# 4.0 Lesson - Xcode Fundamentals

Paul Solt - Paul@SuperEasyApps.com
SuperEasyApps.com

---

## 4.1.1 Lecture - Xcode Fundamentals Introduction and Setup

Xcode is a powerful tool and will take a long time to really master. With that said, you really don't need to know most of it in order to make your first app.

As a beginner, it is very important to understand what's really important to know, and what you can learn as you go. You'll use this approach for this entire course.

Xcode is a tool that allows you to create apps, which are bundles of executable code (compiled), artwork, design assets, sound files, video files, and data files. It enables you to create multimedia applications that can be sold around the world using Apple's App Stores for any Apple device.

### Links

- Xcode Guide
- WWDC 2015 Video on Xcode 7 - apple.com

### Questions and Comments

- [Chapter 1] 4.0 Lesson - Xcode Fundamentals

## 4.1.2 Important - Additional Help and Support

Important: If you want fast help from me, one of my TAs, or another student, you must watch the lessons:

- 4.6 Tutorial - The Right Way to Add Files to Xcode
- 4.7 Tutorial - How to Share your iPhone App Project using ZIP files

## 4.2.1 Tutorial - Xcode Playground 101 Using Swift 2

Xcode has a new feature called an Xcode Playground, which allows you to write code in a scratch space.

### Links

- WWDC 2014 Swift Playgrounds - apple.com
- WWDC 2015: Authoring Rich Playgrounds - apple.com
- Apple Bug Report

### Questions and Comments

- [Chapter 1] 4.0 Lesson - Xcode Fundamentals

### 4.2.2 Bug Fix - Avoid HUGE Images in an Xcode Playground

Make sure you don’t use 15 megapixel images in your playground. Open your image and edit the size using the Preview app on your Mac.

Preview > Tools > Adjust Size (600 x 600 pixels or smaller for a playground)

### Questions and Comments

- [Chapter 1] 4.0 Lesson - Xcode Fundamentals

## 4.3 Tutorial - When to Use an Xcode Project vs. Xcode Playground

As a beginner you should default to creating an Xcode project, unless one of my lessons starts with a playground. Until Xcode 8, I can't really recommend playground files as the default starting point.

Playground files show you how code might work, but they don't show you how that code fits into a real app that you can run on your iPhone. Some code features behave differently in a playground than in a working iPhone app Xcode project.

### Questions and Comments

- [Chapter 1] 4.0 Lesson - Xcode Fundamentals

## 4.4 Tutorial - How to Debug Your iPhone App With Xcode 7

When you hear the term "debug," you think of removing insects … and that used to have a literal meaning: bugs would literally get stuck in Vacuum tube computers (pre-transistor), and they required an operator to remove them before the computer could continue operation.

Today bugs are used to describe any case which an app isn't working as the user expects, the developer intends, or the business logic describes in the requirements of the app (as defined by you, your team, your client, or your boss). Apps may seem mystical, but they really are just like recipes–the computer will process one step, or command at a time.

Xcode's debugger allows you to literally step line-by-line through your source code. This is an incredibly enlightening exercise where you can see how code is actually executed on a modern CPU (i.e.: the brains).

## Links

- WWDC 2014 - Debugging in Xcode 6 - apple.com
- WWDC 2015 - Whats New in LLDB (Debugging) - apple.com

## Questions and Comments

- [Chapter 1] 4.0 Lesson - Xcode Fundamentals

# 4.5 Tutorial - When to Use the iOS Simulator vs. Your iPhone

When you are testing an iPhone app, the iOS Simulator (now just Simulator) is way faster than your iPhone (unless you have a very old 2008 Mac).

You should always use your Simulator when you write code, because it gives you the shortest feedback loop. You always want to minimize your feedback loop—it should be as short as possible to maximize your effectiveness and productivity.

Running your app on your iPhone is slower because your app needs to be copied over USB to your iPhone before it can start. When you run a simulator (unlike Android's Emulator), the code runs as fast as the CPU in your computer, which gives you the shortest feedback loop (i.e.: fastest!).

## When Should You Test on a Real iPhone?

You should test your logic on an iPhone when you reach any critical stage in your app. This might be when you need to use the accelerometer, camera, file system for saving or loading, location, or another sensor that isn't simulated on the Simulator.

As an example, when I'm writing code, I will test code as often as possible by using the Run command (Play button) with the keyboard shortcut: Command + R.

I test code hundreds of times as I write line by line, this reduces any issues with bugs because I can fix them as soon as possible.

I only test the app on a real device a few times per day, usually when I'm about to launch a new beta test or when I want to show off the app working on a real device.

As a beginner, you should use the Simulator, as it is the best use of your time. The more time you wait for your app to compile (the process of taking your logic and creating a fully functional app), the less productive you will be, and that also means that you will make less progress toward your launch date.

### Questions and Comments

- [Chapter 1] 4.0 Lesson - Xcode Fundamentals

## 4.6 Tutorial - the Right Way to Add Files to Xcode

When you add files to Xcode you need to make sure that you copy them into the project. As a beginner, you ALWAYS want to copy files into your project directory.

This lesson is very important to you as you learn. If you have a problem and you send me your project, I can't help you fix the problem until you re-submit an Xcode project that runs. This might delay something that's quick to fix (1 minute) and delay the solution to you.

### Links

- What does "don't break the build" really mean in the software engineering context? - quora.com
- Breaking the Build, Why is it a bad thing? - stackoverflow.com

### Questions and Comments

- [Chapter 1] 4.0 Lesson - Xcode Fundamentals

## 4.7 Tutorial - How to Share Your iPhone App Project Using ZIP Files

Sharing your project is essential for working in a team or getting help from me, my team, or other students online. In the previous lesson, you learned how to properly add resources to your project.

Now you need to learn how to properly ZIP up your project folder. Xcode projects and standalone Playground files both need to be zipped properly in order to share them.

### Steps

1. In Xcode, right click on the project settings.
2. Choose “Show in Finder.”
3. Right click on parent (top level) folder.
4. Click on Compress “Your Folder Name.”

### Xcode Project

You need to make sure you have the top level directory selected before you archive the folder on Mac. If you grab the wrong one, you’ll be missing files and the 3rd party won’t be able to help you. This must include the folder, any top level files (if they exist), and the .xcodeproj file.

### Playground

A playground is really a bundle of files that can include additional resources. Since it is not a single file, you need to ZIP it in order to share it. Just right click and then Archive it.

### Always Test Your Zipped Code

1. Drag the .zip file to your Desktop (or any other folder) and then double-click it.
2. Open the .xcodeproj file or the .playground file.
3. Run the app (Play button).

If you don’t have any issues after running your app, you’re good to go. Send the email, post the blog post, get help, and fix your issue quickly. This way you don’t have to play email- or message-tag with someone on a forum.

You’ll get answers *and* learn more quickly.

### Questions and Comments

- [Chapter 1] 4.0 Lesson - Xcode Fundamentals

# 4.8 Tutorial - Easy Xcode Project Backup and an Xcode Bug

As you work on your project–especially when you're new–there's a chance that you could mess things up.

Don't worry!

It's easy to make a mistake. I make them all the time. I make a ton of mistakes that never get recorded in my video lessons during my research and testing phase. It's ok to mess up–you can always fix it.

### Questions and Comments

- [Chapter 1] 4.0 Lesson - Xcode Fundamentals

## 4.9 Tutorial - Xcode Quick Help and Documentation

Xcode includes documentation and guides beginners on how to use the code provided by Apple, and this code lets you create apps by leveraging your smart-sensor packed devices.

In this lesson, you will learn how to use Quick Help (Option/Alt + Left-Click on code) and access the built-in API documentation (Command + 0).

In the next lesson, you'll learn a better and faster alternative to the documentation (viewer + search) that Apple provides with Dash, a 3rd party Mac app on the Mac App Store.

### Links

- iOS Developer Documentation - apple.com

### Questions and Comments

- [Chapter 1] 4.0 Lesson - Xcode Fundamentals

## 4.10 Tutorial - Dash Documentation - Faster and Better Than Xcode Documentation

Note: You don't have to buy this app, but it might help you become more productive if you use it to search your Xcode documentation faster. The QuickHelp plugin is pretty good, but you also might not want to use it, see the following video on how to remove it.

The built-in Xcode documentation is a great source of insight, but unfortunately, as of Xcode 7, it's not always the best place to go for help.

Searching the documentation in Xcode is slow, sluggish, and generally provides the wrong results or hides the results you really want, so please submit an Apple Bug report to help improve it. In the meantime, however, use Dash 3+ from the Mac App Store.

Dash is a documentation viewing app (not just Apple documentation), and it is very fast, provides accurate search results, and can even replace the default behavior of Xcode's Quick Help lookups using the keyboard shortcut: Option/Alt + Left-click code element.

### Links

- Dash 3 on the Mac App Store
- Xcode plugin for Dash 3 Quick Look
- Dash 3 user guide

### Questions and Comments

- [Chapter 1] 4.0 Lesson - Xcode Fundamentals

## 4.11 Tutorial - How to Uninstall the Dash Quick-look Xcode Plugin

If you want to leverage native Xcode Quick help in your code, you can remove the Dash quick help plug-in.

1. Go to Finder > Go > Go to Folder
2. Paste: `~/Library/Application Support/Developer/Shared/Xcode/Plug-ins`
3. Remove the `OMQuickHelp.xcplugin` file
4. Restart Xcode (reopen any files)

### Links

- Uninstall Dash Quick Help Plugin - OMQuickHelp

## Questions and Comments

- [Chapter 1] 4.0 Lesson - Xcode Fundamentals

# 4.12 Quiz - Xcode Fundamentals

## Test your knowledge of Xcode by taking the quiz:

- 4.11 Quiz - Xcode Fundamentals