

Making Everything Easier!™

Swift™

FOR DUMMIES®

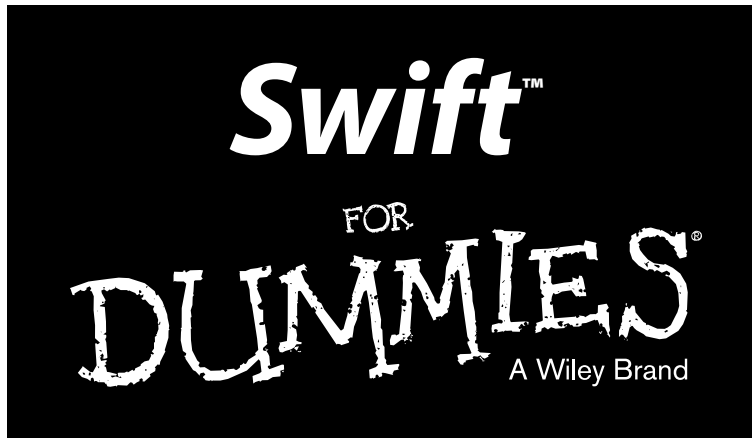
A Wiley Brand

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





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

Copyright © 2015 by John Wiley & Sons, Inc., Hoboken, New Jersey

Media and software compilation copyright © 2015 by John Wiley & Sons, Inc. All rights reserved.

Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the Publisher. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at www.wiley.com/go/permissions.

Trademarks: Wiley, For Dummies, the Dummies Man logo, Dummies.com, Making Everything Easier, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and may not be used without written permission. Swift is a trademark of Apple, Inc. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book. *Swift For Dummies®* is an independent publication and has not been authorized, sponsored, or otherwise approved by Apple, Inc.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002. For technical support, please visit www.wiley.com/techsupport.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at <http://booksupport.wiley.com>. For more information about Wiley products, visit 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

Introduction	1
Part I: Getting Started with Swift	5
Chapter 1: Setting Up an Xcode Swift Project	7
Chapter 2: Playing in the Playground.....	39
Chapter 3: Using the Xcode Editing Tools	47
Chapter 4: Creating a Swift App	63
Part II: Introducing Actions.....	105
Chapter 5: Operating on Data.....	107
Chapter 6: Using Swift Types.....	127
Chapter 7: Collecting Objects.....	153
Chapter 8: Controlling the Flow	175
Chapter 9: Functioning Successfully	191
Part III: Putting Expressions Together.....	217
Chapter 10: Expressing Yourself.....	219
Chapter 11: Declaring the Symbols	235
Chapter 12: Initializing and Deinitializing Data	243
Part IV: Using Components and Subcomponents	255
Chapter 13: Expanding Objects with Extensions.....	257
Chapter 14: Managing Access Control for Your Objects	267
Chapter 15: Building Classes, Structures, and Enumerations.....	273
Chapter 16: Using Properties, Variables, Outlets, and Actions	287
Chapter 17: Working with Enumerations to Consolidate Values.....	303
Chapter 18: Using Protocols to Provide Templates for Functionality	313
Chapter 19: Mixing Objective-C and Swift.....	327
Part V: The Part of Tens	335
Chapter 20: Ten Swift Features That Aren't in Objective-C.....	337
Chapter 21: Ten Swift Features That Are Not in C.....	341
Chapter 22: Ten Objective-C Features That Aren't in Swift.....	345
Index	349

Table of Contents

<i>Introduction</i>	1
About This Book	1
Conventions Used in This Book	2
Foolish Assumptions	2
Icons Used in This Book	3
Beyond the Book	4
Where to Go from Here	4
<i>Part I: Getting Started with Swift</i>	5
Chapter 1: Setting Up an Xcode Swift Project	7
Looking Ahead to the End	8
Working with Swift	11
Getting the Developer Tools	13
Setting Up Your Mac	14
Registering as a developer	14
Preparing your credentials	15
Choosing your program	15
Planning Your Environment	16
Using source control	16
Using GitHub	17
Organizing files	18
Preparing for environmental changes	18
Getting Started with Swift	20
Installing Xcode and setting preferences	20
Creating your project	28
Exploring Your Project	31
Touring your new project	31
Adding a map to the project's interface	35
Chapter 2: Playing in the Playground	39
Creating a Playground	39
Using a Playground	41
Testing a line of code	41
Checking syntax	42
Using the Timeline in the Playground	45

Chapter 3: Using the Xcode Editing Tools 47

Getting Started with Editing Tools	48
Completing Code with Code Completion	48
Using basic code completion	48
Using code completion in a project.....	50
Working with code completion.....	52
Using Fix-It to Correct Code	52
Folding and Unfolding Code	55
Using Code Snippets	56
Working with built-in code snippets	57
Creating your own code snippets.....	58

Chapter 4: Creating a Swift App 63

Double-Checking Your Environment.....	64
Creating the Project	65
Choosing the name.....	65
Understanding bundles.....	66
Reviewing a project's file structure.....	67
Working with workspaces.....	68
Testing the Template	70
Setting the Location for iOS Simulator	73
Adding devices to iOS Simulator	73
Managing the iOS Simulator location	76
Adding the Map to the Storyboard and Project	78
Testing the App.....	81
Adding Swift Code to Locatapp	82
Setting and confirming location settings in iOS Simulator and on devices.....	82
Making the app display the user's location (Part 1: Core Location)	86
Making the app display the user's location (Part 2: MapKit).....	92
Storing the user's location (Core Data)	94
Displaying the stored location.....	102
Testing the App with Location Data.....	103

Part II: Introducing Actions*..... 105*Chapter 5: Operating on Data 107**

Classifying Operators.....	108
Answering Syntax Questions with Playgrounds.....	109
Clearing the Way for Operators.....	111
Assigning Values with Assignment Operators	112
Counting On Arithmetic Operators for Math.....	113

Addition	114
Handling over- and underflow conditions	114
Subtraction	115
Multiplication	116
Division	117
Handling undefined results and errors	117
Using remainder division	120
Incrementing and Decrementing Numeric Values	122
Combining Operators	123
Comparing Values	124
Choosing and Checking Values with Logical Operators	125

Chapter 6: Using Swift Types 127

Understanding Types and Type Safety	128
Considering type safety	128
Exploring Swift types	130
Swift Standard Library Types	131
Strings	132
Arrays	132
Dictionaries	133
Numeric types	133
Protocols	133
Free functions	137
Specifying Variable or Expression Types with Type Annotations	138
Dealing with Tuples	142
Decomposing a tuple	142
Accessing tuple values with names	144
Accessing tuple values with index numbers	144
Skipping tuple values with underscores	144
Creating a tuple with types	145
Type safety with tuples	145
Working with Optional Types	146
Using optional types for values that may or may not exist	147
Using forced unwrapping with exclamation marks	148
Using optional binding to access an optional value	149
Using Generic Types	150
Using a generic function	150
Using a generic type	151

Chapter 7: Collecting Objects 153

Playing by Swift Collection Rules	153
Managing a collection of objects	155
Looking inside a Swift collection object	157
Mutability of collections	158
Switching mutability	159

Organizing Data Sequentially with Arrays.....	160
Comparing full and shorthand array syntax forms	160
Creating arrays.....	161
Working with arrays and their elements	162
Organizing Data Logically with Dictionaries	169
Comparing full and shorthand dictionary syntax forms	170
Creating dictionaries	170
Working with dictionaries and their elements.....	170
Fast enumeration over a dictionary	171
Reporting errors with a dictionary.....	171
Chapter 8: Controlling the Flow	175
Looping through Code	175
Using for loops	176
Using while loops.....	185
Using Conditions.....	186
Working with if statements.....	187
Working with switches	187
Transferring Control	189
Using Assertions	189
Chapter 9: Functioning Successfully	191
Setting the Stage for the Social Media Location App	192
Introducing Locatapp.....	192
Creating the Locatapp project	193
Modifying the template.....	196
Exploring the Functions in Locatapp	208
Understanding the Locatapp Architecture	209
Uncovering the Function Features	210
Basic function syntax — configureView().....	210
Overriding a function — viewDidLoad()	211
Calling the super function — viewDidLoad()	211
Adding a parameter to a function — controllerWillChangeContent(_____).....	212
Using multiple parameters in a function — controller(_____:didChangeSection:atIndex:forChangeType:)	212
Using external names for parameters — controller (_____:didChangeSection:atIndex:forChangeType:).....	213
Returning a value from a function — numberOfSectionsInTableView(_____).....	214
Adding Location Support.....	214

Part III: Putting Expressions Together..... 217**Chapter 10: Expressing Yourself 219**

Surveying the Types of Swift Expressions.....	219
Exploring optional type operators	220
Prefix expressions.....	222
Postfix expressions.....	223
Binary expressions	223
Primary expressions.....	225
Understanding Lazy Loading	229
Lazy loading with Objective-C.....	230
Lazy loading the easy way with Swift	232
Summarizing lazy loading	233

Chapter 11: Declaring the Symbols..... 235

Navigating through Symbols with the Symbol Navigator	235
Choosing types of displays.....	236
Choosing what to display	238
Preventing Disasters with Assertions.....	238
Patterns.....	239
Ranges.....	241

Chapter 12: Initializing and Deinitializing Data 243

Understanding Initialization.....	244
Performing Initialization	245
Setting default values for stored properties	246
Creating initializers for stored properties.....	248
Adding parameters to initializers	250
Understanding Deinitialization	254

Part IV: Using Components and Subcomponents..... 255**Chapter 13: Expanding Objects with Extensions 257**

Working with a Swift Extension	258
Using Swift Extensions with a Built-In Class.....	262

Chapter 14: Managing Access Control for Your Objects 267

Introducing Access Control Levels	268
Using Swift Access Control Terminology	269
Modules.....	269
Files.....	269
Entities	272



Chapter 15: Building Classes, Structures, and Enumerations 273

Exploring Classes, Structures, and Enumerations	273
Declaring a Simple Class	276
Exploring a Swift Class, Structure, or Enumeration File	278
Comments and copyrights	278
Import declarations	279
Class declaration	280
Structure declaration	281
Enumeration declaration	281
Body	283
UI actions and outlets (classes only)	283
Variables and properties	283
Functions and methods	284

Chapter 16: Using Properties, Variables, Outlets, and Actions 287

Understanding Properties and Variables	288
Encapsulating Data for Good Design and Maintainability	288
Understanding Properties and Variables in Locatapp	292
Types of Swift properties	296
Declaring Outlets and Actions	297

Chapter 17: Working with Enumerations to Consolidate Values . . . 303

Using Enumerations with Swift	303
Understanding Traditional C Structures and Enumerations	304
Exploring Swift Enumerations	305
Working with members of an enumeration	307
Working with a function inside an enumeration	308

**Chapter 18: Using Protocols to Provide Templates
for Functionality 313**

Understanding Protocols	313
Experimenting with Protocols	315
Declaring a protocol	315
Adopting and conforming a class, structure, or enumeration to a protocol	317
Exploring Protocols and a UITableViewController	321
Looking at delegation and protocols	321
Setting delegates in Interface Builder	322

Chapter 19: Mixing Objective-C and Swift 327

Comparing Frameworks in Objective-C and Swift	329
Identifying the key method	329
Comparing declarations	330

Calling an Objective-C Method in Objective-C within Swift to Set a Pin on the Map	331
Bridging between Objective-C and Swift	334
<i>Part V: The Part of Tens</i>	335
Chapter 20: Ten Swift Features That Aren't in Objective-C	337
Using Playgrounds to Explore Code and Syntax	338
Using Tuples	338
Using Ranges to Save Code	338
Taking Advantage of Strict Typing and Type Safety	339
Initializing Your Variables and Constants	339
Understanding Optional Types	339
Looking at Frameworks for Your Own Code	339
Including Annotations and Attributes in Declarations	340
Deinitializing Variables Where Necessary	340
Use Patterns to Simplify Your Code	340
Chapter 21: Ten Swift Features That Are Not in C	341
Strong Typing	341
Libraries Extend C	342
Switch Statements Fall through Cases in C	342
C Is an International Standard	343
Swift Is Tightly Linked to the Cocoa and Cocoa Touch Frameworks	343
Swift Includes Memory Management	343
Swift Is Designed to Function in a Multi-Threaded Environment	343
Types Can Be Created Easily in Swift	344
Swift Has Its Own IDE and Compiler	344
Types Can Be Classes, Structures, or Enumerations	344
Chapter 22: Ten Objective-C Features That Aren't in Swift	345
Saying Goodbye to Header (.h) Files	345
Saying Farewell to Dangling Pointers (Almost Always)	345
Forgetting About Uninitialized Variables and Properties	346
Exploiting a Common Superclass Like NSObject	346
Managing Type Casting	346
Preferring Closures to Blocks	346
Getting Rid of Legacy Memory Management	347
Replacing Property Decorators	347
Using Swift Style to Access Class Properties	347
Clarifying Swift Access Control	347
<i>Index</i>	349

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.



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



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



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.



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).



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.



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

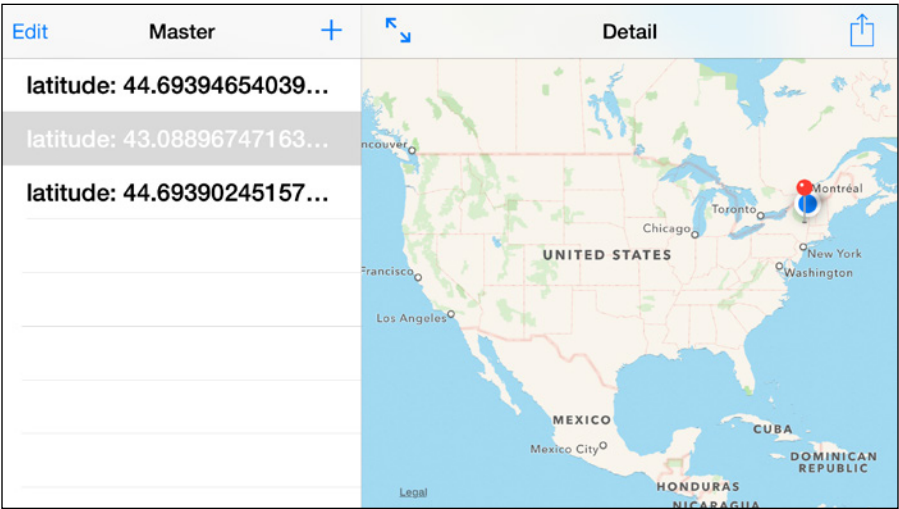
As you make progress in Swift, this book helps you build an app — a real, live app — based on one of the built-in Xcode templates. *Sure*, you’re probably thinking, *that’s just what I need — another “Hello World” app*.

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

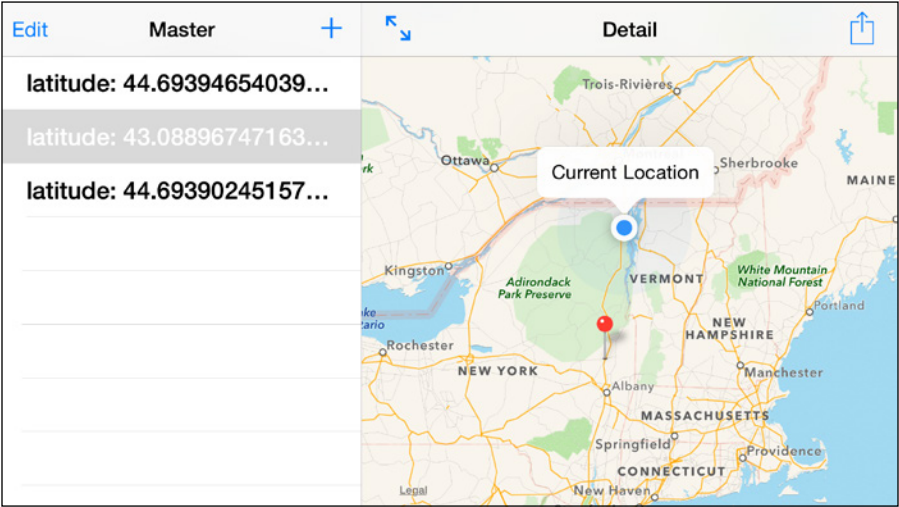
Figure 1-1:
Locatapp
finds your
location.



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.

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.'"

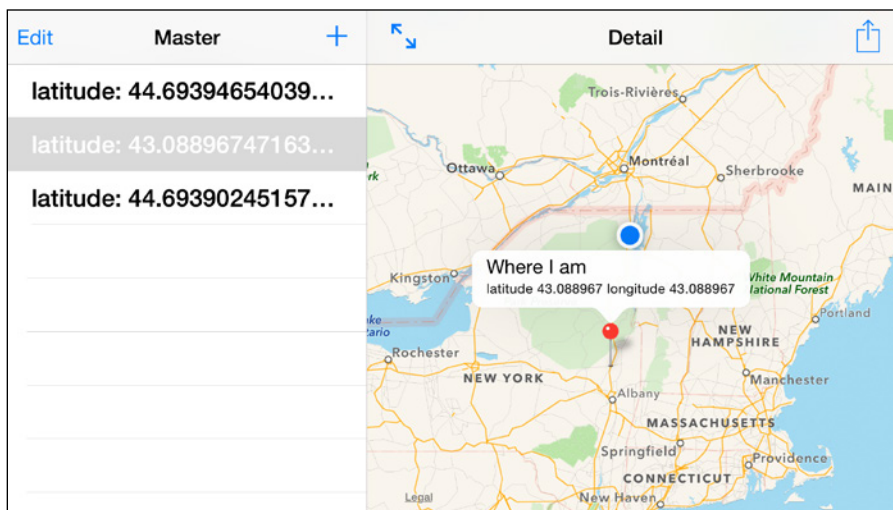


Figure 1-3:
Writing
your own
annotations.

Figure 1-4:
Implementing
an action
button.

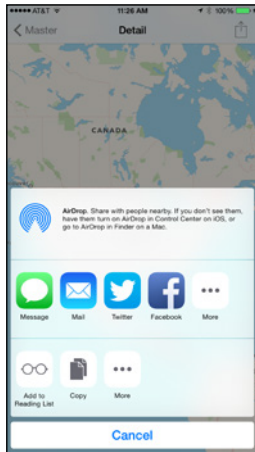
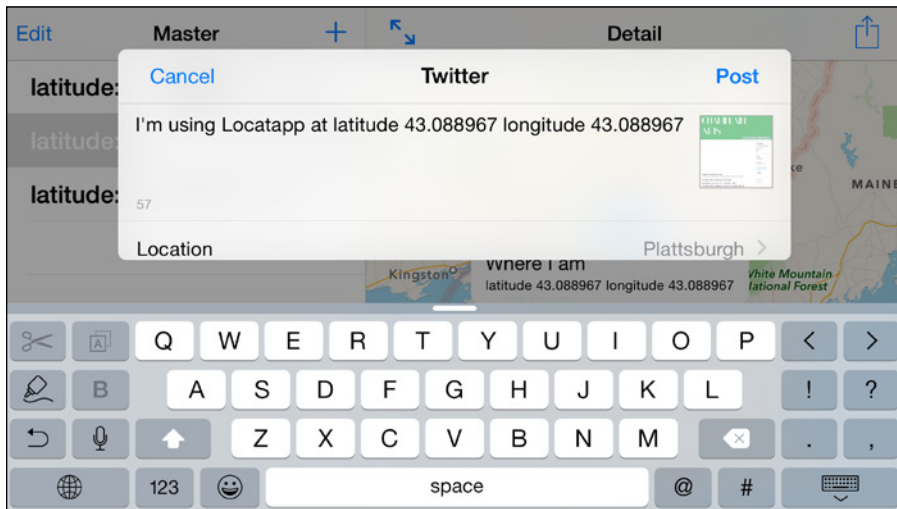


Figure 1-5:
Constructing
social media
messages
from your
code.



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.



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.



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.



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.