Making Everything Easier!™

Swift™ For Dummies®

A Wiley Brand

Learn to:

- Get up and running with Swift on Xcode®
- Set up a playground environment to test Swift syntax quickly
- Collect, declare, and type data
- Create an app that uses location, mapping, and social media

Jesse Feiler

Swift™ For Dummies®

A Wiley Brand

by Jesse Feiler

For Dummies®
A Wiley Brand

Swift™ For Dummies®

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Library of Congress Control Number: 2014954655

ISBN 978-1-119-02222-0 (pbk); ISBN 978-1-119-02224-4(ebk); ISBN 978-1-119-02223-7 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Contents at a Glance

Table of Contents

Introduction

In June of 2014, one of the highlights of Apple's Worldwide Developers Conference (WWDC) was the announcement — a surprise to many attendees, including the multitudes of developers watching the videos around the world — of the development of a new language aimed at developers to use with iOS and OS X devices. Called Swift, it was presented as the language of the future for Apple's developers, but it was made very clear that it would cooperate with the existing basic development language — Objective-C. (In describing the ways Swift and Objective-C would interact, Apple repeatedly used the phrase "mix and match" — not only in the presentations at WWDC, but in other venues as well.)

Think about that date— Swift has only been around since June 2014: We're all beginners with Swift.

About This Book

Swift For Dummies is a beginner's introduction to Apple's new programming language. The book gets you started developing with Swift. You'll quickly see how to create projects in Swift from the built-in templates that are part of the Xcode development tool. From there, you delve into the features of the language, from the basic to the advanced. Some of these features are unique to Swift whereas other, possibly more familiar features were inherited from other programming languages.

Before we get started with Swift, consider these two points:

- **Apple has done this before, and they know how to do it.** On both the hardware and software sides, Apple has successfully managed transitions to new technologies. Developers have sometimes cheered, sometimes booed, and even sometimes not even noticed much difference, but nonetheless, Apple has managed to bring them along to a new technology that makes their lives easier and improves things for users.
- **The languages are only part of the development environment for Apple.** When you develop apps for iOS or OS X, you use the Xcode development tool (technically an Integrated Development Environment, or IDE), the Cocoa or Cocoa Touch frameworks, and a programming language — either Objective-C or Swift. What differentiates the iOS and OS X development environment from most others is that the language is only one-third of the overall environment, as well as the fact that a single company (Apple) controls all of that environment.

Conventions Used in This Book

Cocoa is the framework you use for developing Mac apps; Cocoa Touch is the framework for iOS apps. Both have a common heritage and many similar classes. In general, classes that start with `NS` are Cocoa classes, and classes that start with `UI` are Cocoa Touch classes. Many Cocoa `NS` classes are also used in Cocoa Touch, so you'll find both types of classes in many of your apps and in the sample code and templates.

Code examples in this book appear in a `monospaced` font so that they stand out a bit better. Some non-syntax components appear in an italicized monospaced font. (Thus, `weatherConditions` might be a variable, but *variable* could be any variable you want to use.)

Like many languages, including Objective-C, Swift is case-sensitive, so please enter the code that appears in this book *exactly* as it appears in the text. I also use the standard Cocoa naming conventions — such as capitalizing class names and leaving the names of methods and instance variables lowercase.

Note that all URLs in this book appear in a monospaced font as well. In accordance with common usage, most URLs in this book include the subdomain (such as `www`) at the beginning of many URLs except for addresses that don't require that component (such as `developer.apple.com`).

If you're ever uncertain about anything in the code, you can always look at the source code on my website at `www.northcountryconsulting.com` or the *For Dummies* website at `www.dummies.com`. From time to time, I'll provide updates for the code there and post other things you might find useful.

Foolish Assumptions

This book makes few assumptions about readers because Swift programmers come from many backgrounds and with varying degrees of proficiency in various languages. As to the future, however, there's one simple assumption: You want to create apps based on the Cocoa and Cocoa Touch frameworks, and you want to do it in the simplest way possible.

Fittingly, then, this book is aimed at Cocoa and Cocoa Touch developers at all stages of expertise, from those who've developed a multitude of App Store apps to those who have only thought about developing an app . . . someday.

I also assume you have some Mac or iOS experience. If you have never used a Mac or iOS device, you may find it hard to follow this book. I explain advanced technical terms as they arise, but my assumption is that you know, for example, what Settings (on iOS devices) and System Preferences (on Macs are), and that similar concepts are familiar to you.

You must have access to a Mac that can run the current version of Xcode (a free download from `developer.apple.com`). Without Xcode and the Mac to run it on, you can't experiment with the sample code.

REMEMBER Note that Xcode runs only on Macintosh computers running Mac OS X v10.9.4 (Mavericks) or later on a 64-bit Intel-based Mac.

Additionally, you must have Internet access. It's very important to stress, however, that I don't mean "always-on" Internet access. I only mean that you must at least have limited Internet access — so you can access the App Store, for example, and connect with Apple's `developer.apple.com` to download software and upload apps.

Perhaps the most foolish assumption of all may be your own: that you can't learn Swift or the Cocoa and Cocoa Touch frameworks. You can, and this book is designed to help you. Bear in mind that app development is not easy: If it were, the App Store would have far more than just over a million apps. It's not easy, but you can do it.

Icons Used in This Book

TIP This icon indicates a useful pointer that you shouldn't skip.

REMEMBER This icon represents a friendly reminder. It describes a vital point that you should keep in mind while proceeding through a particular section of the chapter.

TECHNICAL STUFF This icon signifies that the accompanying explanation may be informative (dare we say interesting?), but it isn't essential to understanding Swift. Feel free to skip past these tidbits if you like (though skipping while learning may be tricky).

WARNING! This icon alerts you to potential problems that you may encounter along the way. Read and obey these blurbs to avoid trouble.

Beyond the Book

A lot of extra content that you won't find in this book is available at `www.dummies.com`. Go online to find the following:

- **Source code for the examples in this book at**

  ```
  www.dummies.com/extras/swift
  ```

  This book contains a lot of code, and you might not want to type it. In fact, it's probably better if you don't type this code manually. Fortunately, you can find the source code for this book on the Dummies.com website at `www.dummies.com/extras/swift`. The source code is organized by chapter. The best way to work with a chapter is to download all the source code for it at one time.

- **Online articles covering additional topics at**

  ```
  www.dummies.com/extras/swift
  ```

  Here you'll find out how to know whether to use a type, collection, flow control, or function to implement an action; how to initialize stored properties in a class or structure; and how to let Xcode create actions and outlets for you.

  Ongoing discussions at `developer.apple.com` (for registered developers only) and at my website (`www.northcountryconsulting.com`) provide even more information.

- **The Cheat Sheet for this book is at**

  ```
  www.dummies.com/cheatsheet/swift
  ```

  Here you'll find an examination of the anatomy of a Swift class, the best way to update Xcode for a new Swift release, and advice about working with both Swift and Objective-C.

- **Updates to this book, if we have any, are also available at**

  ```
  www.dummies.com/extras/swift
  ```

Where to Go from Here

It's time to start your Swift adventure! If you're new to programming, start with Chapter 1 and progress through the book at a pace that allows you to absorb as much of the material as possible. If you're in an absolute rush to get going with Swift as quickly as possible, you could possibly skip to Chapter 2 with the understanding that you may find some topics a bit confusing later.

Part I
Getting Started with Swift

Visit `www.dummies.com` for great *For Dummies* content online.

In this part . . .

- ✓ Set up an Xcode Swift project.
- ✓ Find out how to use a playground.
- ✓ Explore the Xcode editing tools.
- ✓ Write your first Swift app.

Chapter 1

Setting Up an Xcode Swift Project

In This Chapter

- Introducing Swift
- Setting up your computer for Swift
- Defining your development preferences
- Creating and exploring your first project

Swift is Apple's new language for developers to use with iOS and OS X devices. As such, it is the successor to Apple's existing iOS/OS X development language, Objective-C, but Swift has been designed to cooperate with and work alongside Objective-C, so this should be a slow transition to power.

Some Swift beginners come to the language with proficiency in other languages, ranging from C and its offshoots such as C++ and Objective-C, to newer languages such as Ruby, Python, and Java, as well as scripting languages such as PHP and JavaScript.

Whether you're just starting out as an Apple developer or are an experienced developer who wants to add Swift to your skills, this chapter helps you get started. There's one very important point to remember: As of this writing, the iOS API (application programming interface) and SDK (software development kit) are less than ten years old. (They were launched in early 2008, six months after the launch of iPhone.) The early years of iOS development were an exciting period as the pieces of today's hardware and software environment fell into place. Only as thousands of developers and millions of users started actually using these devices and the languages that support them did some issues — bugs as well as great enhancements — begin to take shape.

Arguably, it took several years for the SDK to reach maturity. Many developers (including your humble author) believe that it was only with the release of iOS 4 in 2010 that the platform more or less stabilized as the operating system we recognize today. This was the first version to be called "iOS" rather than "iPhone OS," and, with the release of iOS 4.2.1 in the fall of 2010, it was the first to support both iPhone and iPad. The first version of multitasking was present, and preparations were made for iCloud that was first released in iOS 5.

If you haven't looked at iOS since that time, a lot has changed. The release of Swift and iOS 8 is a good opportunity to look around and get up to date with iOS (and, for that matter, OS X). This chapter helps you do that.

WARNING!

In this book, you'll occasionally find warnings like this one about serious issues you should avoid. The warnings are used sparingly, so pay attention to them when they appear. The focus in this book is on getting you up and running as a Swift developer. That involves giving you the information you need as well as helping you along the way with encouragement and, from time to time, reminding you that you're not the first person to learn Swift. Others have been there before, and, in most cases, others (most definitely including the author) have encountered the problems you may be facing. There are a multitude of warnings in this chapter. This isn't intended to scare you off: Rather, it's designed to help you over that first hump of becoming a Swift developer. After you have your first clean compile and have finished a build of a project (in the section, "Planning Your Environment," later in this chapter), you'll be on your way.

Looking Ahead to the End

As you make progress in Swift, this book helps you build an app — a real, live app — based on one of the built-in Xcode templates. *Sure,* you're probably thinking, *that's just what I need — another "Hello World" app.*

Actually, no. There's no "Hello World" here. Instead, the app you'll be building, called Locatapp, is a full-fledged Swift app created using the Master-Detail Application template that's built into Xcode, and it uses Cocoa Touch and a number of its frameworks to do its work. Locatapp uses location services on Cocoa Touch and the iOS mobile devices to find your location, as you see in Figure 1-1.

If you prefer, you can download Locatapp from this book's companion website, as described in the Introduction, but be warned — some of the details of registering as a developer described later in this chapter are needed to get Locatapp to run on your own device.

The pulsing blue dot shows your current location. Locatapp lets you store other locations you've visited. The latitude and longitude values of locations that have been visited are shown in the list at the left of Figure 1-1. Tap one

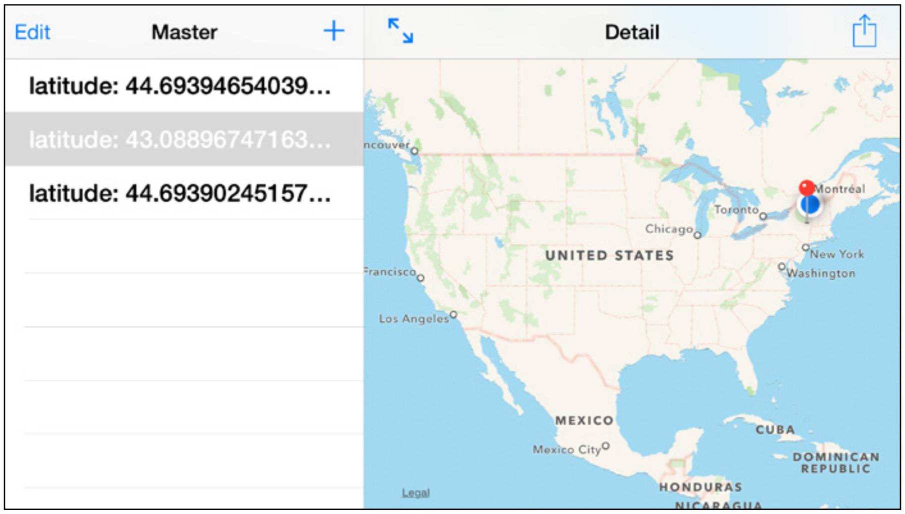

Edit
Master
Detail
latitude: 44.69394654039...
latitude: 43.08896747163...
latitude: 44.69390245157...
UNITED STATES
MEXICO
CUBA
Legal

Figure 1-1: Locatapp finds your location.

of them, and you'll see a map with your current location and with the tapped location indicated by a red pin.

You can zoom in on the map (see Figure 1-2). This zoom-in functionality is all built into MapKit and the device so you don't have to write any code. As you zoom in, you can see that the two locations shown in Figure 1-1 are actually over 100 miles apart. The annotation for Current Location is also part of the framework.

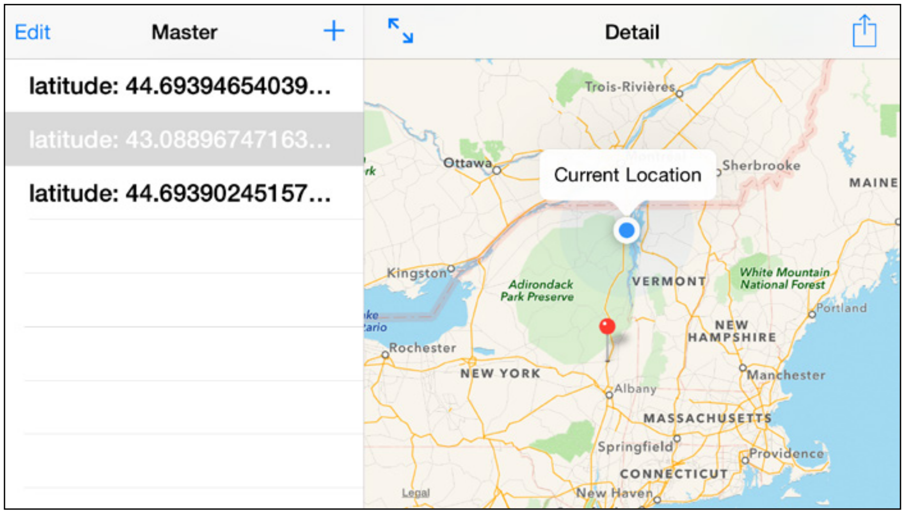

Edit
Master
Detail
latitude: 44.69394654039...
latitude: 43.08896747163...
latitude: 44.69390245157...
Current Location
VERMONT
NEW YORK
Legal

Figure 1-2: Zooming the map.

In addition to the built-in annotation for Current Location, you can write your own annotations in Locatapp. Figure 1-3 shows a custom annotation that you'll write in the course of this book.

That action button at the right of the bar in the interface (the box with the arrow poking out of the top) is an interface element you can drag from the Xcode library into your user interface (called a *storyboard*). What happens when you tap that action button depends on a method you'll build in this book. This method uses the built-in actions such as Messages, Mail, Twitter, Facebook, and so forth, as shown in Figure 1-4. (Yes, you'll write this code, but Cocoa Touch writes the supporting code to interact with Messages, Mail, Twitter, Facebook, and more.)

Figure 1-5 shows a tweet you can construct in your app. Users can modify it (note that there are 57 characters left), but you write the code for the message and to insert the map coordinates. Note, too, that the image of a web page is part of the tweet. You'll see how to automatically put that into the tweet. Although you can tap the image of the web page all you want in this book, in Locatapp, tapping that image will take you to the web page in Safari.

I'm sure you'll like Locatapp and enjoy thinking of ways you can build on it.

As I like to say, "Goodbye, 'Hello World.'"

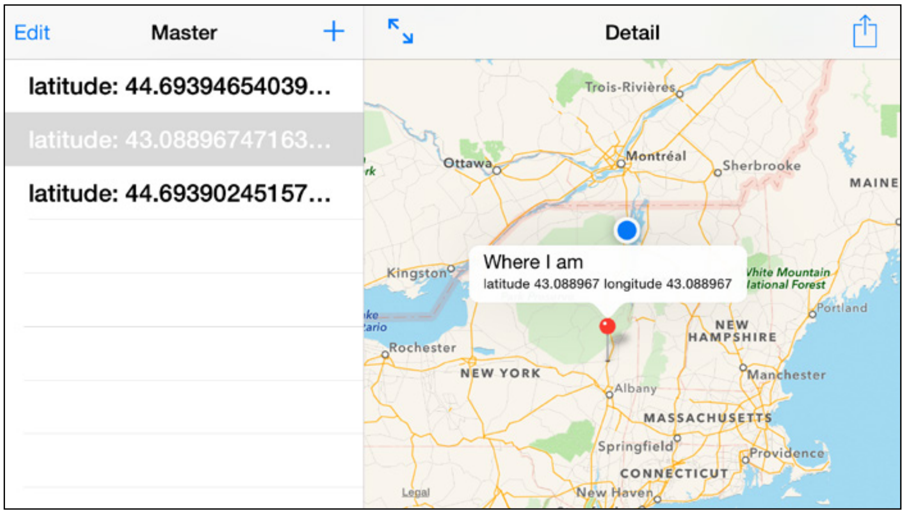


Figure 1-3: Writing your own annotations.

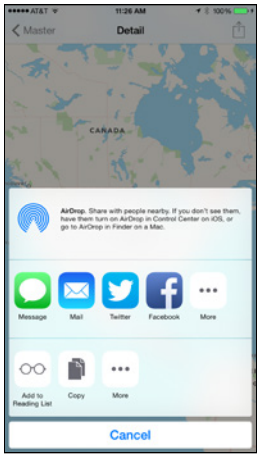
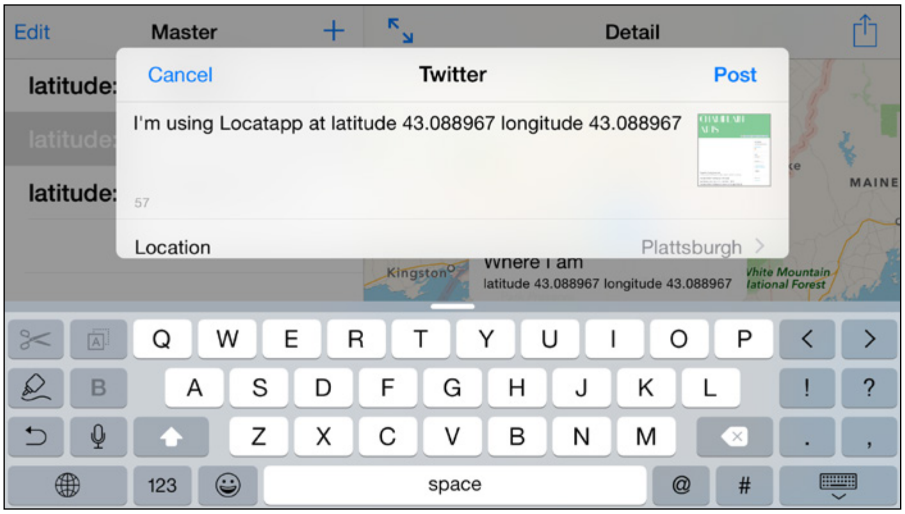

Master
Detail
CANADA
AirDrop. Share with people nearby. If you don't see them, have them turn on AirDrop in Control Center on iOS, or go to AirDrop in Finder on a Mac.
Message
Mail
Twitter
Facebook
More
Add to Reading List
Copy
More
Cancel

Figure 1-4: Implementing an action button.



I'm using Locatapp at latitude 43.088967 longitude 43.088967

57

Where I am

latitude 43.088967 longitude 43.088967

Figure 1-5: Constructing social media messages from your code.

Working with Swift

Apple has two annual calendars of events. Each is highlighted by one or more major announcements with periodic updates throughout the year. For consumers and end users, the annual calendar focuses on the releases of new and updated devices. As is true throughout the world of electronics, a large portion of annual sales occur during the summer and fall ("back-to-school") and during the year-end holiday season.

On the software side, there is a related peak period. It's no accident that Apple, Google, and Microsoft all hold conferences for their developers in May and June. Typically, they unveil the new features in their operating systems at that time, allowing developers a few months to work with those features before the peak period of hardware sales.

In June of 2014, one of the highlights of Apple's Worldwide Developers Conference (WWDC) was the announcement — a surprise to many attendees — of a new development language for iOS and OS X devices. Called Swift, it was presented as the language of the future for Apple's developers, but it was made very clear that it would co-operate with the existing basic development language — Objective-C.

This book gets you started developing with Swift. You'll quickly see how to create projects in Swift from the built-in templates that are part of the Xcode development tool. From there, you'll delve into features of the languages ranging from the basics to the advanced features that are unique to Swift as well as some features of Swift that may be familiar to you from other modern programming languages.

Swift and Objective-C are the languages most often used in building apps for iOS and OS X. Combined with the Cocoa (OS X) and Cocoa Touch (iOS) frameworks and Xcode, these languages allow you to develop just about anything you can dream of. It is hard to find an app that can't be written with these tools: OS X and iOS apps as well as other Apple products such as Pages, Keynote, and Numbers are developed using Xcode and the Cocoa frameworks. Most of the language work for these products is in Objective-C or Swift, although some sections are still in C++. Apps developed with these technologies are *native* apps.

If you don't want to go the native route, you can consider using other (non-Apple) frameworks. Three widely used frameworks are Titanium Appcelerator, PhoneGap, and HTML5, which is frequently used as a development tool without being a framework. In the world of Titanium Appcelerator, you typically write in JavaScript, whereas in PhoneGap you use Javascript, HTML, and CSS. HTML5, of course, is itself a language, which you can use in conjunction with JavaScript as well as other languages.

The advantages of using these non-native frameworks center around two features:

- With these frameworks, the development process may be faster than the native-app framework.
- Using these frameworks can help you develop cross-platform apps.

The biggest disadvantage is that non-native frameworks are third-party tools and as such aren't guaranteed to support new (or even all current) features of Apple's operating systems and hardware.

The cost of developing a native app for iOS, OS X, Android, or even Windows is likely to be significantly higher than that of using one of the tools listed here. Before making a decision, you may want to explore tools such as FileMaker (a wholly-owned subsidiary of Apple), which is designed for use by non-programmers. Originally a database application, FileMaker now has become a key tool for people who may never write a line of code in their life but are comfortable (and happy!) spending their time analyzing data and the processes that use it. Give FileMaker a look — particularly if your app idea is data-related.

In terms of cost and difficulty, FileMaker is likely to be your best choice; Titanium Appcelerator, PhoneGap, and HTML5 the second-best; and native app development with Swift or Objective-C (or both) the third. In terms of flexibility, however, this order would be reversed — the native apps are the most flexible, and tools like FileMaker are the least.

If your goal is to “get an app deployed by next Tuesday,” FileMaker may be your best bet (if it can be done at all). On the other hand, if your goal is to build something that will last or — perhaps more importantly — something that will develop your skills and expertise in this rapidly-growing mobile world, native apps with iOS and OS X are usually the way to go.

Getting the Developer Tools

The tools you need to develop with Swift for iOS or OS X are simple:

- **A Mac:** Any current Mac has enough processor power and memory to build apps. As of this writing, you must be running Mavericks (OS X 10.9) or Yosemite (OS X 10.10).
- **Xcode:** Xcode is the integrated development environment (IDE) for building OS X and iOS apps. You can download it for free from the Mac App Store.

These are the essentials: They’re really all you need. However, a few additional tools can be very helpful — even necessary for some developers — but if you don’t have them, don’t worry about it. There are some workarounds and compromises if you’re missing these tools. They include:

- **An iOS device (if you’re developing for the iOS platform):** It’s possible to develop an iOS app with no hands-on experience, but the results will show, particularly when you get reviews in the App Store along the lines of, “Hasn’t the developer ever tried to use an iPhone?” Writing an app based on an understanding of iOS devices gained from articles and advertising is very difficult. It’s much easier if you have access to the real thing. The one exception to this is when you’re working on only a component of a larger app, one that doesn’t involve the user interface.

- **Internet access:** If you only have intermittent Internet access, you can still develop, but with limitations. Certain tasks, such as getting your app into the App Store, require Internet access.
- **Plenty of disk space and a robust backup mechanism, such as Time Machine:** “Plenty of disk space” can mean a large internal disk, one or more external disks connected to your Mac, or cloud-based storage. The backup mechanism is important because you need to be confident that all of your project’s files are properly backed up and can be restored, if necessary.

WARNING!

The Apple developer discussion boards (Apple Developer Forums, located at `https://devforums.apple.com`) provide a valuable resource, as I describe in the following section. Every few months, a plea for help shows up on those boards that goes something like this: “How do I get my app into the App Store? I’ve written it and now I have to get it into the App Store.” (Such questions pop up in other places on the Internet as well.) Native apps need to be written in Swift or Objective-C, and they must abide by the rules of the App Store and the Mac App Store. Make certain you understand this before you start work. Writing an app in Java, HTML5, JavaScript, Objective-C without the Cocoa or Cocoa Touch framework, or even COBOL counts only as a personal experiment with a user interface; your app can’t be moved into the App Store. You may be able to distribute it as an app-like website, but not as a native app. Along those lines, the various commercial app development frameworks and services may or may not be useful to you.

Setting Up Your Mac

This section provides a quick overview of setting up your Mac for development in general and for Swift in particular. The basic steps are included here, and, as you’ll see, each one can open a variety of doors to additional steps and information.

Registering as a developer

The basic developer tools are free, and you can start work immediately, but Apple requires that you register as a developer at `developer.apple.com` before you can access many of the tools and features that you’ll need, particularly for testing. More and more of the developer website is available without logging in, but some of the key features require registration.

Not only do you need to register, but in some areas of `developer.apple.com`, you need to sign a non-disclosure agreement. You will be invited to join a developer program, which may entail an annual fee (see the upcoming section, "Choosing your program," for more on this), but if you're on a limited budget or just exploring, you can get quite a long way by saying "No, Thanks" to the offers. Rejecting offers like these and surveying the site is a good way to get your feet wet and get a good general understanding of the environment — although it does not let you submit your apps to the App Store.

TIP

The rules and policies of Apple's developer programs change from time to time and from country to country. Look at `developer.apple.com` for the latest definitive information. Be particularly careful of web postings you find that may be from the past. Always check the posting date from any articles you rely on — and that includes articles from Apple on `developer.apple.com`. You can usually find this date at the bottom of an article.

Preparing your credentials

For basic registration, you need to identify yourself to Apple. The standard way of doing this is to provide an Apple ID. Your Apple ID is not confidential, but your password is. Go to `appleid.apple.com` to register or to manage your Apple ID (including changing your password, if you want).

You may have more than one Apple ID (and you may need to have more than one, in some cases). Some developers have separate Apple IDs for personal use in addition to ones used for development. You may have a different Apple ID for developing apps than you have for developing iBooks — in fact, at the moment, this is a requirement.

TIP

For more on this, see the section, "Planning Your Environment," later in this chapter.

Choosing your program

Apple offers a number of different developer programs in which you can enroll. The simplest one is an individual program, which, at the moment, costs $99 a year, with separate programs for iOS and OS X. Thus, if you register for both, you'll pay $198 a year.

You can also register as a business. This registration allows you to build teams of individual developers and share code among yourselves as you develop apps.

There are also programs for educational institutions. Make certain that you have an Apple ID that you will use for your development, and then choose your program. If you are working for a business or are enrolled in a school, check to see if you are eligible to join a program there: It may save you some money.

Planning Your Environment

Your development environment is centered around a Mac with Xcode installed on it, but you can use several Macs for development. Each one should have Xcode on it (see further topics in this section for a discussion of multiple versions of Xcode). Because Xcode is free, it's easy to install it on a number of Macs. In fact, if you have access to a shared computer lab (perhaps at school), Xcode may be a component that is part of the lab.

There are no special settings to use Xcode with Swift, except to choose Swift (rather than Objective-C) from the pop-up menu when creating a new project or file. You also need to download the latest Swift documentation and SDK, but this is part of the standard installation process. If you are working with an old version of Xcode, you may be better off downloading and installing a new version of Xcode from scratch and proceeding from there. Your older Xcode projects should be able to be opened (and automatically converted if necessary) by the new Xcode.

Using source control

Source control is built into Xcode with both Git and Subversion. Because of its architecture, Git is more closely integrated into Xcode than Subversion. If you use one of these tools, you can easily store your source code in a repository. If you make a practice of committing changes to your source code repository on a regular basis (at the end of each work session perhaps, or when a significant milestone has been achieved), you'll know that you can download the project on demand.

Using source control means that no matter which Mac you find yourself using, as long as you have a network (or Internet) connection, you can check out the latest copy of the project (or a branch thereof), make your changes, and then check your files back in. And the next time you want to work on it, just check out the project on another computer and keep going.