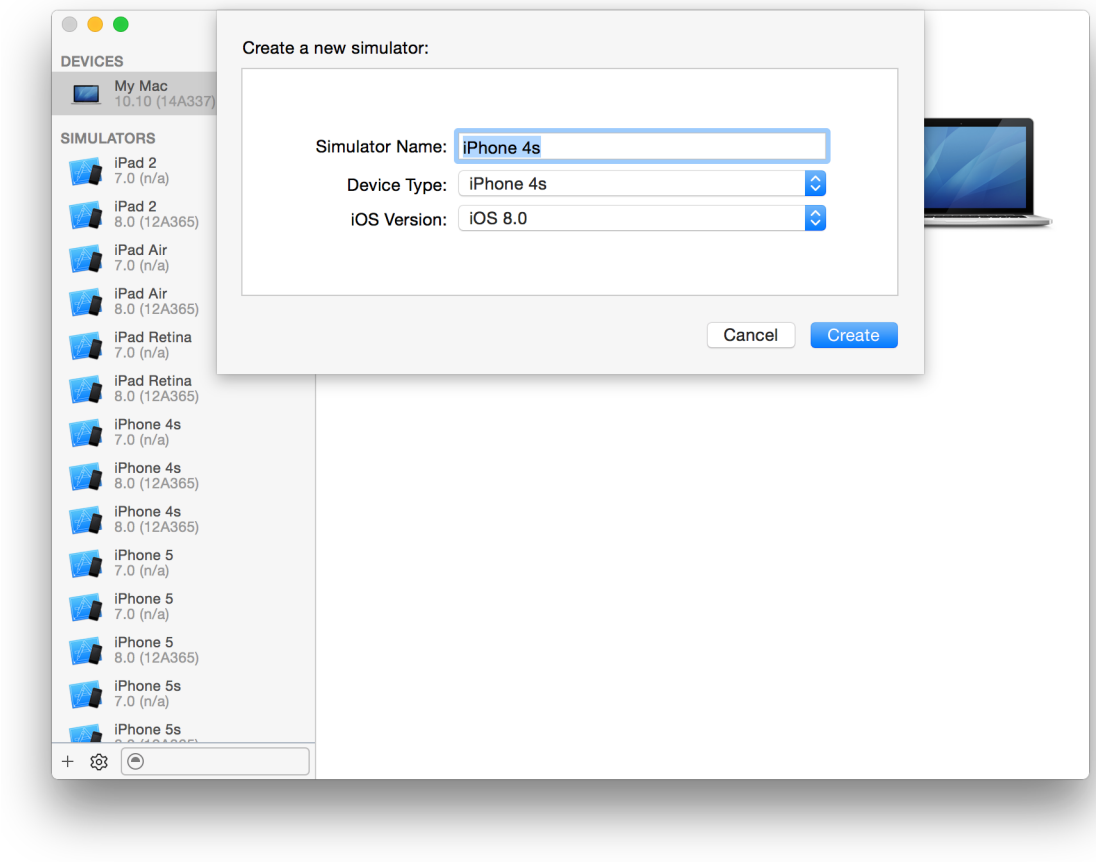
To add a simulator

1. Choose Hardware > Device > Manage Devices.

   Xcode opens the Devices window.

2. At the bottom of the left column, click the Add button (+).

3. In the dialog that appears, enter a name in the Simulator Name text field and choose the device from the Device Type pop-up menu.



4. Choose the iOS version from the iOS Version pop-up menu.

   Alternatively, if the iOS version you want to use isn't in the iOS Version pop-up menu, choose "Download more simulators" and follow the steps to download a simulator.

5. Click Create.

If the iOS version you want to use is not installed, download it and follow the steps to add a simulator again.

To download a simulator

1. In Xcode, choose Xcode > Preferences.
2. In the Preferences window, click Downloads.
3. In Components, find the legacy simulator version you want to add, and click the Install button.

You can also delete and rename simulators in the Devices window.

To delete a simulator

1. In iOS Simulator, choose Hardware > Device > Manage Devices, or in Xcode, choose Window > Devices.

   Xcode opens the Devices window.
2. In the left column, select the simulator.
3. At the bottom of the left column, click the Action button (the gear next to the Add button).
4. Choose Delete from the Action menu.
5. In the dialog that appears, click Delete.

To rename a simulator, choose Rename from the Action menu and enter a new name.

For how to manage real devices that appear in the Devices window, read *Devices Window Help*.

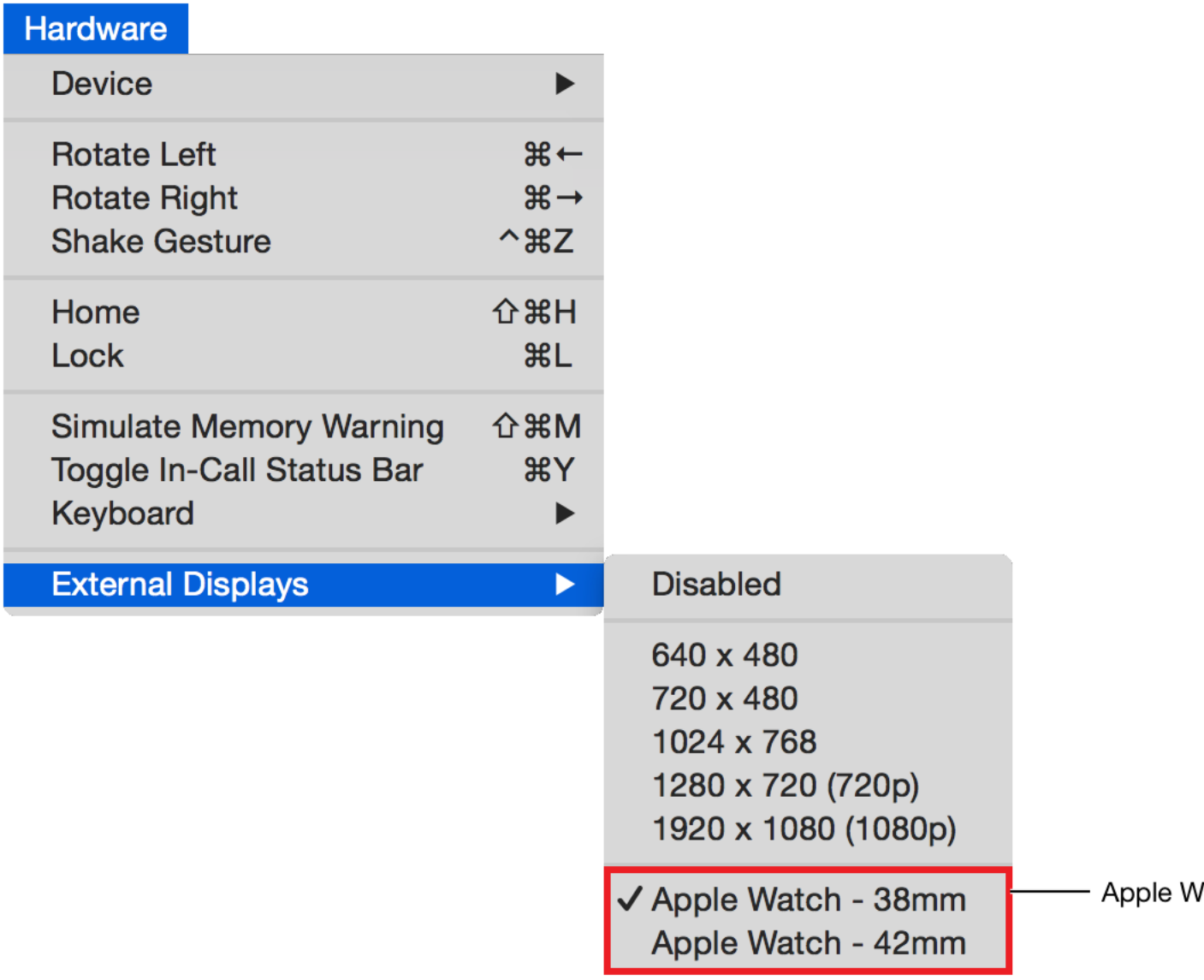
Showing the Apple Watch

iOS Simulator runs your WatchKit app in an Apple Watch, shown as an external display for an associated iPhone simulator. Make sure the simulated iPhone is running iOS 8.2 or later.

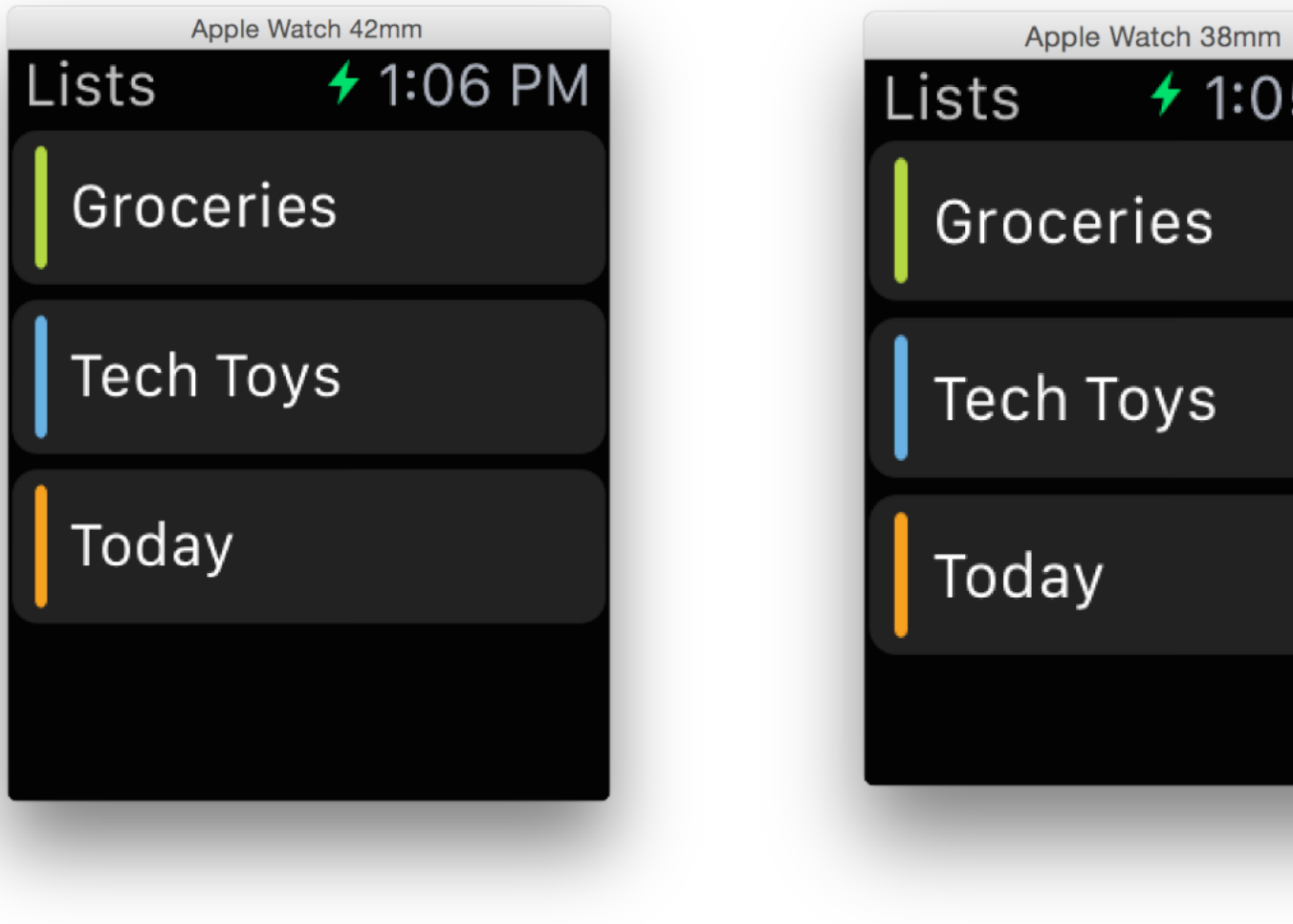
To show a simulated Apple Watch

1. Open any iPhone simulator running iOS 8.2 or greater.
2. Choose Hardware > External Displays, and select an Apple watch.

The External Displays menu lists all available sizes.



Run your WatchKit app from Xcode to show the Apple Watch simulator for the selected size. The figure below shows the Lister sample code running on the 38mm and 42mm screen sizes.



For information on how to build and run your WatchKit app, see The Build, Run, and Debug Process.

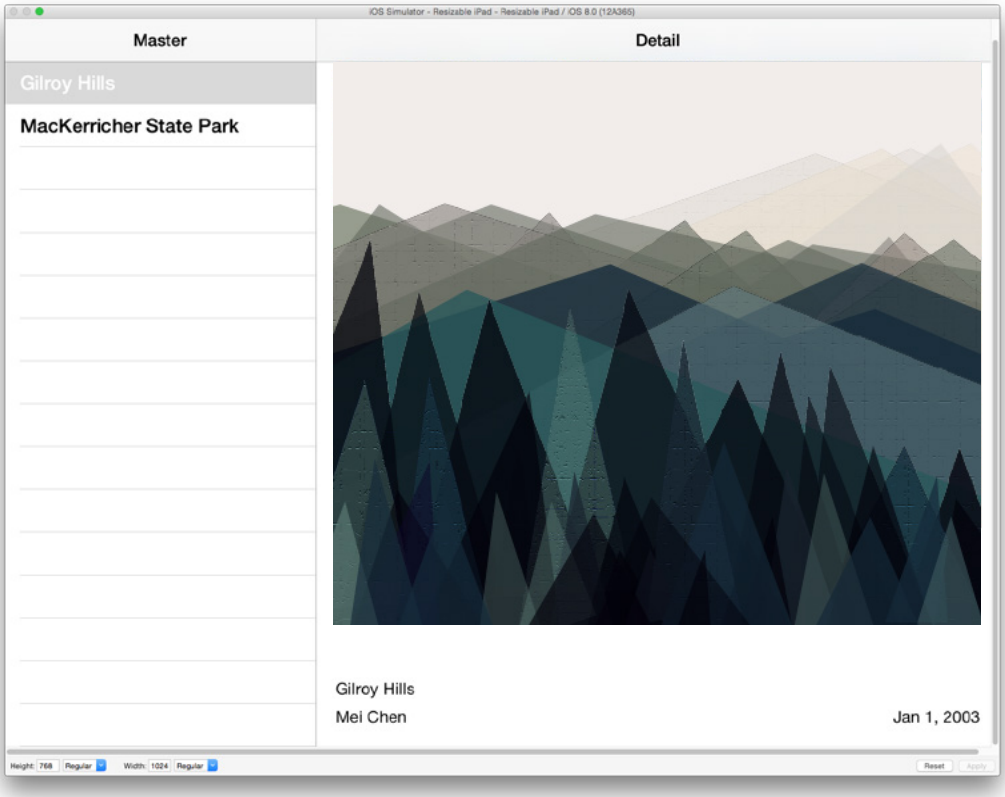
Resize the Simulated iPhone or iPad

To test your app using a variety of screen sizes within a reasonable range, choose a resizable device.

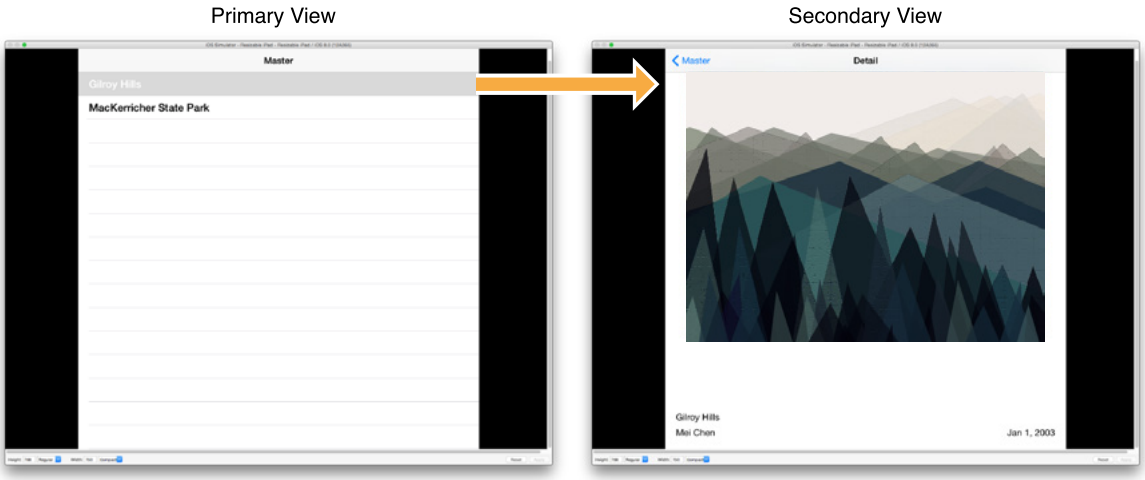
In Xcode, choose Resizable iPhone or Resizable iPad from the scheme pop-up menu and click Run. Alternatively, in iOS Simulator, choose Hardware > Device and then Resizable iPhone or Resizable iPad. At the bottom of the iOS Simulator window, enter the dimensions of the screen in the Width and Height text fields, and click Apply. iOS Simulator resizes the running app. If you choose a width or height that is outside the allowed range, iOS Simulator resizes the app to fit within the range.

Important: The iOS Simulator window resizes only a third-party app, not the Home screen or any app that ships with iOS.

If you use a single storyboard for all iOS devices, your app can adapt to different size classes. To simulate different size classes, choose Regular or Compact from the pop-up menu adjacent to the Width and Height text fields and click Apply. To simulate an iPad in portrait or landscape mode, set the width and height to Regular. For example, in a master-detail interface on an iPad, the table view appears on the left and a detail view appears on the right.



To simulate an iPhone in landscape mode, set the width and height to Compact. To simulate an iPhone in portrait mode, set the width to Compact and the height to Regular. For example, in a primary/secondary interface on an iPhone in portrait mode, the primary view fits the entire screen. Selecting an item in the table view transitions to the secondary view.

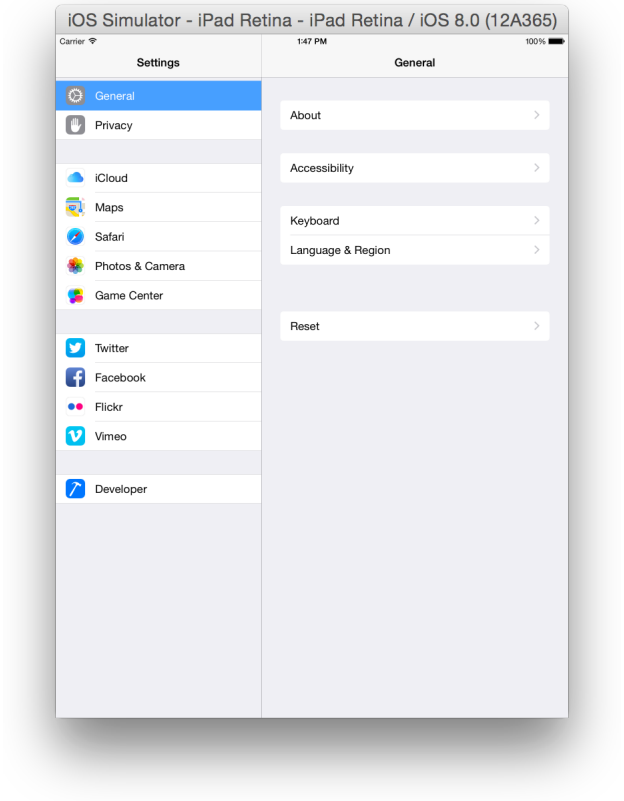


To learn how to preview different size classes in Interface Builder, read About Designing for Multiple Size Classes.

Alter the Settings in iOS Simulator

You can alter the settings within iOS Simulator to help test your app. To open the Settings app in iOS Simulator, go to the Home screen in iOS Simulator and click Settings. In Figure 1-5 you see the Settings app as it appears when launched in the iOS simulation environment.

Figure 1-5 Example of the Settings app in a simulated iPad device



The iOS Simulator settings differ from the settings found on a hardware device. iOS Simulator is designed for testing your apps, whereas a hardware device is designed for use. Because iOS Simulator is designed for testing apps, its settings are naturally focused on testing, too. For example, in iOS Simulator the Accessibility menu provides the ability to turn on the Accessibility Inspector, and the Accessibility menu on a device allows you to turn on and off different accessibility features.

Through the settings, you can test both accessibility and localization of your app. See Testing and Debugging in iOS Simulator (page 39) for information on how to manipulate your settings for the various types of testing you are interested in.

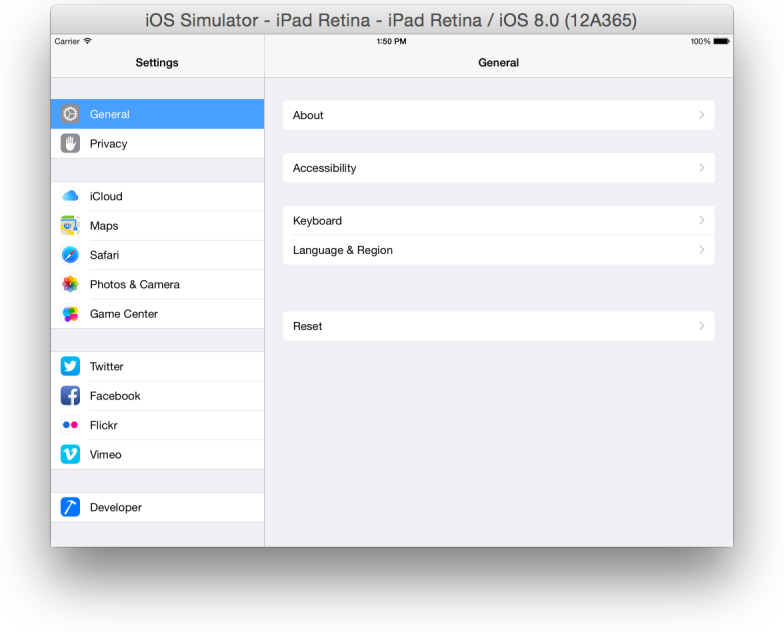
Remember: Changes made in the Settings app of iOS Simulator affect only the simulation environment that is currently running.

Rotate the Device

You can use iOS Simulator to manipulate the simulated device much as you do a physical device.

To rotate your simulated device, choose Hardware > Rotate Left. When you rotate your simulated device, Settings rotates (see Figure 1-6), just as it would on a hardware device.

Figure 1-6 A rotated simulated iPad running in the iOS simulation environment



Test in iOS Simulator and on a Device

iOS Simulator is designed to assist you in designing, rapidly prototyping, and testing your app, but it should never serve as your sole platform for testing. One reason is that not all apps are available in the simulator. For example, the Camera app is available only on hardware devices and cannot be replicated in the simulator.

In addition, not all bugs and performance problems can be caught through testing in iOS Simulator alone. You'll learn more about performance differences in Testing and Debugging in iOS Simulator (page 39). You can also find more information on testing your app on a device in Launching Your App on Devices in *App Distribution Guide*.

Quit iOS Simulator

iOS Simulator continues running until you quit it. Even if you quit Xcode, iOS Simulator continues to run because it is a separate app. To quit iOS Simulator, choose iOS Simulator > Quit iOS Simulator. If Xcode is running, Xcode remains open.

Interacting with iOS Simulator

Interacting with iOS Simulator differs from interacting with an actual device. In this chapter you learn how to:

- Simulate hardware actions such as rotate and shake
- Simulate Multi-Touch gestures using a mouse and keyboard
- Control the watch simulator window
- Uninstall an app you previously installed in a simulation environment
- Copy and paste text and images between the simulator and your Mac

Simulating Hardware Interactions

With iOS Simulator, you can simulate most of the actions a user performs on a device. Table 2-1 lists hardware manipulations you can perform in iOS Simulator by using the Hardware menu.

Table 2-1 Manipulating iOS Simulator from the Hardware menu

Menu item	Hardware action
Rotate Left	Rotates the simulator to the left.
Rotate Right	Rotates the simulator to the right.
Shake Gesture	Simulates shaking the device.
Home	Displays the Home screen of the simulated device.
Lock	Displays the Lock screen.
Simulate Memory Warning	Sends the frontmost app a simulated low-memory warning. For information on how to handle low-memory situations, see Responding to Low-Memory Warnings in iOS.

Menu item	Hardware action
Toggle In-Call Status Bar	Toggles the status bar between its normal state and its in-call state. This command shows how your app's user interface looks when a user launches your app during a call or while navigation is running. The in-call state bar is used when a phone call is in progress, a FaceTime call is in progress, or Maps in iOS 6 is navigating. The status bar is taller in its in-call state than in its normal state.
Keyboard > iOS Uses Same Layout As OS X	Automatically selects the iOS keyboard that most closely matches the keyboard layout of your Mac. Changing the keyboard layout on your Mac will change the layout on the simulated device
Keyboard > Connect Hardware Keyboard	Toggles between using the Mac keyboard as input into the simulator. This option simulates using a keyboard dock or a wireless keyboard.
Keyboard > Toggle Software Keyboard	Toggles the presence of the onscreen software keyboard. This option is available only if a hardware keyboard is connected to the device.
External Displays	Opens a window simulating an Apple Watch or the device's TV Out signal using the chosen resolution.

Simulating Keyboards in iOS Simulator

The iOS Simulator can use the keyboard on your Mac as input to the simulated device. For you to most accurately simulate a device in iOS Simulator, the simulator uses iOS keyboard layouts, as opposed to OS X keyboard layouts. If you have chosen Hardware > Keyboard > iOS Uses Same Keyboard Layout As OS X, iOS Simulator automatically selects the keyboard that most closely matches the keyboard layout of your Mac. For most cases, leave this option selected. However, if you do feel a need to disable it—allowing you to select completely different keyboard layouts for your Mac and iOS Simulator—choose Hardware > Keyboard > iOS Uses Same Keyboard Layout As OS X. Choose the same menu item again to enable the option.

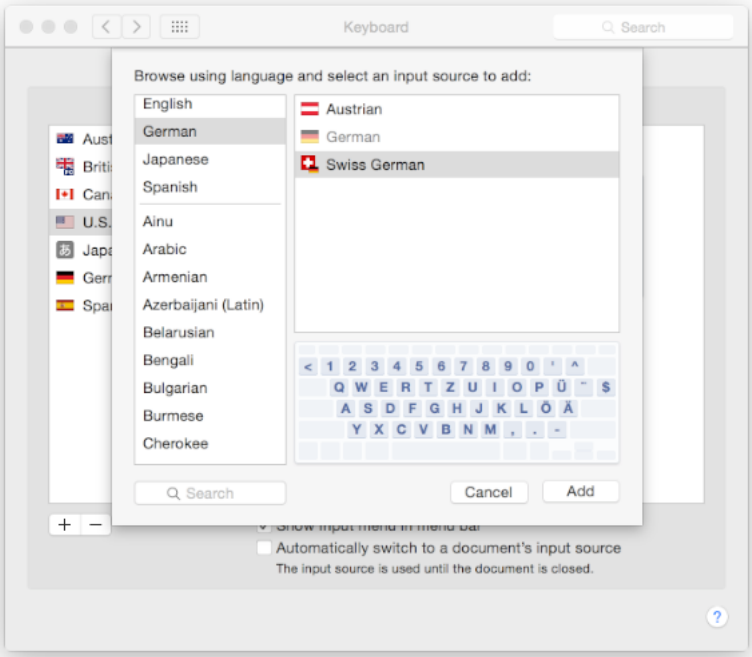
> **Note:** For the simulator to automatically switch keyboard layouts when the Mac layout is changed, both Connect Hardware Keyboard and iOS Uses Same Layout As OS X must be selected.

To add a keyboard layout on your Mac

1. Open System Preferences, and choose the Keyboard preference.
2. Select the Input Sources pane.
3. Press the Add button (+) to show the keyboard layout chooser.

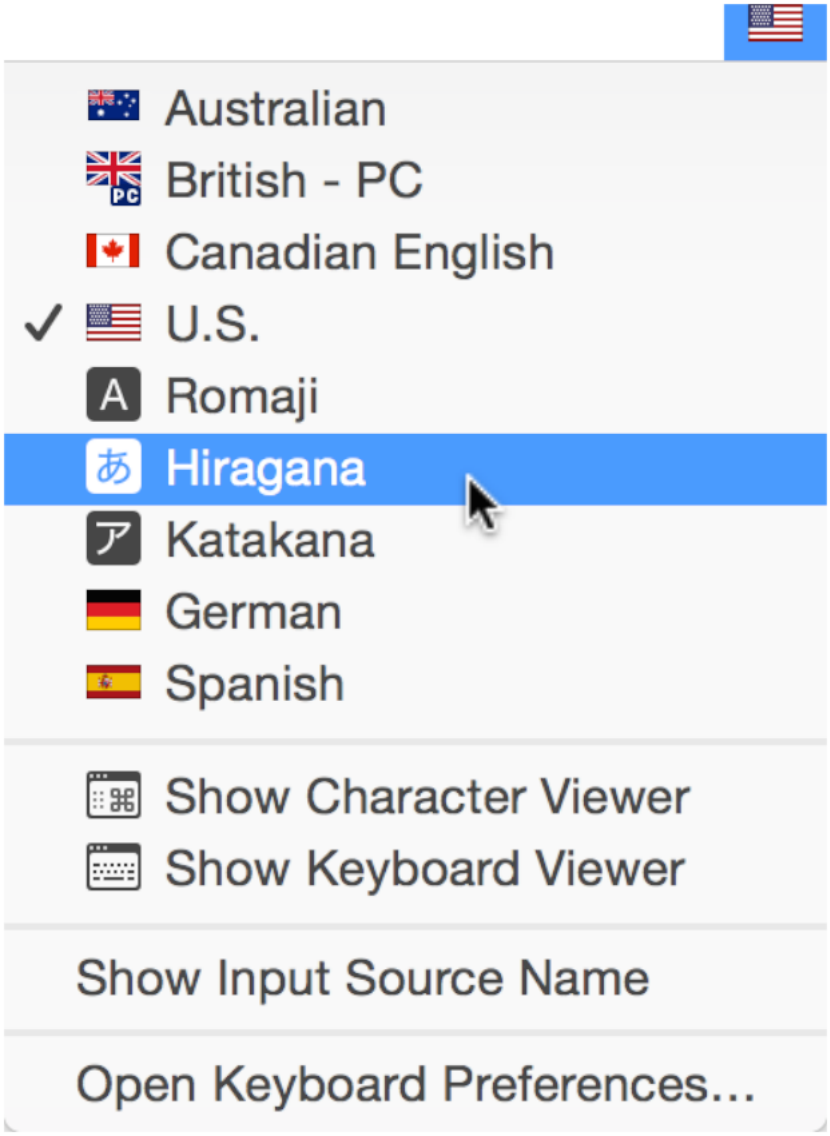
4. Choose the desired keyboard, and press Add. The new keyboard layout is added to the list of available layouts.

   This screenshot shows the keyboard layout chooser with the Swiss German layout selected:



To select a keyboard layout on your Mac

1. Select the desired keyboard from the Input menubar dropdown. An example menu is shown below.

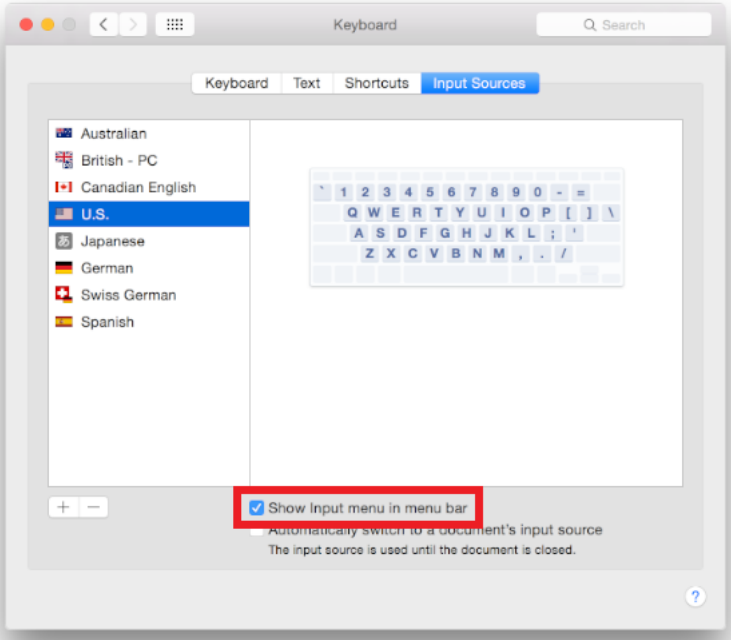


If the Input menubar item is not in the Mac menubar, use the following steps to add it:

a. Open System Preferences and choose the Keyboard preference.

b. Select the Input Sources pane.

c. Select “Show Input menu in menu bar” as shown here:



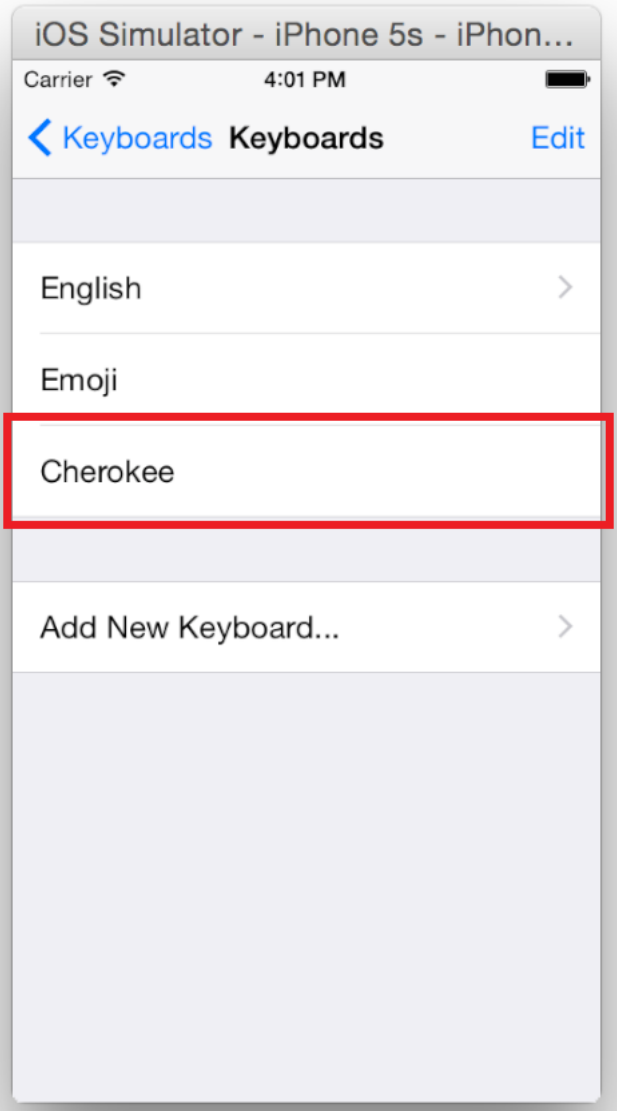
Selecting the keyboard layout on the simulated device

In addition to using the keyboard that most closely matches your Mac keyboard layout, you can also manually select a keyboard layout in the iOS Simulator settings. This approach can be helpful if you’re using a keyboard layout that iOS Simulator cannot automatically associate with a keyboard.

To add a new keyboard for a specific language and region

1. From the Home screen, open Settings.
2. Choose General > Keyboard > Keyboards > Add New Keyboard.
3. Choose a language and a keyboard layout.
4. Tap Done. The new keyboard is now available as soon as the user selects it.

Here is what the screen looks like after adding a Cherokee keyboard:



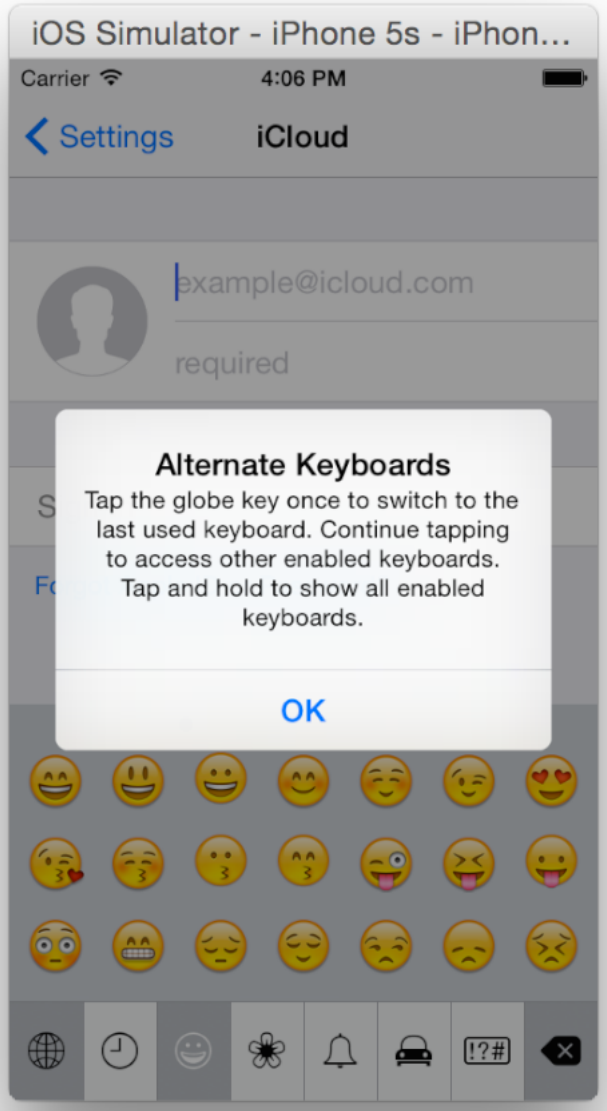
To show the new keyboard

1. Open the keyboard on the simulator.

   This can be done by tapping in any text entry view on the simulated device.

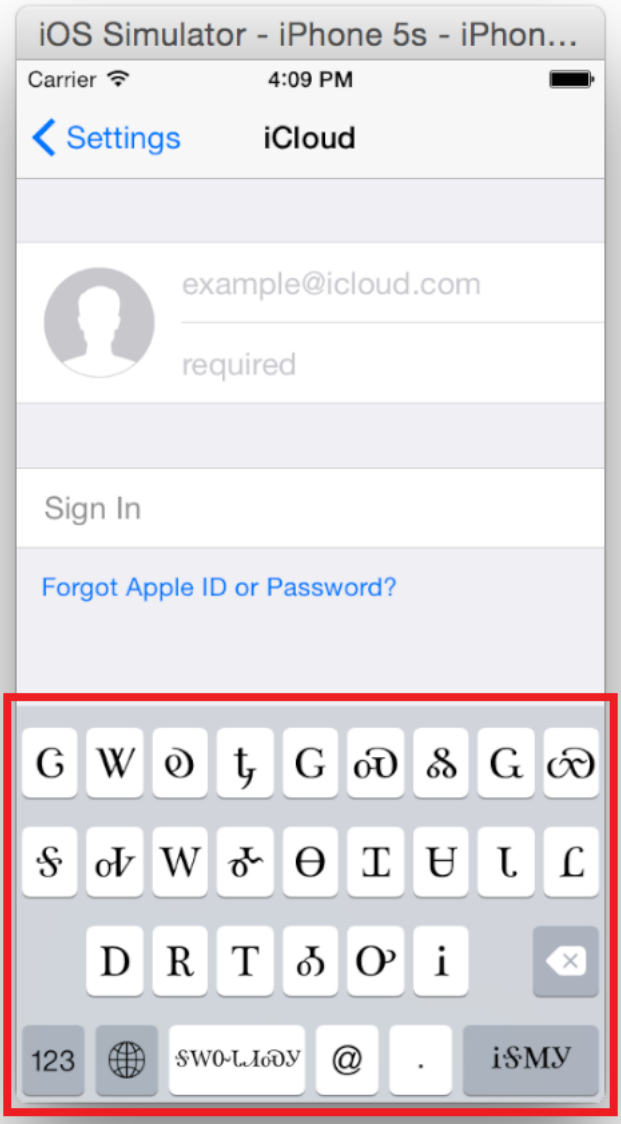
2. Tap the Globe key on the keyboard.

If the Globe button displays an alert like the one shown here, dismiss the alert.



3. Tap the Globe key until the desired keyboard is shown.

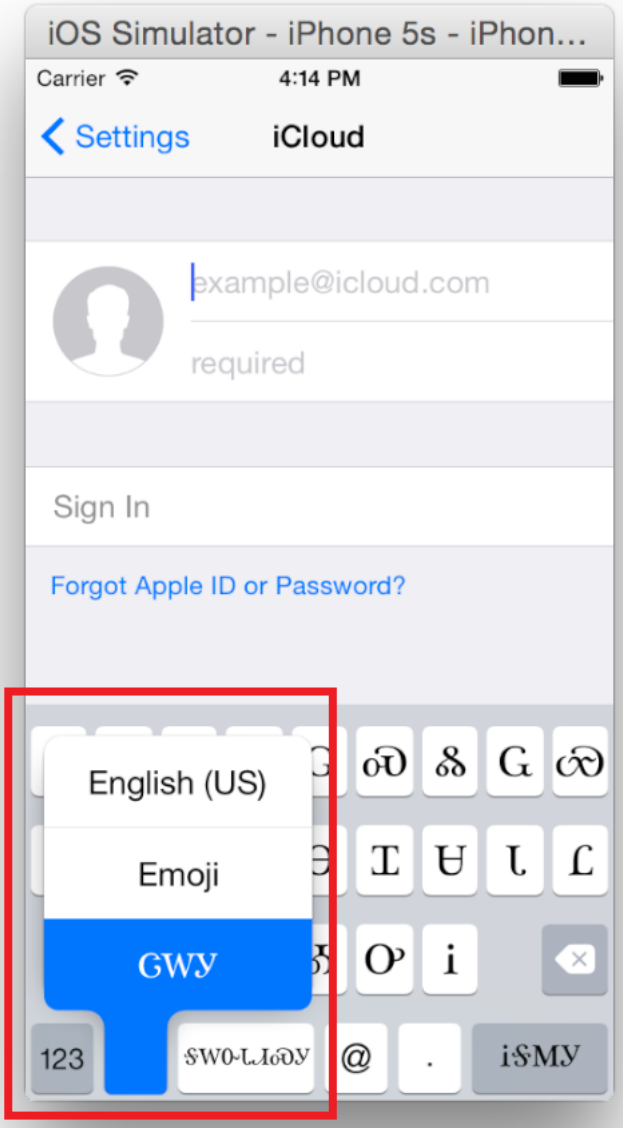
Here is a screenshot of the Cherokee keyboard that was added earlier:



Alternate

Tap and hold down the Globe key to see a pop-up menu of keyboards and select a keyboard from that list.

The screenshot below shows the list with the Cherokee keyboard selected.



Simulating User Gestures

With iOS Simulator, you can perform traditional Multi-Touch gestures using the mouse and keyboard. Table 2-2 lists gestures you can perform in iOS Simulator. For more information about gestures, see *iOS Human Interface Guidelines*.

> **Note:** All gestures can be performed using a mouse or a trackpad.

Table 2-2 Performing gestures in iOS Simulator

Gesture	Desktop action
Tap	Click.
Touch and hold	Press and hold down the mouse button or trackpad.
Double-tap	Double-click.
Drag	Drag.
Swipe	Drag.
Flick	Drag quickly.
Two-finger drag	1. Place the pointer where you want the two-finger drag to occur. 2. Hold down the Option key. 3. Move the circles that represent finger touches to the start position. 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. 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.
Pinch	1. Place the pointer where you want the pinch to occur. 2. Hold down the Option key. 3. Move the circles that represent finger touches to the start position. 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. 5. Hold down the mouse button, move the circles in and out to the end position, and release the Option key.

Gesture	Desktop action
Rotate	1. Place the pointer where you want the rotation to occur. 2. Hold down the Option key. 3. Move the circles that represent finger touches to the start position. 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. 5. Hold down the mouse button, rotate the circles to the end position, and release the Option key.

Simulating Watch Interactions

With iOS Simulator you can simulate most of the interactions with the Apple Watch using the mouse and keyboard. Table 2-3 lists the simulated interactions you can perform in the watch simulator.

Table 2-3 Interacting with the watch simulator

Interaction	Desktop action
Tap	Click.
Double-tap	Double-click.
Force touch	Press and hold down the mouse button or trackpad.
Twist the crown clockwise	Drag up in the content window of the watch.
Twist the crown counter-clockwise	Drag down in the content window of the watch.
Twist the crown quickly	Drag quickly.

Installing and Uninstalling Apps

When you build your app for iOS Simulator, Xcode automatically installs it in the selected simulation environment. Each simulation environment emulates a different device. Installing your app in one environment does not install it in any other. It is also possible to have different versions of your app in different environments.

Note: You cannot install apps from the App Store in simulation environments.

To uninstall apps that you have installed in a simulation environment

1. Select the simulation environment from which to remove the app by choosing Hardware > Devices > *device of choice*.
2. Place the pointer on the icon of the app you want to uninstall, and then press and hold down the mouse button or trackpad until the icons start to jiggle and a close button appears.
3. To uninstall the app, click the close button on the app icon.
4. In the dialog that appears, click Delete.
5. To stop the icons from jiggling, press Shift-Command-H or choose Hardware > Home.

Copying and Pasting in iOS Simulator

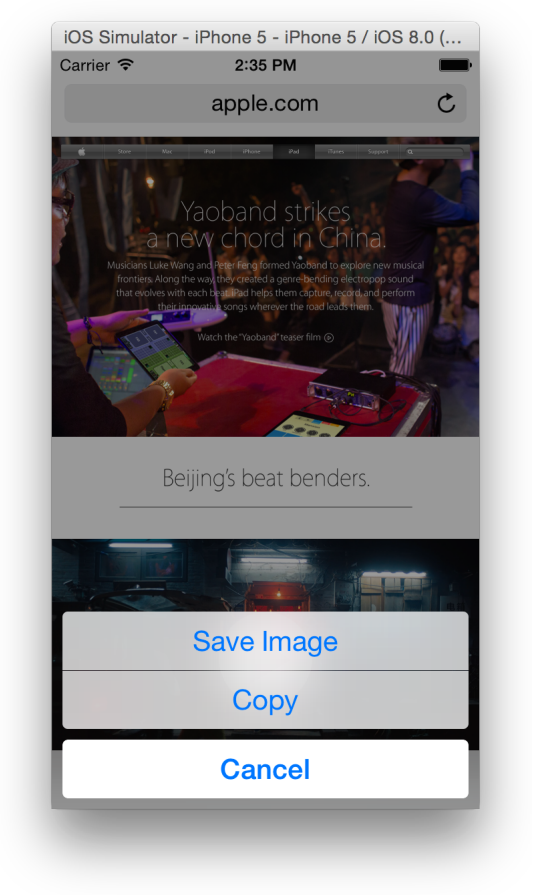
iOS Simulator provides a variety of copy and paste operations, both within the simulator and between the simulator and your Mac. The actual copy and paste operations in iOS Simulator are performed in the same way that they are on an iOS device, but if you are trying to copy and paste between the simulator and your Mac, additional steps must be taken. Copy and paste operations can be used on strings and images.

If you are copying an image from a webpage in iOS Simulator, save it to the Photos app first.

To save an image from a webpage to the Photos app

1. Place the pointer on the image you want to save, and hold down the mouse button or trackpad.

2. When the menu appears, click Save Image to save the image to the Photos app in iOS Simulator.



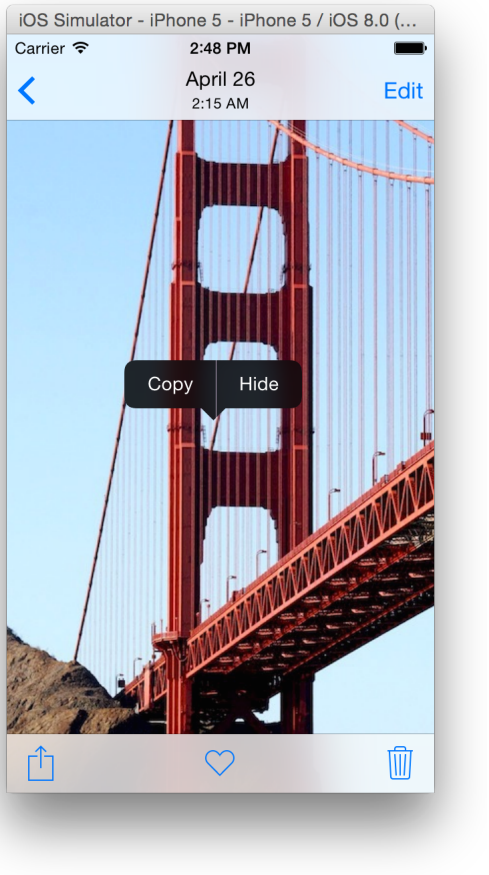
The image is saved to the Saved Photos album in the Photos app.

Alternatively, you can drag an image from the Finder on your Mac to iOS Simulator, and it is saved to the Saved Photos album.

To copy an image in iOS Simulator

1. In the Photos app, open the photo you want to copy.

2. Place the pointer on the image you want to copy, and press and hold down the Command key and the mouse button or trackpad.

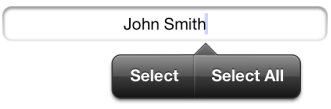


3. Click Copy.
4. If you intend to paste the image on your Mac outside iOS Simulator, choose Edit > Copy instead.

   This action copies the image to the Mac Clipboard. To paste the image in another app on the Mac, use that app's Paste command.

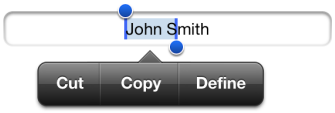
To copy text in iOS Simulator

1. Click the insertion point to display the selection buttons.



2. Click the Select button to select the adjacent word, or click Select All to select all text.
3. Drag the grab points to select more or less text.

4. Click Copy.



5. If you are pasting the text on your Mac outside of iOS Simulator, choose Edit > Copy.

   This action copies the text to the Mac Clipboard. To paste the text in another app on the Mac, use that app's Paste command.

To paste text into iOS Simulator

1. If the text you are pasting was copied from your Mac, choose Edit > Paste.

   This action copies the text from the Mac Clipboard to the simulator's Clipboard.

2. Navigate to the location where you want to paste the text you just copied.
3. Double-click the location where you want to paste the text, and then click Paste.

Taking a Screenshot of the Simulator

In iOS Simulator you can copy a screenshot of the iOS device simulator to your Mac Clipboard. To capture the Apple Watch or any simulated external display save the screenshot as a file.

- To take a screenshot of the iOS device and save it to your Mac Clipboard, choose Edit > Copy Screen.
- To save a screenshot of the iOS device and of any WatchKit app or external display as files, choose File > Save Screen Shot. A screenshot of each open simulated device is saved to the desktop of your Mac.

Viewing the Simulated Device's Screen

Even though iOS Simulator runs on all Mac computers, how it appears may differ between models. If the resolution of the simulated device is too large for the iOS Simulator window to fit on your screen, scale iOS Simulator by choosing Window > Scale > *percentage of choice*.

Testing Retina and Non-Retina Display Devices

With iOS Simulator, you can simulate iOS devices both with and without Retina displays, regardless of whether you have a Mac with Retina display.

> **Note:** The following equivalencies pertain when iOS Simulator is scaled to 100 percent; when scaled differently, the equivalencies scale in the same manner.

When working on a Mac without a Retina display, the simulator is mapped from pixel to pixel instead of from point to point. When simulating an app for an iOS device with Retina display on a Mac without a Retina display, the simulator appears twice as large as it would for a non-Retina display app to account for the extra pixels in a Retina display.

When working on a Mac with a Retina display, your computer maps each point in the iOS app to a point on the Mac screen. If the simulated app is for an iOS device with a Retina display, each point is composed of 1 pixel. If the app being simulated is for an iOS device without a Retina display, each point is composed of 2 pixels.

To learn more about mapping points to pixels, see Points Versus Pixels.

Testing and Debugging in iOS Simulator

iOS Simulator is a great tool for rapid prototyping and development before testing your app on a device. iOS Simulator also has features that can assist you in testing and debugging both iOS apps and web apps. By understanding the tools that iOS Simulator offers, you can more efficiently develop your app.

Differences Testing in iOS Simulator

The iOS Simulator is a useful tool, and it should not be the only way you test an app. Because the iOS Simulator is an app running on a Mac, it has access to the computer's resources, including the CPU, memory, and network connection. All of these resources are likely to be faster than those found on a mobile device. As a result, the simulator is not an accurate test of an app's performance, memory usage, and networking speed. For this same reason, always test the performance of your app's user interface on a device. In iOS Simulator, your app's user interface may appear to run both faster and smoother than on a device.

Also keep in mind that some user interface elements can be easier to interact with in iOS Simulator using a mouse than when trying to interact with the app through touch on a device.

Finally, there are some hardware and API differences in iOS Simulator. These differences may affect your app when testing in iOS Simulator.

Hardware Differences

Though most of the functionality of devices can be simulated in iOS Simulator, some hardware features must be tested directly on a device. The hardware features that are not simulated as of iOS 8.2 are:

- Motion support
  - Accelerometer
  - Gyroscope
- Audio and video input
  - Camera
  - Microphone
- Proximity sensor
- Barometer

- Ambient light sensor

Additionally, WatchKit apps have a reliable connection to the simulated host device because they both are running in the iOS Simulator.

To test your app on a device, you must be a member of the iOS Developer Program. To learn more about enrolling in the iOS Developer Program, see Managing Accounts in *App Distribution Guide*.

OpenGL ES Differences

iOS Simulator includes complete implementations of OpenGL ES 1.1, 2.0, and 3.0 that you can use to start developing your app. The capabilities of iOS Simulator are similar to those of the A7 GPU; for more information on the iOS hardware, see *iOS Device Compatibility Reference*. iOS Simulator differs from the hardware processor in a few ways:

- iOS Simulator does not use a tile-based deferred renderer.
- iOS Simulator does not provide a pixel-accurate match to the graphics hardware.
- Rendering performance of OpenGL ES in iOS Simulator has no relation to the performance of OpenGL ES on an actual device.

Important: The OpenGL ES support in iOS Simulator should be used to help you get started writing an OpenGL ES app. Never assume that iOS Simulator reflects the real-world performance or the precise capabilities of the graphics processors used in iOS devices. Always profile and optimize your drawing code on a real device.

API Differences

iOS Simulator APIs don't have all the features that are available on a device. For example, the APIs don't offer:

- Receiving and sending Apple push notifications
- Privacy alerts for access to Photos, Contacts, Calendar, and Reminders
- The `UIBackgroundModes` key
- iCloud document syncing and key-value storage support
- Handoff support

In addition, iOS Simulator doesn't support the following frameworks:

- External Accessory

- Media Player
- Message UI
- EventKit
- In UIKit, the `UIVideoEditorController` class

Backward Compatibility Support

iOS Simulator does not include backward compatibility with all versions of iOS.

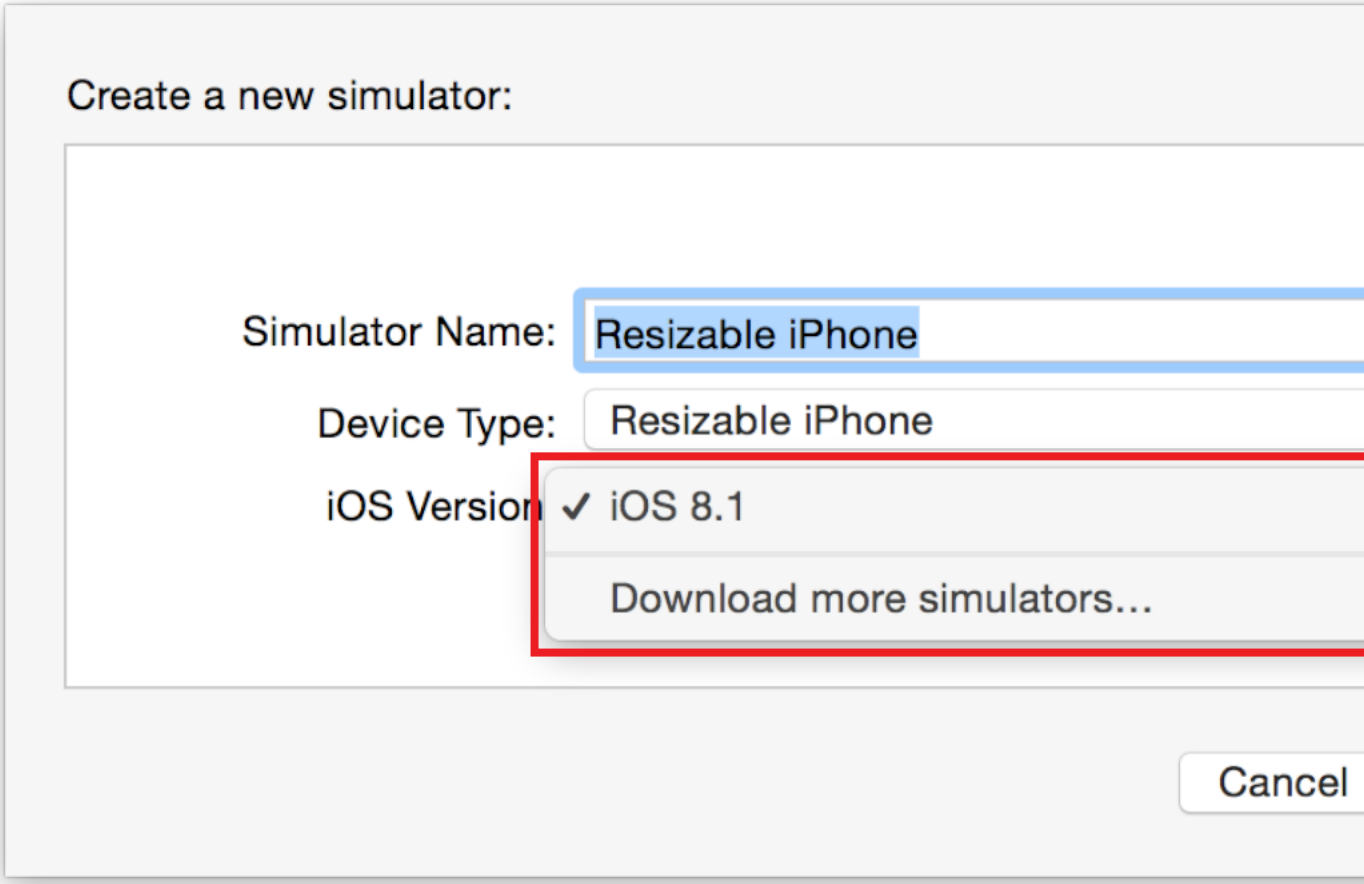
To find a list of the simulated versions of iOS in Xcode

1. Choose Hardware > Device > Manage Devices.

   Xcode opens the Devices window.

2. At the bottom of the left column, click the Add button (+).
3. In the dialog that appears, select the iOS Version pop-up menu.

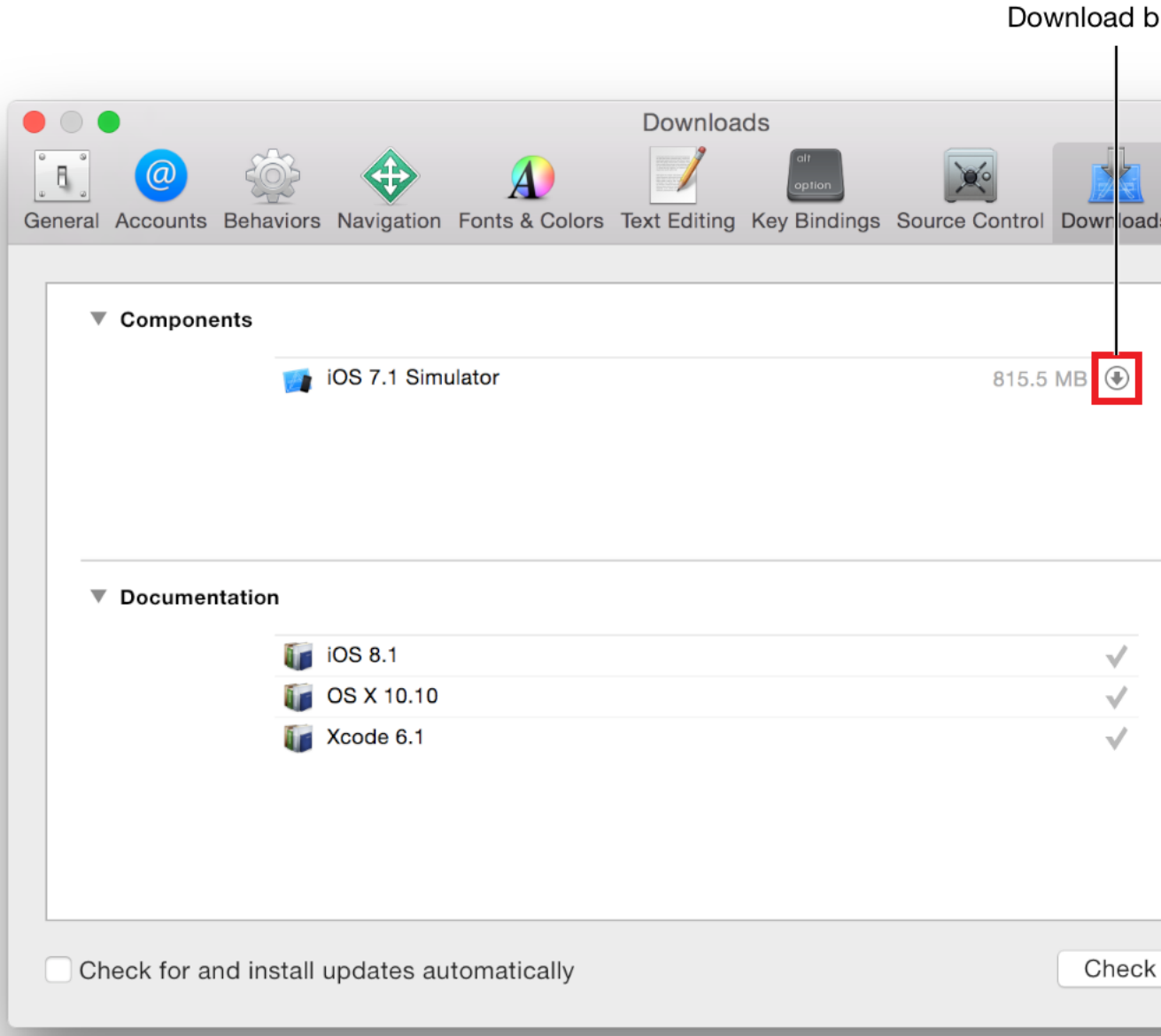
The top part of the menu lists any installed versions of iOS.



To download other available versions of iOS simulators

1. Choose Xcode > Preferences.
2. In the preferences window that appears, click the Downloads tab.

3. To download a simulator for a version of iOS, click the Download button.



After Xcode downloads the new iOS version it will be available for new device simulators. For information on adding a new device, see Change the Simulated Device and iOS Version (page 13).

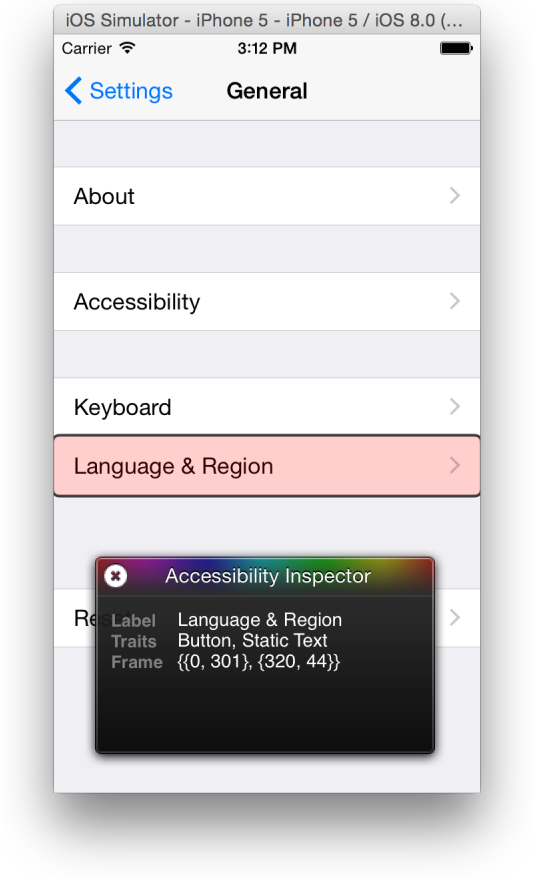
Testing for the iPad mini

To test apps for the iPad mini in the simulator, run your app on a simulated iPad with the corresponding type of display, either Retina or non-Retina depending on the iPad mini model.

Testing App Accessibility

Use the Accessibility Inspector to test the accessibility of your app. The Accessibility Inspector displays accessibility information about each accessible element in an app. Figure 3-1 shows what the Accessibility Inspector looks like as it runs in iOS Simulator.

Figure 3-1 The Accessibility Inspector running in a simulated iPhone

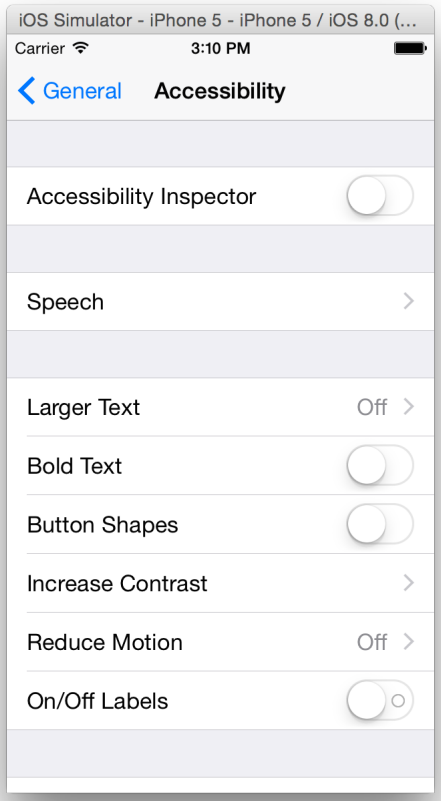


To start the Accessibility Inspector

1. When iOS Simulator is running, choose Hardware > Home to reveal the Home screen.
2. Click Settings.

3. Go to General > Accessibility.
4. Slide the Accessibility Inspector switch to On.

> **Note:** The Accessibility Inspector remains on until you turn it off, even if you quit and restart iOS Simulator.



Turning on the Accessibility Inspector in iOS Simulator alters the behavior of the simulator. After the Accessibility Inspector is on, clicking an element moves the focus of the inspector to that element instead of activating it. To activate an element, you must double-click it. Additionally, swiping and dragging gestures are unsupported while the Accessibility Inspector is enabled. To perform these gestures, you must first disable the Accessibility Inspector.

- To disable and reenable the Accessibility Inspector, click the close button in the upper-left corner of the inspector panel (the close button looks like a circle with an “X” in it).

For more information on using the Accessibility Inspector and testing the accessibility of your app, see *Verifying App Accessibility on iOS*.

Testing App Localization

If you have created an app with multiple localizations, you can test them in iOS Simulator by changing the Internationalization settings. For how to set the language and region on iOS, read Reviewing Language and Region Settings on iOS Devices. For more information on localizing your app, read *Internationalization and Localization Guide*.

Testing Web Apps

If you are building a web app and want to test its usability on an iOS device, iOS Simulator can assist you.

To test a web app in iOS Simulator

1. Select the simulator environment you would like to test in by choosing Hardware > Device > *device of choice*.
2. Open Safari from the Home screen of iOS Simulator.
3. Navigate to the location of your web app in the browser.

For more information on creating web apps for iOS, see *Getting Started with iOS Web Apps*.

Testing iCloud

If you are building an app that uses iCloud, you can test iCloud syncing from within iOS Simulator before testing on physical devices. This can also assist you in testing iCloud syncing across many devices if you have a limited number of devices to test on.

To simulate iCloud syncing, you must first sign in to iOS Simulator using an Apple ID. It is strongly encouraged that you create and use a separate Apple ID specifically for testing iCloud in iOS Simulator.

> **Note:** iCloud simulator works only when simulating iOS 7.0 and later.

To sign in to iOS Simulator with your Apple ID

1. Launch iOS Simulator with a simulated device running iOS 7.1 or later.
2. From the Home screen, open Settings and select iCloud.
3. Enter your Apple ID and password, and click Sign In.

After signing in with your Apple ID, you can then test your iCloud syncing. To test to see whether your app is syncing properly with iCloud, choose Debug > Trigger iCloud sync.

Testing Background Fetching

If you are building an app that receives frequent content updates and you have enabled background fetching, you can test the background-fetch capability using Xcode and iOS Simulator. To simulate a background fetch, launch your app in iOS Simulator and then go to Xcode and choose Debug > Simulate Background Fetch.

You can also configure a scheme to control how Xcode launches your app. To enable your app to be launched directly into a suspended state, choose Product > Scheme > Edit Scheme and select the Background Fetch checkbox.

Using the Debugging Tools in iOS Simulator

Access the debugging tools in iOS Simulator through the Debug menu, as shown in Table 3-1.

Table 3-1 Performing debugging through the iOS Simulator Debug menu

Menu item	Debug result
Toggle Slow Animations in Frontmost App	Slows down the animation taking place within the app. Use to identify any problems in the animation. Toggle Slow Animations can also be activated by pressing the Shift key three times.
Color Blended Layers	Shows blended view layers. Multiple view layers that are drawn on top of each other with blending enabled are highlighted in red, while multiple view layers that are drawn without blending are highlighted in green. Reduce the amount of red in your app when this option is selected to dramatically improve your app's performance. Blended view layers are often the cause of slow table scrolling.
Color Copied Images	Places a blue overlay over images that are copied by Core Animation in blue.
Color Misaligned Images	Places a magenta overlay over images whose bounds are not aligned to the destination pixels. If there is not a magenta overlay, places yellow overlay over images drawn with a scale factor.
Color Off Screen Rendered	Places a yellow overlay on content that is rendered offscreen.

Menu item	Debug result
Location	Allows you to set the Core Location to be used by your app. Choose from the different location settings: • **None.** Does not return a location. Use for testing how an app responds when no location data is available. • **Custom Location.** Allows use of a custom latitude and longitude. • **Apple.** Uses the coordinates of the Apple Headquarters. • **City Bicycle Ride.** Simulates a bike ride in Cupertino, CA. This option simulates the device moving on a predefined route. • **City Run.** Simulates a run in Cupertino, CA. This option simulates the device moving on a predefined route. • **Freeway Drive.** Simulates a drive through Cupertino, CA. This option simulates the device moving on a predefined route.

Viewing Crash Logs

If your app experiences a problem that causes it to crash, a crash log can help you determine what problem occurred. You open the crash log using Console.

To view a crash log

1. Open Console by going to `Applications/Utilities/Console` in the Finder.
2. Look for the line in Console that reads “Saved Crash Report for.”
3. Expand this item using the arrow at the left.
4. Click Open Report.

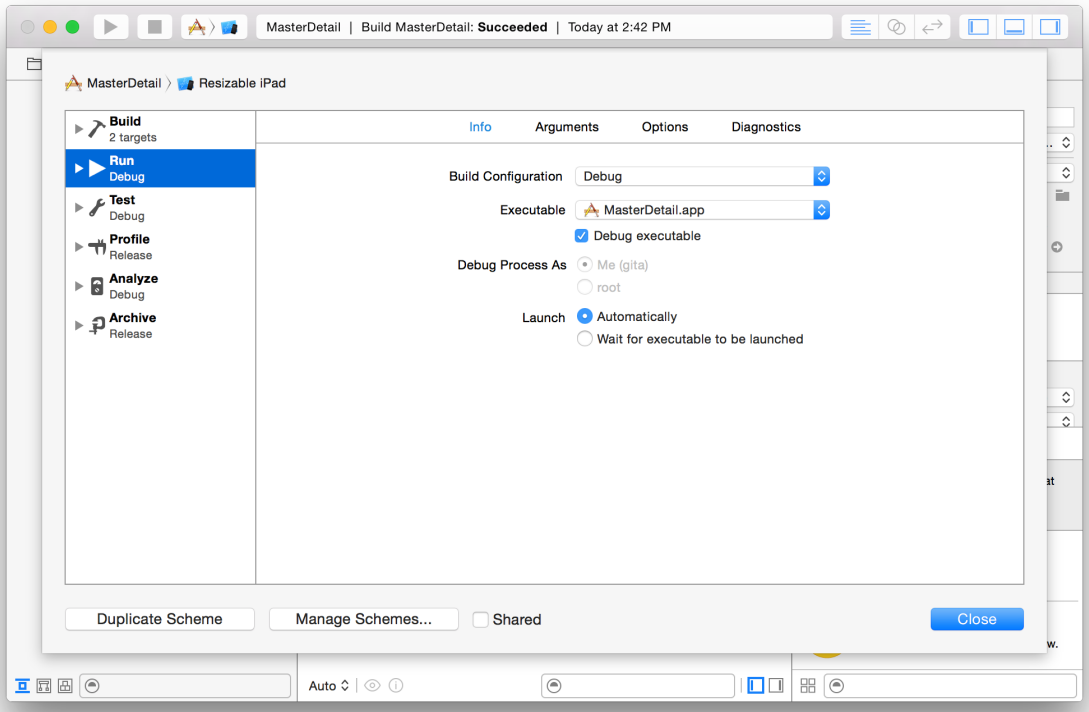
> **Note:** The simulated operating system maintains its own log, separate from the device log. To view the simulated operating system’s log, choose Debug > Open System Log.

Customizing Your iOS Simulator Experience with Xcode Schemes

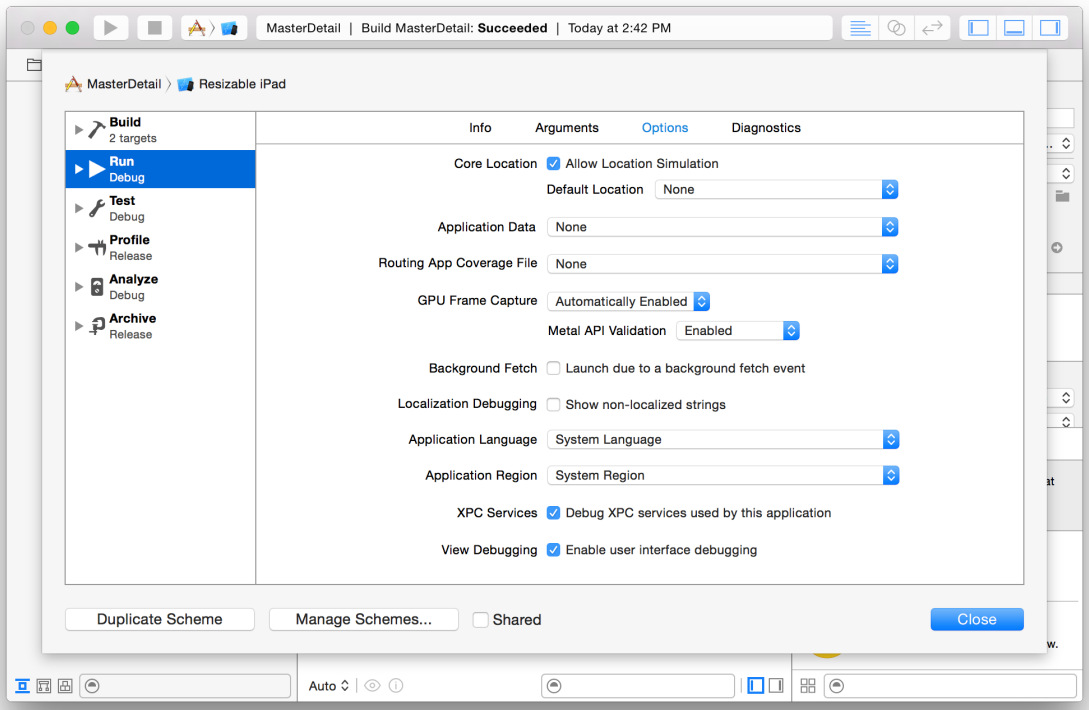
You can customize your iOS Simulator experience using the Xcode scheme editor. In fact, some iOS Simulator features are accessible only within this editor. The biggest advantage of using an Xcode scheme is the ability to load app data files and routing app coverage files.

To access scheme settings

1. Choose Product > Scheme > Edit Scheme.
2. In the dialog that appears, click the Run option in the scheme editor's left pane.



3. In the main pane, click Options.



4. Specify the options you want.

- **Core Location.** If you want to define the default Core Location setting, select Allow Location Simulation and choose a default location from the pop-up menu.

  To specify a route for the simulated device, choose "Use Add GPX File to Project" from the pop-up menu and select a file specifying the route in the GPX format. For more information on GPX, see The GPS Exchange Format.

  > **Note:** Xcode and the iOS Simulator support specifying a route as a series of waypoints using the GPX `<wpt>` tag. The route (`<rte>`) and track (`<trk>`) tags are not supported.

- **Application Data.** If you want to load app data into the simulator, choose the app data file from the pop-up menu. In this way, you can replicate the settings that were present when a problem occurred.
- **Routing App Coverage File.** If your app uses routing, use this file to define the location boundaries in which your app will provide routes. For more information on routing app coverage files, see *Location and Maps Programming Guide*.
- **GPU Frame Capture.** Enable this option if you need to capture a GPU frame in the debug navigator in order to analyze function calls that render the frame.

  For more information, see Xcode Scheme Settings and Performance.

- **Background Fetch.** Select this option if you want Xcode to launch your app directly into a suspended state.
- **View Debugging.** Select this option to enable debugging the view hierarchy of the simulated app in Xcode.

  For more information, see Examine Your App's View Hierarchy at Runtime

For a description of the localization options, read Testing Your Internationalized App.

5. Click Close.

For more information on using schemes in Xcode, see Edit, Create, and Manage Schemes.

Document Revision History

This table describes the changes to *iOS Simulator User Guide*.

Date	Notes
2015-03-09	Added information on simulating WatchKit apps and made other updates and revisions.
2014-11-15	Applied minor edits throughout.
2014-10-20	Applied minor edits throughout.
2014-09-17	Updated for Xcode 6.
2014-03-10	Updated for Xcode 5.1.
2013-10-22	Updated links to App Distribution Guide.
2013-09-18	Updated to include iOS Simulator changes in Xcode 5.
2013-04-23	Minor updates and the addition of information on testing Bluetooth using iOS Simulator.
2013-01-28	New document that explains how to test iOS apps on a Mac computer during development.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010