Making Everything Easier!™

iOS App Development

FOR DUMMIES®

A Wiley Brand

Learn to:

- Download the iOS SDK and use Apple's developer tools
- Build a universal app for the iPad® and iPhone®
- Make the most of the latest iOS features in your app designs
- Provide a great user experience and make your app stand out in the crowd

Jesse Feiler
Author of iWork For Dummies®, 2nd Edition

Get More and Do More at Dummies.com®

Start with **FREE** Cheat Sheets

Cheat Sheets include

- Checklists
- Charts
- Common Instructions
- And Other Good Stuff!

To access the Cheat Sheet created specifically for this book, go to
www.dummies.com/cheatsheet/iosapplicationdevelopment

Get Smart at Dummies.com

Dummies.com makes your life easier with 1,000s of answers on everything from removing wallpaper to using the latest version of Windows.

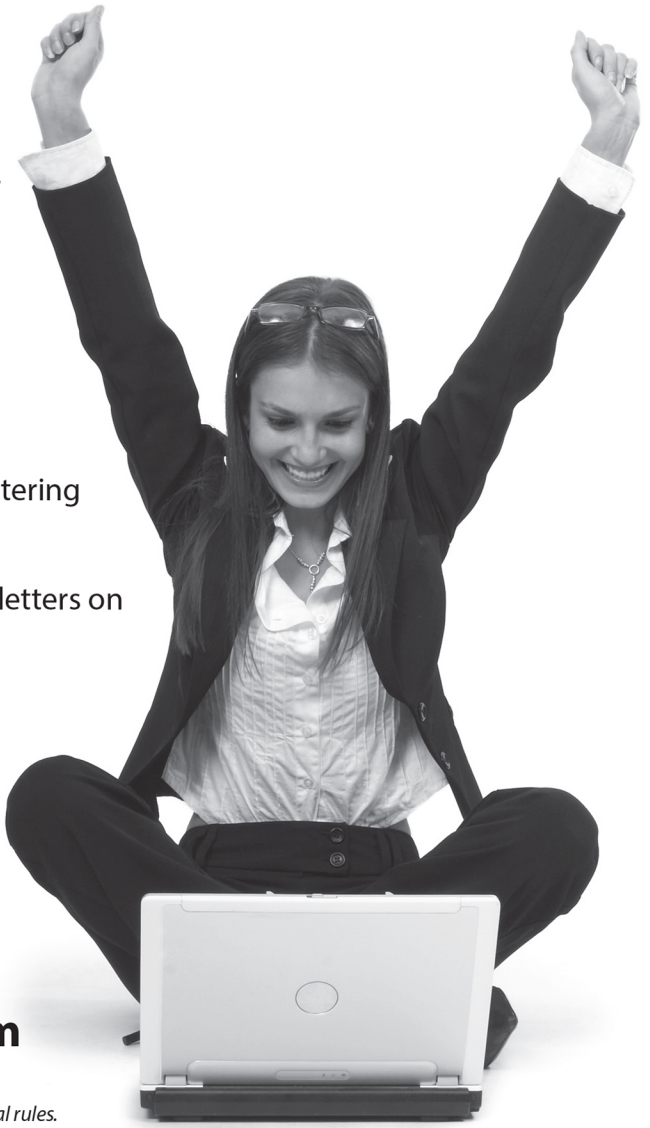
Check out our

- Videos
- Illustrated Articles
- Step-by-Step Instructions

Plus, each month you can win valuable prizes by entering our Dummies.com sweepstakes. *

Want a weekly dose of Dummies? Sign up for Newsletters on

- Digital Photography
- Microsoft Windows & Office
- Personal Finance & Investing
- Health & Wellness
- Computing, iPods & Cell Phones
- eBay
- Internet
- Food, Home & Garden



Find out "HOW" at Dummies.com

**Sweepstakes not currently available in all countries; visit Dummies.com for official rules.*

by Jesse Feiler

FOR
DUMMIES
A Wiley Brand

iOS App Development For Dummies®

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

Library of Congress Control Number: 2013957974

ISBN 978-1-118-87105-8 (pbk); ISBN 978-1-118-87107-2 (ebk); ISBN 978-1-118-87110-2 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Table of Contents

Introduction

iOS App Development For Dummies is a beginner's guide to developing iOS apps. And not only do you not need any iOS development experience to get started, but you also don't need any Mac development experience, either. I've written this book as though you are coming to iPhone and iPad app development as a blank slate, ready to be filled with useful information and new ways to do things. Well, almost a blank slate, anyway; see the upcoming "Foolish Assumptions" section for details on what you *do* need to know before using this book.

Because of the nature of the iPhone and iPad, you can create small, bite-sized apps that can be quite powerful. Also, because you can start small and create real applications that do something important for a user, it's relatively easy to transform yourself from an "I know nothing" person into a developer who, though not (yet) a superstar, can still crank out quite a respectable app.

But the iPhone and iPad can be home to some pretty fancy software as well — so I'll take you on a journey through building an industrial-strength app and show you the ropes for developing one on your own.

A Bit of History

It's 6:00 a.m. PST on January 9, 2007. A distressingly long line of nerds wrapped all the way around San Francisco's Moscone Center. Why? To hear Steve Jobs give his MacWorld Expo keynote address. It was nuts to get up so early on a cold morning, but Steve Jobs was rumored to be introducing an Apple phone.

No one knew whether an Apple phone would be any good, but perhaps Steve would show us magic — something that would revolutionize an industry. Perhaps it would be as cool and important as the iPod! A few hours later, Steve told the crowd that "Apple is going to reinvent the phone." Steve was never modest, but he was certainly correct — Apple completely blew away the phone industry that day. The damage was not yet visible to the current phone vendors (Palm, Motorola, Nokia, Sony, Ericsson, RIM, and Microsoft), but they were suddenly left back in the 20th century. The future had arrived.

The iPhone stands alone

The first iPhone shipped in late June 2007. It came with a bunch of Apple's native apps such as Contacts, Maps, Stocks, Safari, and so on. The problem was that only Apple could develop these native apps. The Apple developer "evangelists" told developers that we should be happy writing web apps. This did not make us happy — we wanted to write native Objective-C apps.

Steve Jobs and Apple eventually saw the light (in fact, some people believe that there was always the possibility of releasing tools to let developers write native apps, but getting the iPhone itself launched took a higher priority). Apple released a beta version of the iPhone Software Development Kit (SDK) in the spring of 2008, and it opened the App Store for business in July 2008. At this point, you could develop native apps — but only for the iPhone, because the iPad did not yet exist.

Enter the App Store

The App Store in July 2008 was a far cry from today's App Store. The numbers of apps and the numbers of downloads today are staggering. Search for "App Store" on Wikipedia to get the latest numbers. The billions of dollars that developers have earned directly from the App Store are fantastic.

But beyond these large numbers, there's something about the App Store that I didn't truly appreciate until my first app went on sale. On the first day, more than 20 copies were sold. (My initial advertising was a mass e-mail to friends.) This was very much a niche-of-a-niche product, but it continued to sell a few copies each week. I added a link to the app to my e-mail signature, and, when I saw a few copies had been sold in Great Britain, I assumed that some of my English cousins had pitched in (bless their hearts, as they say in the South).

But I don't have any relatives in Argentina. I'm pretty certain I don't know anyone in Malaysia. Okay, the first couple of sales in Canada might be explained by the fact that I live 20 miles south of the border. But why would someone in China be buying the app? I certainly hope the people in South Africa who have bought the app are using it productively.

Almost all of those people found the app by searching on the App Store. Apple provides a great deal of help and advice for you to put your app's best face forward on the App Store, and they provide tips and continually refine their search algorithms so that if you use good keywords, people can discover your apps. Apple wants to sell hardware, and they want users of their devices to discover apps that enhance their experiences with the devices.

The numbers of iOS devices are so vast that, with good keywords and a good app description, a niche-of-a-niche-of-a-niche app can find a home on the App

Store. You may write the next blockbuster app, but you also may write an app that gets modest results. The highly automated App Store provides the infrastructure to make it all possible.

You can count me among the people who think that the App Store itself may turn out to be a more significant achievement for Apple than the iPhone itself.

The iPad joins the party

Apple released the first iPad in April 2010. In some ways, the iPad was an even more remarkable achievement than the iPhone. The mobile phone existed before iPhone. iPad was the first time that a high-powered computing and communicating device that was truly mobile caught on.

Initially, the iPad ran the iPhone OS. That was a little hard for some people to understand, and before long, the operating system was renamed iOS. We're now at version 7 of iOS. In addition to the renaming, there has been some restructuring of the developer tools and environment so that things fit together very well.

When I look at developer features such as universal apps (they can run on both iPhone and iPad with minimal code changes), support for in-app purchases, and iBeacon integration, I see a full-featured environment that matches and even surpasses some of the most sophisticated development environments I've worked with.

Even though there are many more features today than there were back in 2008, iOS development today is easier than it was a few years ago. The developer tools have matured, and the frameworks themselves have been tweaked with tools such as auto layout that make the placement of interface elements on a screen automatic as devices are rotated and as new screen sizes appear on the devices.

The Plan for This Book

You will build this book's RoadTrip app using the following steps:

1. Create the initial storyboards for both the iPad and iPhone versions, starting with Xcode's Master-Detail project template.

   The template's iPad storyboard is based on using UIKit's `UISplitView Controller`, which uses the same custom `MasterViewController` and custom `DetailViewController` used in the iPhone version. The Master View controller will appear on the left when the iPad is held in landscape orientation, whereas the Detail View controller appears on the right.

2. Build and test the iPad version in the iPad simulator. You should see a Table view embedded in a Navigation controller in the Master view.

   The template's initial iPhone storyboard design begins with a custom `MasterViewController` (a Table view) embedded in a Navigation view. Selecting an item in the Table view displays data managed by a custom `DetailViewController`.

3. Build and test the iPhone version in the iPhone simulator. It should also work perfectly because you haven't had a chance to make any mistakes yet.
4. Add a `TestDriveController` to the iPhone storyboard. Build and test. Add it to the iPad storyboard. Build and test.
5. Add animation and sound to the Test Drive controller. Build and test both the iPhone and iPad versions.
6. Add additional features to each version until done.

iOS and Xcode Game Changers

With iOS 7 and — more importantly — with Xcode 5 (and later versions), the nuts and bolts of iOS app development have changed dramatically. Xcode 5 has added much more functionality to the integrated development environment (IDE) you use to develop iOS applications, especially when it comes to writing syntactically correct (and bug-free) code that's better able to manage memory. The latest versions bring much simpler integration with the App Store as well as newly designed performance-monitoring tools. Of course, the rub is getting the hang of Xcode 5. That's where this book comes in. I carefully take you through Xcode 5, pointing out its features and describing how to best use them. When you set this book aside, you'll have a great understanding of how to take advantage of all those features that will make your life easier.

You find out how to develop a single app that includes features that readers and students of earlier editions have been asking for — including more animation and sound — as well as an infrastructure that people can use to develop more robust applications. The resulting example is an app called RoadTrip, which can send you on your way to developing apps that you can be proud of and that other people will want to have.

This new edition is based on iOS 7 *and* Xcode 5. If you want to find out how to develop applications, the tools discussed in this book are the tools you absolutely need to use to do it the right way.

About This Book

iOS App Development For Dummies distills the hundreds (or even thousands) of pages of Apple documentation (not to mention my own development experience and that of many colleagues and friends) into only what's necessary to start you developing real applications. But this is no recipe book that leaves it up to you to put it all together. Rather, it takes you through the frameworks (the code supplied in the SDK) and iOS architecture in a way that gives you a solid foundation in how applications really work, and also acts as a road map to expand your knowledge as you need to.

I assume that you're in this for the long haul and that you want to master the whole app-development ball of wax. I use real-world applications to show the concepts and give you the background on how things actually work on iOS.

For many people, their first excursion into programming has been the classic Hello World C program. Depending on how you space it, it can be written in anywhere from one to three lines of code. For a long time now, I've thought that this program has long since outlived its usefulness. We just don't write code like that today (and many of us never did). As you'll see in this book, the development process involves typing code, but it also involves drawing an interface with your mouse, choosing options in check boxes to configure your app, and a whole bunch of other activities that are as far as you can get from typing a few lines of code into a blank document.

It's a new world for developers, and I think its excitement and opportunities have barely started.

Conventions Used in This Book

This book guides you through the process of building iOS apps. Throughout the book, you use the classes provided by Apple's iOS frameworks (and create new classes of your own, of course). You code them using the Objective-C programming language.

Code examples in this book appear in a monospaced font so that they stand out a bit better. That means that the code you see will look like this:

```
#import <UIKit/ UIKit.h>
```

Objective-C is based on C, which *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 Objective-C naming conventions — for example, class names always start with a capital letter, and the names of methods and instance variables always start with a lowercase letter.

Note that all URLs in this book appear in a monospaced font as well, like this:

```
www.northcountryconsulting.com
```

When I ask you to add code to a program, it will be in bold like this:

```
#import <UIKit/ UIKit.h>
```

You'll notice — starting around Chapter 8 — that you will sometimes be asked to delete some of the code you have in place for your project to make room for some new stuff. When that happens, I comment out the code to make things really clear. I refer to code I want you to delete as commented out, bold, underline, and italic code because said code will show up as commented out, bold, underlined, and italic. Simple enough, as shown in the following example:

```
// Delete this
```

If you're ever uncertain about anything in the code, you can always look at the source code on the companion website. (More on that in the section "Beyond the Book," later in this Introduction.)

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 I say interesting?), but it isn't essential to understanding iOS app development. 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.

This icon indicates how to use an important part of Xcode functionality. This helps you wade through Xcode's complexity and focus on how to get specific things done.

Foolish Assumptions

To begin programming your iOS applications, you need an Intel-based Mac with the latest or next-to-latest version of OS X on it. (No, you can't develop iOS applications directly on the iPhone or iPad.) You also need to download the iOS Software Development Kit (SDK) — which is free. And, oh yeah, you need an iPhone and/or iPad. You won't start running your app on it right away — you'll use the iOS Simulator that Apple provides with the iOS SDK during the initial stages of development — but at some point, you'll want to test your app on a real, live iOS device.

This book assumes that you have some programming knowledge and that you have at least a passing acquaintance with object-oriented programming, using some variant of the C language (such as C++, C#, or maybe even Objective-C). In case you don't, I'll point out some resources that can help you get up to speed. The app example in this book is based on the frameworks that come with the SDK; the code is pretty simple (usually) and straightforward. (I don't use this book as a platform to dazzle you with fancy coding techniques.)

I also assume that you're familiar with the iPhone and iPad themselves and that you've at least explored Apple's included applications to get a good working sense of an iOS app's look and feel. It might also help to browse the App Store to see the kinds of applications available there and maybe even download a few free ones (as if I could stop you).

How This Book Is Organized

iOS App Development For Dummies has six main parts, which are described in the following sections.

Part I: Getting Started

Part I introduces you to the iOS world. You find out what makes a great iOS app and see how an iOS app is structured. In Chapter 2, I give an overview of how Xcode 5 works that gets you up to speed on all its features; you can use this chapter as a reference and return to it as needed. You also create your Xcode project in this part — a universal app that can run equally well on an iPad or iPhone — and I take you on a guided tour of what makes up the Xcode project that will become your home away from home.

Part II: Building RoadTrip

In this part of the book, you find out how to create the kind of user interface that will capture someone's imagination. I explain the Interface Builder editor, which is much more than your run-of-the-mill program for building graphical user interfaces. You also discover storyboards, which are the icing on the Interface Builder cake that let you lay out the entire user experience and app flow — saving you a lot of coding, to boot.

You'll also take a brief tour of the RoadTrip app, the app that you build in this book. I show you not only what the app can do but also how it uses the frameworks and SDK to do that.

I also explain how the main components of an iOS app go together. I describe how the iOS applications work from a viewpoint of classes and design patterns, as well as show how the app works at runtime. I spend some time on three very important ideas: how to extend the framework classes to do what you want them to, how to manage memory, and how to take advantage of declared properties. I also explain how everything works together at runtime, which should give you a real feel for how an iOS app works.

Parts I and II give you the fundamental background that you need to develop iOS applications.

Part III: Getting Your Feet Wet: Basic Functionality

Now that you have the foundation in place, Part III starts you on the process of having your app actually *do* something. You start off by determining whether a network is available to support the app functionality that requires Internet access. You find out how to customize the appearance of the controls provided

by the framework to make your app a thing of beauty. You finish off by adding animation and sound just to get going. You also see how to connect the elements on your storyboard to your app code to make them do things — such as have a '59 pink Cadillac Eldorado Biarritz convertible drive up and down the screen.

Part IV: The Model and the App Structure

Now you begin to get down to the real work. You find out about the iPad's popovers and Split View controllers, and you also add navigation to the app. Along the way, I really get into showing you how to account for the differences between an iPad and an iPhone, and make sure that the app can run flawlessly on whatever device the user has handy. You also add the app model, which provides both the data and the logic you need to create an app that delivers real value to the user. You then finish the storyboard so that you can see your basic application flow. To wrap it all up, I show you how to package your app with a custom icon and prepare it for the App Store.

Part V: Adding the App Content

Now that you have the application foundation and the user experience architecture in place, Part V takes you into the world of applications that contain major functionality. I show you how to display the weather using a web page right off the Internet, how to allow the user to page through local events as if he were reading a book, how to display a map of where the user is going and where he is right now, how to find a location that he has always wanted to visit and display it on a map, and even how to change where he is going (limited in the RoadTrip app to New York and San Francisco, but it's incredibly easy to add other destinations). I don't go slogging through every detail of every detail, but I demonstrate almost all the technology you need to master if you intend to create a compelling app like this on your own.

Part VI: The Part of Tens

Part VI consists of some tips to help you avoid having to discover everything the hard way. It talks about approaching app development in an "adult" way right from the beginning (without taking the fun out of it). I also revisit the app and explain what else you would need to do to make this app a commercial and critical success.

Beyond the Book

This book has additional content you can access online:

- **Sample code:** Sample code for each chapter can be found online at www.dummies.com/extras/iosappdevelopment. Note that the posted code represents the code as it is at the end of the chapter. If you want to download code to follow along with as you read, download the code for the previous chapter.
- **Cheat Sheet:** This book's Cheat Sheet can be found online at www.dummies.com/cheatsheet/iosappdevelopment. See the Cheat Sheet for more on expanding your app with subclassing, target action, and delegation.
- **Dummies.com online articles:** Companion articles to this book's content can be found online at www.dummies.com/extras/iosappdevelopment. The topics range from tips on building an interface, using frameworks in iOS app development, and ten ways to make your app-developing life easier, among others.
- **Updates:** If this book has any Updates after printing, they will be posted to www.dummies.com/extras/iosappdevelopment.

Where to Go from Here

If you're starting from the beginning, I would suggest you start with Chapter 1. However, if you are brushing up your skills, feel free to jump into a chapter that is particularly relevant to your question. Chapters such as Chapter 10, "Adding Animation and Sound to Your App" may be useful to you on their own. The chapters in Part V are relatively independent of one another.

Part I
Getting Started

Visit www.dummies.com for more great content online.

In this part . . .

- Figuring out what makes a great iOS app
- Getting to know the SDK
- Sorting out the parts of an Xcode app

Chapter 1

What Makes a Great iOS App

In This Chapter

- Figuring out what makes an insanely great iOS app
- Discovering the iOS features that can inspire you
- Understanding Apple's expectations for iOS apps
- Making a plan for developing iOS software

July 10, 2008.

That was the day the App Store opened. The next day, Apple launched the iPhone 3G, which came with the brand-new iPhone OS 2.0.1. Owners could choose from the 500 apps in the App Store to expand their phone's power and features. You can find the latest numbers and versions in the App Store article on Wikipedia, which is updated regularly. I think it's fair to say that no one in 2008 envisioned the world of apps that we have today.

This is a world that some people have dreamed of from the dawn of the computer age in the early 1950s. It's a world of computers that are highly portable, that boast terrific connectivity without wires and cables, and that do things that people find useful, such as providing entertainment, education, practical support, and information. Part of that dream was a world in which the ability to program computers is accessible to the largest nations and corporations on almost equal terms as it is to the individual developer or hobbyist.

April 3, 2010.

That was the day the first iPad was shipped. Its operating system was iPhone OS 3.2. iOS 4 was released a few months later, and, today, iOS and its development tools have been substantially rearchitected to make both the user experience and the development process easier.

Today.

Because of the App store and the new development tools, this is a great time to start developing for iOS. Welcome!

Figuring Out What Makes a Great iOS App

A great iOS app can be described simply: It helps people do something that they want to do; it does it well; it does it when and where people want to do it; and it disappears. Because you can leverage the power of the App Store, your app can be successful globally. Yes, that may mean millions of users for your app, but it also may mean that you can find the 100 widely scattered around the world for whom your app may become a necessity.

For the most part, choosing your app's topic and market is beyond the scope of this book, but you can find many articles on the Internet and in books and magazines. You can even study the topic in colleges.

What this book does focus on is the rest of that description — building an app that works well, works when and where people want to use it, and has a user interface that helps people use the app but does not draw attention to itself. If people are thinking about your app when they should be thinking about a plot, a high score, or a trip to a store or another country, your app isn't great.

Making your app work well

The nuts and bolts of making your app work well are described in this book. At the most basic level, they are technical and organizational, but the first step is understanding what your app is going to be and do. Before you write your first line of code, think through the app. Who will the audience be? What will the app look like?

For large app development projects as well as small ones, sketching out a wireframe sequence of your app's screens is a good idea. You may think it's unnecessary in a one-person project, but it may even be more essential there. Show your sketches to friends who will be honest with you. You can search for "ios wireframe" on your favorite search engine to get recommendations for tools to help develop wireframes.

If your app delivers content, make certain that you have the content. If you have expertise in a specific area, that can serve as the basis for an app. If you have access to experts in other areas, see if they will advise you. If you go to the App Store and look at the reviews for apps, you'll see that people quickly provide low ratings for apps that don't work or that don't provide reliable data.

Making your app work well is easier than ever before with Xcode 5. Powerful debugging tools are built in so that you can even watch your app's performance on real-time gauges. Compared to earlier versions of iOS, iOS 7 has a host of improvements for users and simplifications for developers. In my opinion, the milestone was iOS 5. Although significant changes occurred in the later releases, iOS 5 was the first to provide the concept of universal apps where you could write a single code base for both iPhone and iPad. That entailed making some changes to the APIs, but we're over that hump and proceeding full speed ahead.

Handling networking, social media, and location

Networking, social media, and location can make your app great. There certainly are many successful and even great apps that don't use them, but they add additional layers of greatness to your app. Networking means that you can access Internet resources directly from your app. You can display web pages within your app, and you can also access data that you display in your app's interface. The tools to do this are available to you in the APIs.

Today, social media integration scarcely needs promotion: It has become part and parcel of our daily lives. Allowing users to promote your app on Facebook or Twitter with a simple tap is a no-brainer for many developers. iOS lets users enter their sign-in information whenever those taps occur.

Location awareness has opened a wide range of opportunities for apps. The most obvious opportunities involve integration with Maps, but "near me" functionality intrigues developers and users. With iOS 7, developers now have two sets of location tools to use. For traditional mapping, a set of tools uses GPS and cell tower locations to locate the device. Now, iBeacon adds tools to handle low-energy Bluetooth beacons over much shorter distances, such as individual paintings in a museum or specific shelves in a store.

Designing a powerful and intuitive interface that disappears

Designing a disappearing interface is one of the most challenging aspects of app design, and you'll find tips to achieve this throughout the book. A *disappearing interface* is one that works (or as many people say, "just works") without people having to think about it.

When someone looks at your app's interface and notices the interface, you're on the wrong track. What you want to achieve is a situation where someone looks at the screen and immediately sees how to get a weather report, the current temperature, or the prediction for tomorrow's weather in New York City. Users should not notice the interface. Instead, they should notice what the interface can *do*.

Using the iOS Platform to the Fullest

Okay, enough talk about the user experience. Just what exactly is the iOS platform, and what are its features?

Exploiting advantages of the system

One of the keys to creating a great app is taking advantage of what the device offers. In the case of a new platform with new possibilities, exploiting advantages is especially important. The combination of hardware and system software opens up design advantages that depart from the typical design approach for desktop and laptop apps. For example:

- **Multifinger gestures:** Apps respond to multifinger gestures, not mouse clicks. If you design an app that simply uses a single finger tap as if it were a mouse click, you may be missing an opportunity to design a better user experience.
- **Movement and orientation:** iOS devices have a variety of sensors that collect movement and orientation data. The new M7 chip in iPhone 5S, iPad Air, and iPad Mini (second generation) collects sensor information from the integrated accelerometers, gyroscopes, and compasses. This enhances existing location services and takes some of the workload off the main chip.
- **Split views and unique keyboards:** You can use a split view on an iPad to display more than one view onscreen at a time. Both iPad and iPhone provide a special keyboard unique to a task, such as the numbers-and-formulas keyboard that appears in the Numbers app.
- **Internet access:** With quick and easy access, your app doesn't need to store lots of data — all it really needs to do is jump on the Internet and grab what it needs from there. However, to be truly useful, your app needs to be ready to function when the Internet is unavailable to it.