Beginning iPhone Development with Swift 3

Exploring the iOS SDK

—

Third Edition

—

Molly Maskrey

Apress®

Beginning iPhone Development with Swift 3

Exploring the iOS SDK

Third Edition

Molly Maskrey
Kim Topley
David Mark
Fredrik Olsson
Jeff Lamarche

Apress®

Beginning iPhone Development with Swift 3: Exploring the iOS SDK

Molly Maskrey
Parker, Colorado, USA

Kim Topley
New York, USA

David Mark
Arlington, Virginia, USA

Fredrik Olsson
Stockholm, Sweden

Jeff Lamarche
New Hartford, New York, USA

ISBN-13 (pbk): 978-1-4842-2222-5 ISBN-13 (electronic): 978-1-4842-2223-2
DOI 10.1007/978-1-4842-2223-2

Library of Congress Control Number: 2016959268

Managing Director: Welmoed Spahr
Lead Editor: Aaron Black
Technical Reviewer: Bruce Wade
Editorial Board: Steve Anglin, Pramila Balen, Louise Corrigan, James DeWolf, Jonathan Gennick, Robert Hutchinson, Celestin Suresh John, Nikhil Karkal, Michelle Lowman, James Markham, Susan McDermott, Matthew Moodie, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Gwenan Spearing
Coordinating Editor: Jessica Vakili
Copy Editor: Kim Burton-Weisman
Compositor: SPi Global
Indexer: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springer.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a Delaware corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at www.apress.com/bulk-sales.

Any source code or other supplementary materials referenced by the author in this text is available to readers at www.apress.com. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/.

Printed on acid-free paper

I feel incredibly fortunate to have been given the chance to write this revision. It was the very first version, years ago, that launched me toward developing for iOS. To that end, I want to thank the special people in my life that deserve much credit to bringing this to fruition.

First, to Chana, who was there with me for probably 95% of the writing, keeping my spirits up—especially on those days when I didn't think I could go on. She went from someone I met at "work" to one of the closest friends I've ever had. Every day I feel like she was a gift to raise my quality of life. I will be there for her whenever she needs me.

To my BFF Jessica (Goldi) for nearly two years now. She stood with me as my maid of honor and knows when I need her and is always there for me as I have been and will be for her. We drink together, dance together, crash company parties together, laugh together, and cry together.… She won't let me get away with anything, especially feeling sorry for myself. While on her seven-week South American wine journey, we communicated nearly every day. And though we were a continent apart, it brought us closer.

Ashley, I promised you I'd put you in here. Ash relates to me in a way unlike my other friends. A tech geek, we'll sit at her kitchen table drinking wine and talking about rebuilding computers. She's smart, pretty, and one of the sweetest and most fun people I know.

Finally, to my new bride, Jennifer. My friend, business partner, boss, "sometimes" dance partner, and the person beside me at night when I go to sleep and when I wake up. She read every word of this book and made sure it was great.

In my personal struggles over the preceding months, these friends kept me from falling into an abyss so deep I might never have returned. Writing can be a lonely thing and having a support system such as these beautiful, wonderful women are the only reason this endeavor was a success. A special friend told me just a couple weeks ago that some friends are only in your life for a season. I pray that these women are friends for a lifetime.

—MM, September 2016

Contents at a Glance

Contents

About the Author

Molly Maskrey started as an electrical engineer in her 20s at various large aerospace companies, including IBM Federal Systems, TRW (now Northrup Grumman), Loral Systems, Lockheed Martin, and Boeing. After successfully navigating the first dot-com boom, she realized that a break was in order and so she took several years off—moving to Maui and teaching windsurfing at the beautiful Kanaha Beach Park.

Never one to stay still, Molly moved to Denver, Colorado, and opened several businesses with her friend and partner Jennifer, including Global Tek Labs, an iOS development and accessory design services company that is now one of the leading consulting services for new designers looking to create smart attachments to Apple devices.

That same year, Molly, under her previous persona, published her first book through Apress. It was on how to create accessories for the iPhone operating system; to this day it remains the only major description of the process.

In 2014, Molly and Jennifer formed Quantitative Bioanalytics Laboratories—a wholly owned subsidiary of Global Tek—to bring high-resolution mobile sensor technology to physical therapy, elder balance and fall prevention, sports performance quantification, and instrumented gait analysis (IGA).

Molly recently began the Galvanize Data Science Immersive program in Denver, learning everything from Python, SQL, R, web scraping, big data analytics, Bayesian analysis, and many other skills in order to help with her next business adventure: a scalable, real-time analytics platform for IoT (the Internet of Things).

Molly lives in Parker, Colorado, with Jennifer and their two Labradors.

About the Technical Reviewer

Bruce Wade is a software engineer from British Columbia, Canada. He started software development when he was 16 years old by coding his first web site. He went on to study computer information systems at the DeVry Institute of Technology in Calgary, and then to further enhance his skills, he studied visual and game programming at The Art Institute of Vancouver. Over the years, he has worked for large corporations and several start-ups. His software experience has led him to utilize many different technologies, including C/C++, Python, Objective-C, Swift, Postgres, and JavaScript. In 2012, he started the company Warply Designed to focus on mobile 2D/3D and OS X development. Aside from hacking out new ideas, he enjoys spending time hiking with his Boxer, Rasco, working out, and exploring new adventures.

Acknowledgments

First, I want to acknowledge all my friends who gave me the support to persevere and go through with writing when it would have been so easy to just give up. To Sam, Brittany, Amanda, Kristie, Pyper, Peter, the Mikes, Kelly, and the whole team at Galvanize-Platte in Denver who I came to know and love, thank you.

To Children's Hospital Colorado and the Center for Gait and Movement Analysis who have been so generous with letting me be a part of understanding the significance of what they do for young adults with cerebral palsy and other gait disorders: the understanding I've gained drives me to focus efforts to help the many who truly need it.

To the clients and friends of Global Tek Labs who so generously allowed me to include some of their projects in this book for illustrative purposes. To the hundreds of people who have attended my talks over the past year and have given me ideas for what to include, such as John Haley who told me of his personal woes in understanding Auto Layout in Xcode. Those actual experiences helped drive the subject matter I chose to include.

Finally, I want to acknowledge all the authors before me that set the stage for my own little work to fit in to a much broader landscape.

CHAPTER 1

Getting to Know the iOS Landscape

Coding for Apple mobile devices provides developers a rewarding and lucrative career path where you might not only change people's lives with your app (see Figure 1-1), but you'll also have a great time being with bright, like-minded women and men such as yourself. Though you're bound to find some difficulty in learning the language, tools, and processes, these new associates will not only help you through the landscape of this new world, but will also challenge you to be your best and to stand out from the mediocre.

Figure 1-1. *One of the greatest feelings you can experience as an iOS developer is seeing other people using your creation out in the real world*

Electronic supplementary material The online version of this chapter (doi:10.1007/978-1-4842-2223-2_1) contains supplementary material, which is available to authorized users.

M. Maskrey et al., *Beginning iPhone Development with Swift 3*, DOI 10.1007/978-1-4842-2223-2_1

And for now, think of me as one of those friends along your journey of iOS discovery. I'm so proud to be able to help you by providing this initiation into the world of iOS development, whether it is for iPhone, iPod Touch or the iPad. iOS provides an exciting platform that has been seeing explosive growth since it first came out in 2007. The proliferation of mobile devices means that people are using software everywhere they go, whether it is a phone or a wearable, such as the Apple Watch. With the release of iOS 10, Xcode 8, Swift 3, and the latest version of the iOS software development kit (SDK), things continue to be exciting and generally become easier for the new developer.

About the Book

This book guides you down the path to creating your own iOS applications. I want to get you past the initial difficulties to help you understand the way that iOS applications work and how they are built.

As you work your way through this book, you will create a number of small applications, each designed to highlight specific iOS features and to show you how to control or interact with those features. If you combine the foundation you'll gain through this book with your own creativity and determination, and then add in the extensive and well-written documentation provided by Apple, you'll have everything you need to build your own professional iPhone and iPad applications.

■ **Note** Throughout most of this book, I tend to refer to the iPhone and iPad, as they are the devices that we'll most commonly use. This does not preclude the iPod Touch by any means; it is just a matter of convenience.

■ **Tip** The authors of the previous revisions of this book have set up a forum for this book. It's a great place to meet like-minded folks, get your questions answered, and even answer other people's questions. The forum is at `http://forum.learncocoa.org`. Be sure to check it out!

Things You'll Need

Before you can begin writing software for iOS, you'll need a few items. For starters, you'll need an Intel-based Macintosh running Yosemite (OS X 10.10), El Capitan (OS X 10.11), Sierra (macOS 10.12) or later. Any recent Intel-based Macintosh computer—laptop or desktop—should work just fine. Of course, as well as the hardware, you'll need the software. You can learn how to develop iOS applications and get the software tools that you'll need as long as you have an Apple ID; if you own an iPhone, iPad, or iPod, then you've almost certainly already have an Apple ID, but if you don't, then just visit `https://appleid.apple.com/account` to create one. Once you've done that, navigate to `https://developer.apple.com/develop`. That will bring you to a page similar to the one shown in Figure 1-2.

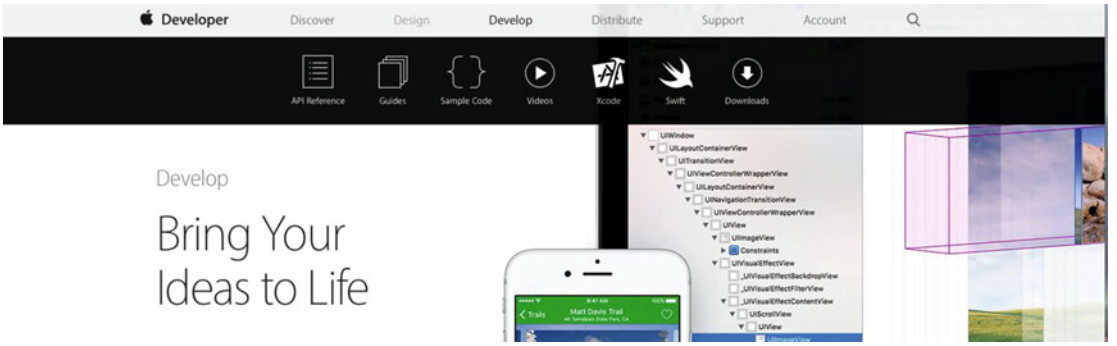

Figure 1-2. *Apple's Development Center resources site*

Click on **Downloads** across the top bar to go to the main resources page (see Figure 1-3) for the current production release and (if there is one) the current beta release of iOS. Here, you'll find links to a wealth of documentation, videos, sample code, and the like—all dedicated to teaching you the finer points of iOS application development. Be sure to scroll to the bottom of the page and check out the links to the Documentation and Videos sections of the web site. You'll also find a link to the Apple Developer Forums, where you can follow discussions on a wide variety of topics covering the whole iOS platform, as well as macOS, watchOS, and tvOS. To post to the forums, you'll need to be a registered Apple developer.

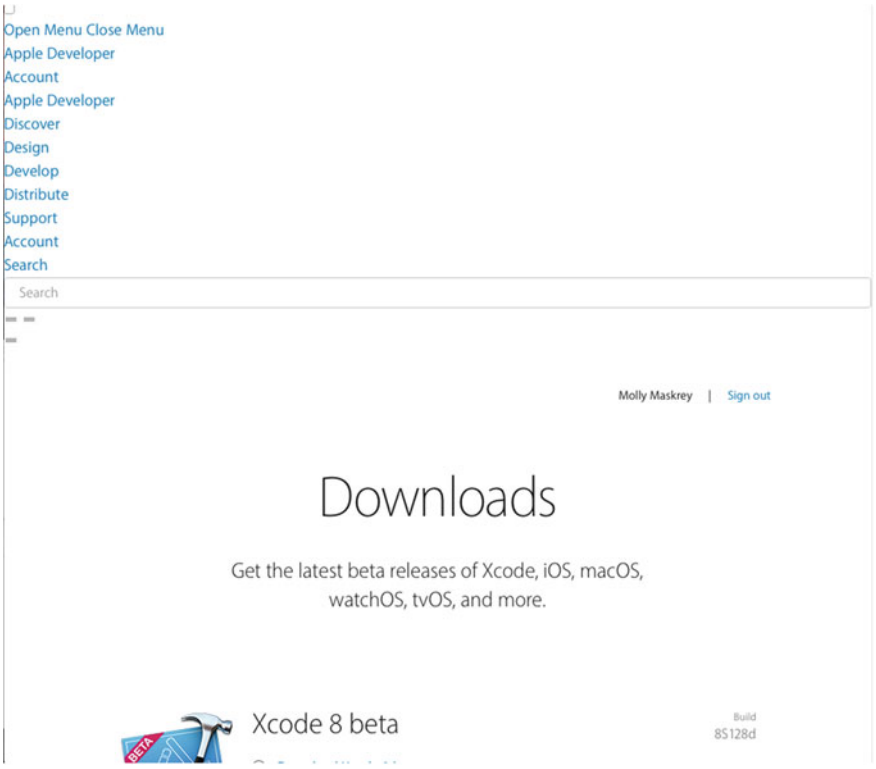

Figure 1-3. *You can download all the production and beta releases of the development tools from the Downloads page. You will need to sign in with your Apple ID*

■ **Note** At the 2016 developer conference, WWDC 2016, Apple changed the name of OS X back to the previously used *macOS* to become more in line with the other naming conventions used throughout the four major system platforms.

The most important tool you'll be using to develop iOS applications is called Xcode, Apple's integrated development environment (IDE). Xcode includes tools for creating and debugging source code, compiling applications, and performance tuning the applications you've written.

You can download the current beta release of Xcode by following the Xcode link from the developer Download page shown in Figure 1-3. If you prefer to use the latest production release, you'll find it in the Mac App Store, which you can access from your Mac's Apple menu.

SDK VERSIONS AND SOURCE CODE FOR THE EXAMPLES

As the versions of the SDK and Xcode evolve, the mechanism for downloading them changed over the past few years. Apple now publishes the current production version of Xcode and the iOS SDK on the Mac App Store, while simultaneously providing developers the ability to download preview versions of upcoming releases from its developer site. Bottom line: unless you really want to work with the most recent development tools and platform SDK, you usually want to download the latest released (non-beta) version of Xcode and the iOS SDK, so use the Mac App Store.

This book is written to work with the latest versions of Xcode and the SDK. In some places, new functions or methods are introduced with iOS 10 that are not available in earlier versions of the SDK.

Be sure to download the latest and greatest source code archive for examples from this book's page at `www.apress.com`. The code is updated as new versions of the SDK are released, so be sure to check the site periodically.

Your Options as a Developer

The free Xcode download includes a simulator that will allow you to build and run iPhone and iPad apps on your Mac, providing the perfect environment for learning how to program for iOS. However, the simulator does *not* support many hardware-dependent features, such as the accelerometer and camera. To test applications that use those features, you'll need an iPhone, iPod touch, or iPad. While much of your code can be tested using the iOS simulator, not all programs can be. And even those that can run on the simulator really need to be thoroughly tested on an actual device before you ever consider releasing your application to the public.

Previous versions of Xcode required you to register for the Apple Developer Program (which is not free) to install your applications on a real iPhone or other device. Fortunately, this has changed. Xcode 7 started allowing developers to test applications on real hardware, albeit with some limitations that we'll cover as we go along, without purchasing an Apple Developer Program membership. That means you can run most of the examples in this book on your iPhone or iPad for free! However, the free option does not give you the ability to distribute your applications on Apple's App Store. For those capabilities, you'll need to sign up for one of the other options, which aren't free:

- **The Standard program** costs $99/year. It provides a host of development tools and resources, technical support, distribution of your applications via Apple's iOS and Mac App Stores. Your membership lets you develop and distribute applications for iOS, watchOS, tvOS, and macOS.
- **The Enterprise program** costs $299/year. It is designed for companies developing proprietary, in-house iOS applications.

For more details on these programs, visit `https://developer.apple.com/programs` (see Figure 1-4). If you are an independent developer, you can definitely get away with just buying the standard program membership. You don't have to do that until you need to run an application that uses a feature such as iCloud that requires a paid membership, or you want to post a question to the Apple Developer Forums, or you are ready to deploy your application to the App Store.

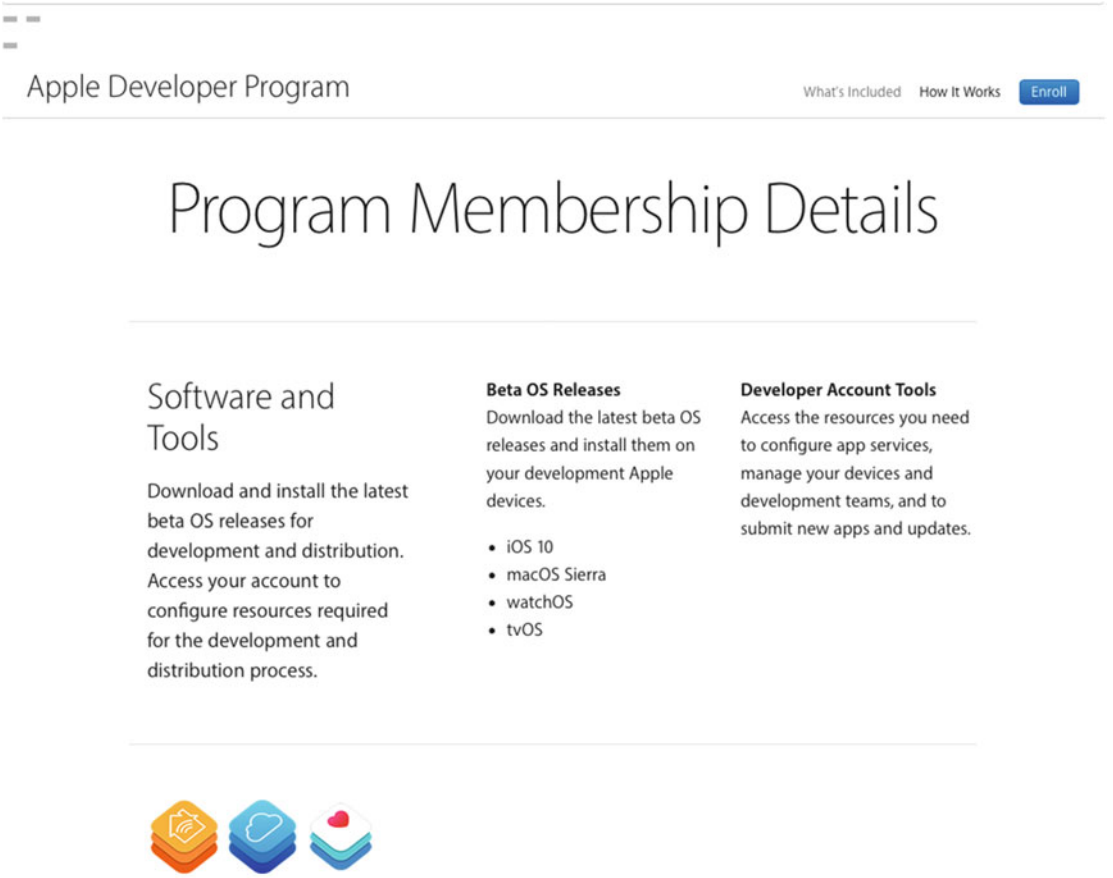

Figure 1-4. *Signing up for a paid membership gives you access to Beta and OS tools releases*

Because iOS supports an always-connected mobile device that uses other companies' wireless infrastructures, Apple has needed to place far more restrictions on iOS developers than it ever has on Mac developers (who are able—at the moment, anyway—to write and distribute programs with absolutely no oversight or approval from Apple). Even though the iPod touch and the Wi-Fi-only versions of the iPad don't use anyone else's infrastructure, they're still subject to these same restrictions.

Apple has not added restrictions to be mean, but rather as an attempt to minimize the chances of malicious or poorly written programs being distributed and degrading performance on the shared network. Developing for iOS may appear to present a lot of hoops to jump through, but Apple has expended quite an effort to make the process as painless as possible. And also consider that $99 is still much less expensive than buying, for example, any of the paid versions of Visual Studio, which is Microsoft's software development IDE.

Things You Should Know

In this book, I'm going to assume that you already have some programming knowledge in general and object-oriented programming in particular (you know what classes, objects, loops, and variables are, for example). But of course, I don't assume that you are already familiar with Swift. There's an Appendix at the end of the book that introduces you to both Swift and the Playground feature in Xcode that makes it easy to try out the features. If you'd like to learn more about Swift after reading the material in the Appendix, the best way to do so is to go directly to the source and read *The Swift Programming Language*, which is Apple's own guide and reference to the language. You can get it from the iBooks store or from the iOS developer site at `https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/index.html`.

You also need to be familiar with iOS itself, as a user. Just as you would with any platform for which you wanted to write an application, get to know the nuances and quirks of the iPhone, iPad, or iPod touch. Take the time to get familiar with the iOS interface and with the way Apple's iPhone and/or iPad applications look and feel.

Because the different terms can be a little confusing at first, Table 1-1 shows the relationships of IDE, APIs, and language to the platform operating system for which you are developing.

Table 1-1. *Platform, Tools, Language Relationships*

DEVELOP FOR	IDE	API	LANGUAGE
macOS	Xcode	Cocoa	Objective-C, Swift
iOS	Xcode	Cocoa Touch	Objective-C, Swift

Some Unique Aspects About Working in iOS

If you have never programmed for the Mac using Cocoa, you may find Cocoa Touch—the application framework you'll be using to write iOS applications—a little alien. It has some fundamental differences from other common application frameworks, such as those used when building .NET or Java applications. Don't worry too much if you feel a little lost at first. Just keep plugging away at the exercises and it will all start to fall into place after a while.

■ **Note** You'll see a lot of reference to "frameworks" in this book. Although the term is a little vague and used in a few different ways depending on the context, a framework is a collection of "stuff," which may include a library, or several libraries, scripts, UI elements, and anything else in a single collection. A framework's stuff is generally associated with some specific function such as location services using the CoreLocation framework.

If you have written programs using Cocoa, a lot of what you'll find in the iOS SDK will be familiar to you. A great many classes are unchanged from the versions that are used to develop for macOS. Even those that are different tend to follow the same basic principles and similar design patterns. However, several differences exist between Cocoa and Cocoa Touch.

Regardless of your background, you need to keep in mind some key differences between iOS development and desktop application development. These differences are discussed in the following sections.

iOS Supports One Single Application at a Time—Mostly

On iOS, it's usually the case that only one application can be active and displayed on the screen at any given time. Since iOS 4, applications have been able to run in the background after the user presses the Home button; but even that is limited to a narrow set of situations and you must code for it specifically (you'll see exactly how to do it in Chapter 15). In iOS 9, Apple added the ability for two applications to run in the foreground and share the screen, but for that, the user needs to have one of the more recent iPads. We'll talk about that feature, which Apple calls Multitasking, in Chapter 11.

When your application isn't active or running in the background, it doesn't receive any attention whatsoever from the CPU. iOS allows background processing, but making your apps play nicely in this situation will require some effort on your part.

There's Only a Single Window

Desktop and laptop operating systems allow many running programs to coexist, each with the ability to create and control multiple windows. However, unless you attach an external screen or use AirPlay, and your application is coded to handle more than one screen, iOS gives your application just one "window" to work with. All of your application's interaction with the user takes place inside this one window and its size is fixed at the size of the screen, unless your user has activated the Multitasking feature, in which case your application may have to give up some of the screen to another application.

For Security, Access to Device Resources Are Limited

Programs on a desktop or a laptop computer pretty much have access to everything that the user who launched it does. However, iOS seriously restricts which parts of the device your program can use.

You can read and write files only from the part of iOS's file system that was created for your application. This area is called your application's **sandbox**. Your sandbox is where your application will store documents, preferences, and every other kind of data it may need to retain.

Your application is also constrained in some other ways. You will not be able to access low-number network ports on iOS, for example, or do anything else that would typically require root or administrative access on a desktop computer.

Apps Need to Respond Quickly

Because of the way it is used, iOS needs to be snappy, and it expects the same of your application. When your program is launched, you need to get your application open, the preferences and data loaded, and the main view shown on the screen as fast as possible—in no more than a few seconds. Your app should have low latency.

■ **Note** By latency, we do not mean speed. Speed and latency are commonly interchanged but that is not really true. Latency refers to the time between an action is taken and a result happens. If the user presses the Home button, iOS goes home, and you must quickly save everything before iOS suspends your application in the background. If you take longer than five seconds to save and give up control, your application process will be killed, regardless of whether you finished saving. There is an API that allows your app to ask for additional time to work when it's about to go dark, but you've got to know how to use it. So, in general, you want to get things done quickly which might mean dumping and losing unnecessary information.

Limited Screen Size

The iPhone's screen is really nice. When introduced, it was the highest resolution screen available on a handheld consumer device, by far. But even today, the iPhone display isn't all that big, and as a result, you have a lot less room to work with than on modern computers. The screen was just 320 × 480 on the first few iPhone generations and it was later doubled in both directions to 640 × 960 with the introduction of the iPhone 4's Retina display. Today, the screen of the largest iPhone (the iPhone 6/6s Plus) measures 1080 × 1920 pixels. That sounds like a decent number of pixels, but keep in mind that these high-density displays (for which Apple uses the term **Retina**) are crammed into pretty small form factors, which has a big impact on the kinds of applications and interactivity you can offer on an iPhone and even an iPad. Table 1-2 lists the sizes of the screens of all of the current commonly available Apple devices that are supported by iOS 10 at the time of writing.

Table 1-2. *iOS Device Screen Sizes*

DEVICE	HARDWARE SIZE	SOFTWARE SIZE	SCALING
iPhone 5 and 5s	640 x 1136	320 x 568	2x
iPhone 6/6s	750 x 1334	375 x 667	2x
iPhone 6/6s Plus	1080 x 1920	414 x 736	3x
iPhone SE	640 x 1136	320 x 568	2x
iPad 2 and iPad mini	768 x 1024	768 x 1024	1x
iPad Air, iPad Air 2, iPad Retina, and iPad mini Retina	1536 x 2048	768 x 1024	2x
iPad Pro	2732 × 2048	1366 × 1024	2x

The hardware size is the actual physical size of the screen in pixels. However, when writing software, the size that really matters is the one in the Software Size column. As you can see, in most cases, the software size reflects only half that of the actual hardware. This situation came about when Apple introduced the first Retina device, which had twice as many pixels in each direction as its predecessor. If Apple had done nothing special, all existing applications would have been drawn at half-scale on the new Retina screen, which would have made them unusable. So Apple chose to internally scale everything that applications draw by a factor of 2, so that they would fill the new screen without any code changes. This internal scaling by a factor of 2 applies to all devices with a Retina display, apart from the iPhone 6/6s Plus, which has a higher-density screen that requires a scaling factor of 3. For the most part, though, you don't need to worry