iPhone, iPad and Apple Watch Made Easy

Includes Apple Watch and HealthKit

Swift 2 for Absolute Beginners

SECOND EDITION

Gary Bennett | Brad Lees

Apress®

Swift 2 for Absolute Beginners

Gary Bennett
Brad Lees

Apress®

Swift 2 for Absolute Beginners

ISBN-13 (pbk): 978-1-4842-1489-3

ISBN-13 (electronic): 978-1-4842-1488-6

Managing Director: Welmoed Spahr
Lead Editor: Michelle Lowman
Development Editor: Douglas Pundick
Technical Reviewer: Stefan Kaczmarek
Editorial Board: Steve Anglin, Louise Corrigan, James DeWolf, Jonathan Gennick, Robert Hutchinson, Michelle Lowman, James Markham, Susan McDermott, Matthew Moodie, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Gwenan Spearing, Steve Weiss
Coordinating Editors: Kevin Walter and Mark Powers
Copy Editor: Kezia Endsley
Compositor: SPi Global
Indexer: SPi Global
Artist: 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.springeronline.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 material referenced by the author in this text is available to readers at www.apress.com/9781484214893 or http://forum.xcelme.com. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/. Readers can also access source code at SpringerLink in the Supplementary Material section for each chapter.

Gary would like to dedicate this book to wife Stefanie and children, Michael, Danielle, Michelle, and Emily, for always supporing him.

Brad would like to dedicate this book to his wife Natalie, for always supporting him. He couldn't do it without her.

Contents at a Glance

Contents

About the Authors

Gary Bennett is president of xcelMe.com, which provides iOS programming courses online. By day, Gary develops iOS apps professionally, and by night, he teaches iOS programming. For more than six years, Gary has taught thousands of students how to develop iPhone/iPad apps and has several popular apps in the iTunes App Store. Gary has a bachelor's degree in computer science and has worked for 25 years in the technology and defense industries. He served 10 years in the U.S. Navy as a nuclear engineer aboard two nuclear submarines. After leaving the Navy, Gary worked for several companies as a software developer, CIO, and president. As CIO, he helped take VistaCare public in 2002. Gary also coauthored two editions of *Objective-C for Absolute Beginners* and *iPhone Cool Projects* for Apress. He lives in Scottsdale, Arizona, with his wife Stefanie and their four children.

Brad Lees has more than 16 years of experience in application development and server management. He has specialized in creating and initiating software programs in financial institutions, credit card processing, point-of-sale systems, and real estate development.

His professional career highlights have been lead iOS developer at Apriva, owner of Innovativeware, product development manager for Smarsh, and vice president of application development for iNation. Brad also coauthored two editions of *Objective-C for Absolute Beginners*.

A graduate of Arizona State University, Brad resides in Phoenix with his wife Natalie with their five children.

About the Technical Reviewer

Stefan Kaczmarek has more than 15 years of software development experience specializing in mobile applications, large-scale software systems, project management, network protocols, encryption algorithms, and audio/video codecs. As chief software architect and cofounder of SKJM, LLC, Stefan developed a number of successful mobile applications including iCam (which has been featured on *CNN*, *Good Morning America*, and *The Today Show*, and which was chosen by Apple to be featured in the "Dog Lover" iPhone 3GS television commercial) and iSpy Cameras (which held the #1 Paid iPhone App ranking in a number of countries around the world including the United Kingdom, Ireland, Italy, Sweden, and South Korea). Stefan resides in Phoenix, Arizona with his wife Veronica and their two children.

Acknowledgments

We would like to thank Apress for all their help in making this book possible. Specifically, we would like to thank Kevin Walter, our coordinating editor, and Michelle Lowman, our acquisitions editor, for helping us stay focused and overcoming many obstacles. Without Kevin and Michelle, this book would not have been possible.

Special thanks to Douglas Pundick, our development editor, for all his suggestions during the editorial review process to help make this a great book. Thanks to Kezia Endsley, the copy editor, who made the book look great.

Introduction

Over the past three years, we've heard the following countless times:

- "I've never programmed before, but I have a great idea for an iPhone/iPad app."
- "Can I really learn to program the iPhone or iPad?"

To the latter we answer, "Yes, but you have to believe you can." Only you are going to tell yourself you can't do it.

For the Newbie

This book assumes you have never programmed before. The book is also written for someone who may have programmed before but never using object-oriented programming (OOP) languages. There are several Swift books out there, but all of these books assume you have programmed before and know OOP and computer logic. We wanted to write a book that takes readers from knowing little or nothing about computer programming and logic to being able to program in Swift. After all, Swift is a native programming language for the iPhone, iPad, and Mac.

Over the past six years, we have taught thousands of students at `xcelMe.com` to be iPhone/iPad (iOS) developers. Many of our students have developed some of the most successful iOS apps in their category in the iTunes App Store. We have incorporated what we have learned in our first two courses—Introduction to Object-Oriented Programming and Logic and Swift for iPhone/iPad Developers—into this book.

For the More Experienced

Many developers who programmed years ago or programmed in a non-OOP language need a background in OOP and logic before they dive into Swift. This book is for you. We gently walk you through OOP and how it is used in iOS development to help make you a successful iOS developer.

How This Book Is Organized

You'll notice that we are all about successes in this book. We introduce the OOP and logic concepts in playgrounds and then move those concepts to Xcode and Swift. Many students are visual learners or learn by doing. We use both techniques. We'll walk you through topics and concepts with visual examples and then take you through step-by-step examples that reinforce the concepts.

We often repeat topics in different chapters to reinforce what you have learned and apply these skills in new ways. This enables new programmers to reapply development skills and feel a sense of accomplishment as they progress. Don't worry if you feel you haven't mastered a topic. Keep moving forward!

The Formula for Success

Learning to program is an interactive process between your program and you. Just like learning to play an instrument, you have to practice. You must work through the examples and exercises in this book. Understanding the concept doesn't mean you know how to apply it and use it.

You will learn a lot from this book. You will learn a lot from working through the exercises in this book. However, you will really learn when you debug your programs. Spending time walking through your code and trying to find out why it is not working the way you want is an unparalleled learning process. The downside of debugging is that a new developer can find it frustrating. If you have never wanted to throw your computer out the window, you will. You will question why you are doing this and whether you are smart enough to solve the problem. Programming is humbling, even for the most experienced developer.

Like a musician, the more you practice, the better you get. By practicing, we mean programming! You can do some amazing things as a programmer. The world is your oyster. Seeing your app in the iTunes App Store is one of the most satisfying accomplishments. However, there is a price, and that price is time spent coding and learning.

Having taught many students to become iOS developers, we have put together a formula for what makes students successful. Here is our formula for success:

- Believe you can do it. You'll be the only one who says you can't do this. So, don't tell yourself that.
- Work through all the examples and exercises in this book.
- Code, code, and keep coding. The more you code, the better you'll get.
- Be patient with yourself. If you were fortunate enough to have been a 4.0 student who could memorize material just by reading it, this will not happen with Swift coding. You are going to have to spend time coding.
- You learn by reading this book. You really learn by debugging your code.

- Use the free xcelMe.com webinars and YouTube videos mentioned at the end of this introduction. The free live and recorded training videos will be invaluable in quickly becoming a successful iOS developer.
- Don't give up!

The Development Technology Stack

We will walk you through the development process for your iOS apps and what technology you need. However, briefly looking at all the technology pieces together is helpful. These are the key iOS development technologies you will need to know in order to build a successful app and get it on the App Store:

- Apple's developer web site
- iTunes Connect
- Xcode
- Swift
- Object-oriented programming and logic
- Debugging
- Performance tuning

We know this is a lot of technology. Don't worry, we will go through it, and you will become comfortable using it.

Required Software, Materials, and Equipment

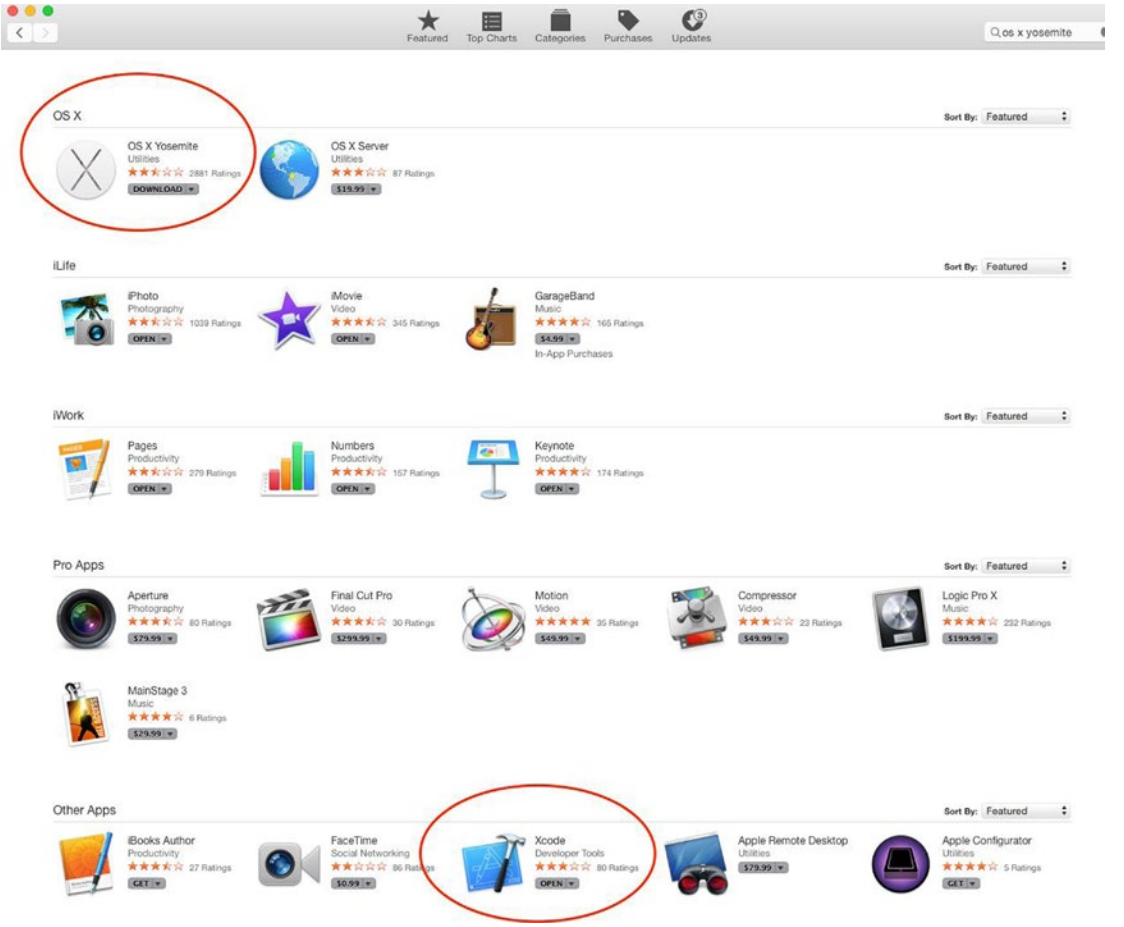
One of the great things about developing iOS apps is that everything you need to develop your app is free.

- Xcode
- Swift (included with Xcode)
- OSX 10.10 Yosemite or 10.11 El Capitan
- Integrated development environment
- iPhone and iPad simulators

All you need to get started is a Mac and knowledge of where to download everything. We will cover this.

Operating System and IDE

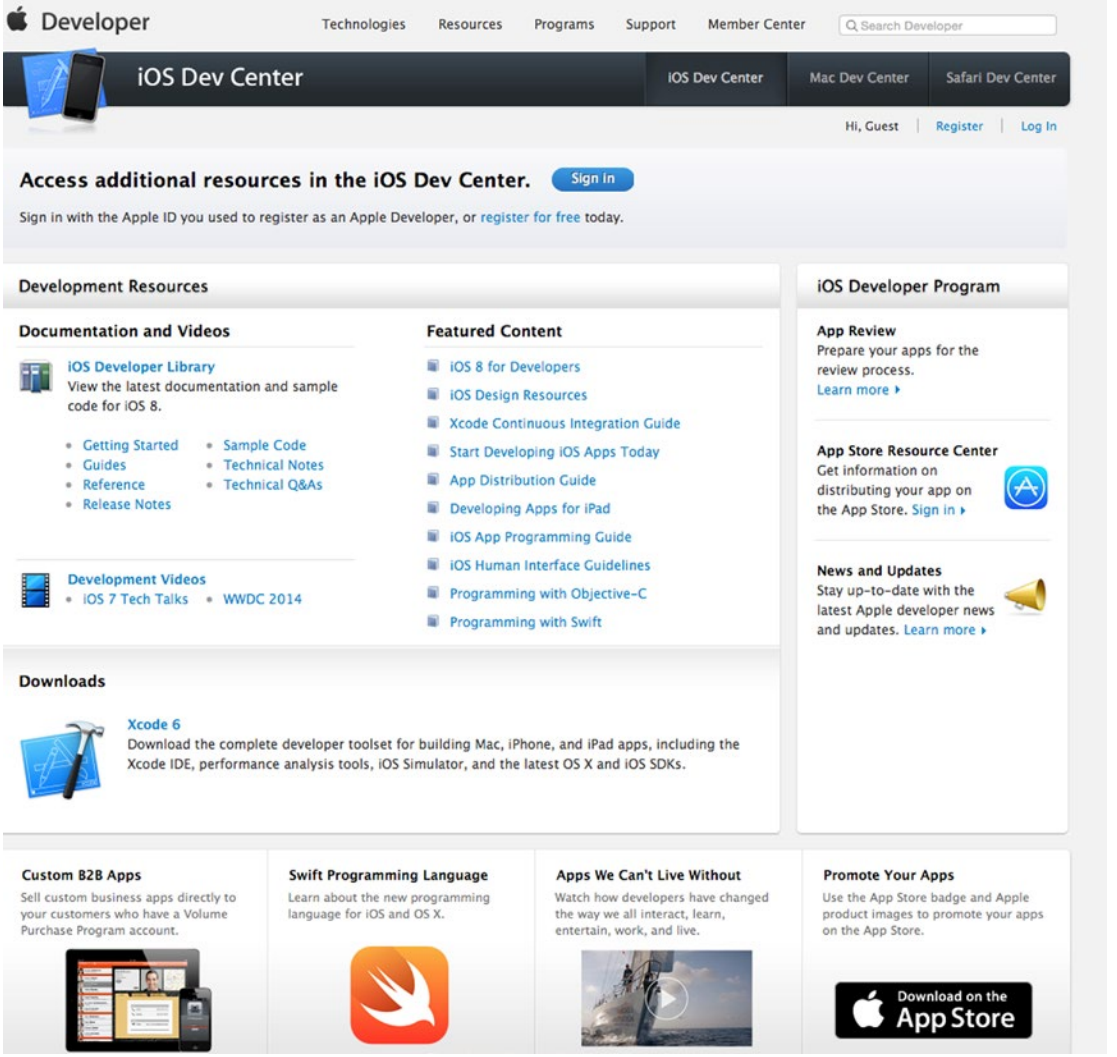
When developing iOS apps, you have to use Xcode and Mac OS X. You can download both of these for free from the Mac App Store.



Software Development Kits

You will need to register as an iOS developer. You can do this for free at http://developer.apple.com/iphone.

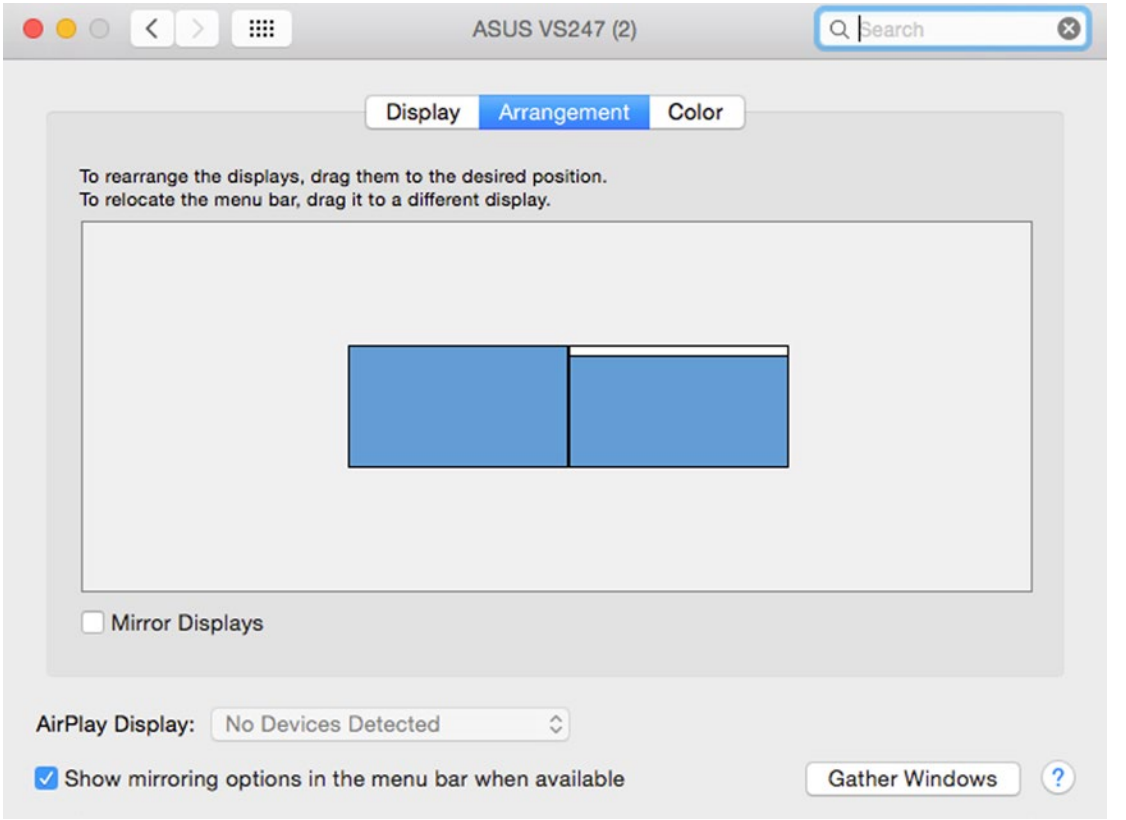
When you are ready to upload your app to the iTunes App Store, you will need to pay $99 per year in order to access iTunes Connect and upload your apps to the App Store.



Dual Monitors

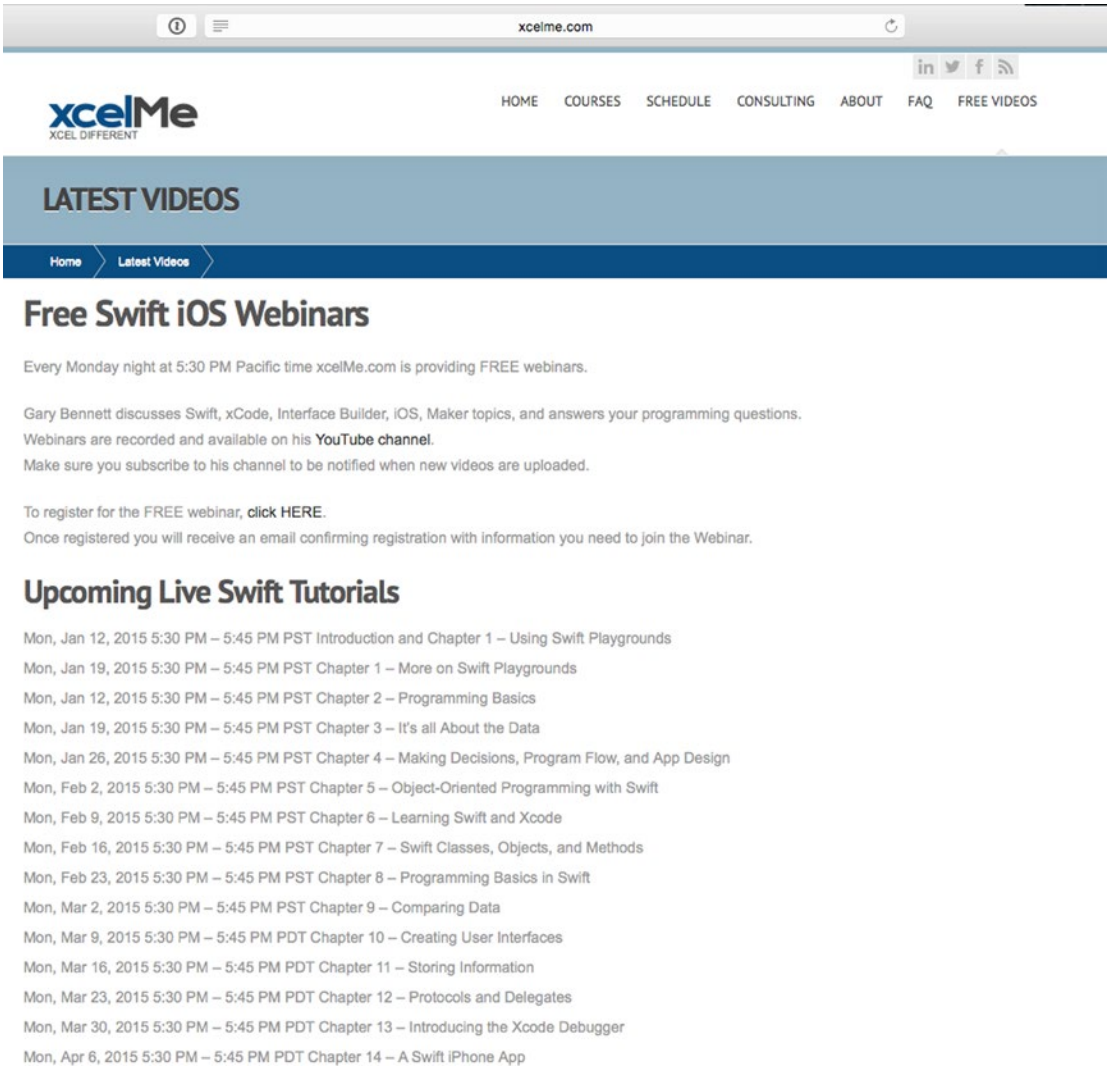
We recommend developers have a second monitor connected to their computers. It is great to step through your code and watch your output window and iOS simulator at the same time on dual independent monitors.

Apple hardware makes this easy. Just plug your second monitor into the display port of any Mac, with the correct Mini DisplayPort adapter, and you have two monitors working independently of one another. Note that dual monitors are not required. You will just have to organize your open windows to fit on your screen if you don't.



FREE LIVE WEBINARS, Q&A, AND YOUTUBE VIDEOS

Every Monday night at 5:30 p.m. Pacific time, we have live webinars and discuss a topic from the book or a timely item of interest. These webinars are free, and you can register for them at www.xcelme.com/latest-videos/.



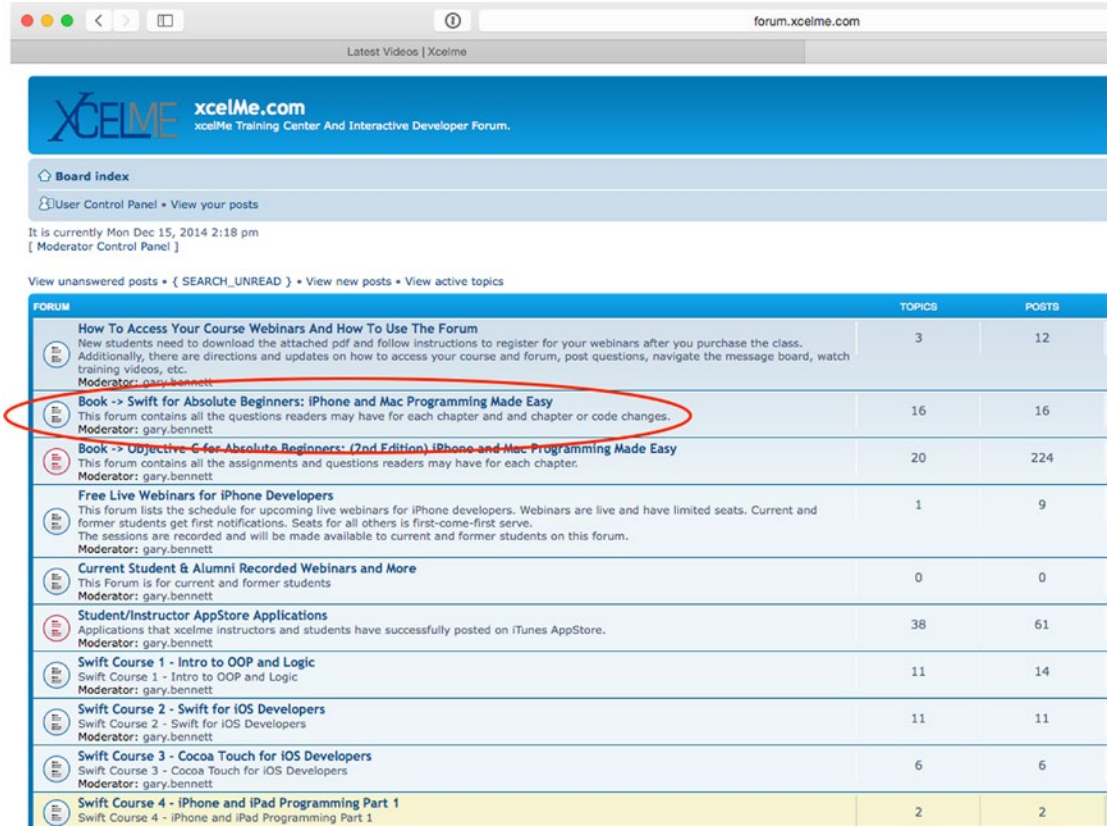
At the end of the webinars, we do a Q&A. You can ask a question on the topic discussed or on any topic in the book.

Additionally, all these webinars are recorded and available on YouTube. Make sure you subscribe to the YouTube channel so you are notified when new recordings are uploaded.

Free Book Forum

We have developed an online forum for this book at http://forum.xcelme.com, where you can ask questions while you are learning Swift and get answers from the authors. Also, Apple makes frequent changes to the programming language and SDK. We try our best to make sure any changes affecting the book are updated on the forum along with any significant text or code changes.

You can download the source code from the chapters on this forum too.



Chapter 1

Becoming a Great iOS Developer

Now that you're ready to become a software developer and have read the introduction of this book, you need to become familiar with several key concepts. Your computer program will do exactly what you tell it to do—no more and no less. It will follow the programming rules that were defined by the operating system and the Swift programming language. Your program doesn't care if you are having a bad day or how many times you ask it to perform something. Often, what you think you've told your program to do and what it actually does are two different things.

> **Key To Success** If you haven't already, take a few minutes to read the introduction of this book. The introduction shows you where to go to access the free webinars, forums, and YouTube videos that go with each chapter. Also, you'll better understand why this book uses the Swift playground programming environment and how to be successful in developing your iOS apps.

Depending on your background, working with something absolutely black and white may be frustrating. Many times, programming students have lamented, "That's not what I wanted it to do!" As you begin to gain experience and confidence in programming, you'll begin to think like a programmer. You will understand software design and logic, and you will experience having your programs perform exactly as you want and the satisfaction associated with this.

Thinking Like a Developer

Software development involves writing a computer program and then having a computer execute that program. A *computer program* is the set of instructions that you want the computer to perform. Before beginning to write a computer program, it is helpful to list the steps that you want your program to perform in the order you want them accomplished. This step-by-step process is called an *algorithm*.

If you want to write a computer program to toast a piece of bread, you would first write an algorithm. This algorithm might look something like this:

1. Take the bread out of the bag.
2. Place the bread in the toaster.
3. Press the toast button.
4. Wait for the toast to pop up.
5. Remove the toast from the toaster.

At first glance, this algorithm seems to solve the problem. However, the algorithm leaves out many details and makes many assumptions. Here are some examples:

- What kind of toast does the user want? Does the user want white bread, wheat bread, or some other kind of bread?
- How does the user want the bread toasted? Light or dark?
- What does the user want on the bread after it is toasted: butter, margarine, honey, or strawberry jam?
- Does this algorithm work for all users in their cultures and languages? Some cultures may have another word for toast or not know what toast is.

Now, you might be thinking this is getting too detailed for making a simple toast program. Over the years, software development has gained a reputation of taking too long, costing too much, and not being what the user wants. This reputation came to be because computer programmers often start writing their programs before they have actually thought through their algorithms.

The key ingredients to making successful applications are *design requirements*. Design requirements can be formal and detailed or simple like a list on a piece of paper. Design requirements are important because they help the developer flesh out what the application should do and not do when complete. Design requirements should not be completed in a programmer's vacuum, but should be produced as the result of collaboration between developers, users, and customers.

Another key ingredient to your successful app is the *user interface* (UI) design. Apple recommends you spend more than 50 percent of the entire development process focusing on the UI design. The design can be done using simple pencil and paper or using Xcode's storyboard feature to lay out your screen elements. Many software developers start with the UI design, and after laying out all the screen elements and having many users look at paper mock-ups, they then write the design requirements from their screen layouts.

Note If you take anything away from this chapter, take away the importance of considering design requirements and user interface design before starting software development. This is the most effective (and least expensive) use of time in the software development cycle. Using a pencil and eraser is a lot easier and faster than making changes to code because you didn't have others look at the designs before starting to program.

After you have done your best to flesh out all the design requirements, laid out all the user interface screens, and had the clients or potential customers look at your design and give you feedback, you can begin coding. Once coding begins, design requirements and user interface screens can change, but the changes are typically minor and easily accommodated by the development process. See Figures 1-1 and 1-2.

Figure 1-1. *This is a UI mock-up of the account balance screen for an iPhone mobile banking app before development begins on the original iPhone in 2010. This UI design mock-up was completed using OmniGraffle*

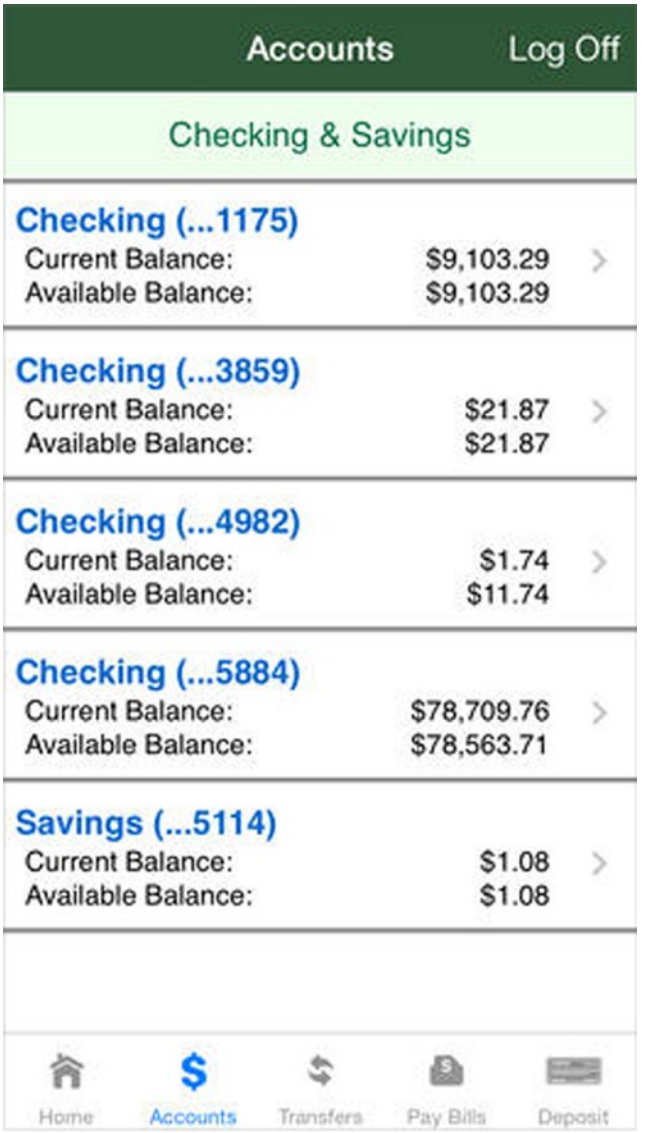


Figure 1-2. *This is a completed iPhone mobile banking application as it appeared on the App Store after several revisions in 2015. This app is called Woodforest Mobile Banking*

Figure 1-1 shows a mock-up of a mobile banking app screen prior to development. Developing mock-up screens along with design requirements forces developers to think through many of the application's usability issues before coding begins. This enables the application development time to be shortened and makes for a better user experience and better reviews on the App Store. Figure 1-2 shows how the view for the mobile banking app appears when completed.

Completing the Development Cycle

Now that you have the design requirements and user interface designs and have written your program, what's next? After programming, you need to make sure your program matches the design requirements and user interface design and ensure that there are no errors. In programming vernacular, errors are called *bugs*. Bugs are undesired results of your programming and must be fixed before the app is released to the App Store. The process of finding bugs in programs and making sure the program meets the design requirements is called *testing*. Typically, someone who is experienced in software testing methodology and who didn't write the app performs this testing. Software testing is commonly referred to as *quality assurance* (QA).

> **Note** When an application is ready to be submitted to the App Store, Xcode gives the file an `.app` or `.ipa` extension, for example, `appName.app`. That is why iPhone, iPad, and Mac applications are called *apps*. This book uses *program*, *application*, and *app* to mean the same thing.

During the testing phase, the developer will need to work with the QA staff to determine why the application is not working as designed. The process is called *debugging*. It requires the developer to step through the program to find out why the application is not working as designed. Figure 1-3 shows the complete software development cycle.

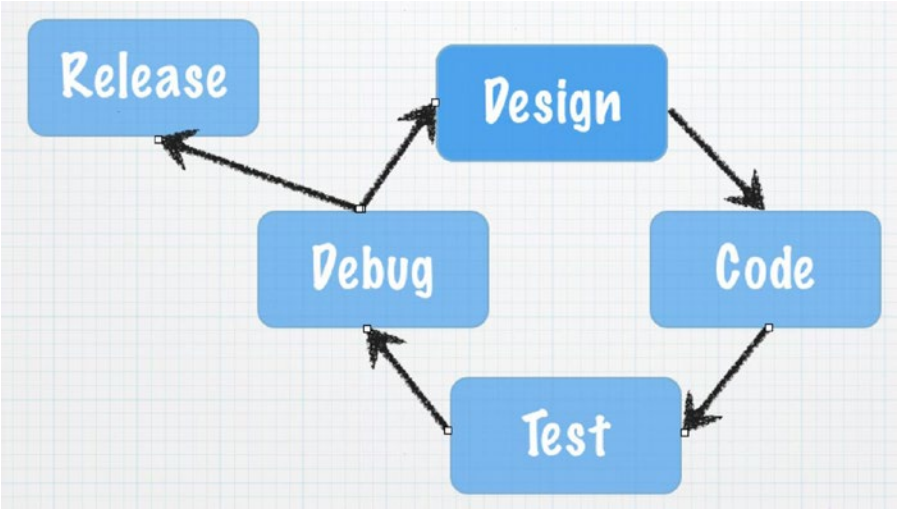


Figure 1-3. *The typical software development cycle*

Frequently during testing and debugging, changes to the requirements (design) must occur to make the application more usable for the customers. After the design requirements and user interface changes are made, the process starts again.