



Swift 4 for Absolute Beginners

Develop Apps for iOS

—

Fourth Edition

—

Stefan Kaczmarek

Brad Lees

Gary Bennett

Apress®

Swift 4 for Absolute Beginners

Develop Apps for iOS

Fourth Edition



Stefan Kaczmarek

Brad Lees

Gary Bennett

Apress®

Swift 4 for Absolute Beginners

Stefan Kaczmarek
Phoenix, Arizona, USA

Brad Lees
Phoenix, Arizona, USA

Gary Bennett
Scottsdale, Arizona, USA

ISBN-13 (pbk): 978-1-4842-3062-6

ISBN-13 (electronic): 978-1-4842-3063-3

<https://doi.org/10.1007/978-1-4842-3063-3>

Library of Congress Control Number: 2017963640

Copyright © 2018 by Stefan Kaczmarek, Brad Lees and Gary Bennett

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Cover image designed by Freepik

Managing Director: Welmoed Spahr
Editorial Director: Todd Green
Acquisitions Editor: Aaron Black
Development Editor: James Markham
Technical Editor: Massimo Nardone
Coordinating Editor: Jessica Vakili
Copy Editor: Karen Jameson
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 <http://www.apress.com/rights-permissions>.

Apress titles 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 Print and eBook Bulk Sales web page at <http://www.apress.com/bulk-sales>.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-3062-6. For more detailed information, please visit <http://www.apress.com/source-code>.

Printed on acid-free paper

Contents

About the Authors..... xi

About the Technical Reviewer xiii

Introductionxv

■ Chapter 1: Becoming a Great iOS Developer 1

 Thinking Like a Developer 1

 Completing the Development Cycle 5

 Introducing Object-Oriented Programming 6

 Working with the Playground Interface 10

 Summary 11

 What’s Next 11

■ Chapter 2: Programming Basics..... 13

 Touring Xcode..... 13

 Exploring the Workspace Window 14

 Navigating Your Workspace 15

 Editing Your Project Files 16

 Creating Your First Swift Playground Program 17

 Installing and Launching Xcode 9..... 18

 Using Xcode 9 20

Xcode Playground IDE: Editor and Results Areas	22
Summary	24
■ Chapter 3: It's All About the Data	25
Numbering Systems Used in Programming	25
Bits	25
Bytes.....	26
Hexadecimal	28
Unicode.....	30
Data Types	30
Declaring Constants and Variables.....	31
Optionals	32
Using Variables in Playgrounds	34
Summary	38
■ Chapter 4: Making Decisions, Program Flow, and App Design.....	39
Boolean Logic.....	39
Truth Tables	41
Comparison Operators.....	43
Designing Apps.....	43
Pseudocode	43
Optionals and Forced Unwrapping	46
Flowcharting.....	48
Designing and Flowcharting an Example App	48
The App's Design	49
Using Loops to Repeat Program Statements.....	50
Coding the Example App in Swift	52
Nested if Statements and else if Statements	55
Removing Extra Characters	55
Improving the Code Through Refactoring	55

Running the App	56
Design Requirements	57
Summary	60
■ Chapter 5: Object-Oriented Programming with Swift.....	63
The Object	64
What Is a Class?	65
Planning Classes	65
Planning Properties	66
Planning Methods.....	68
Implementing the Classes	70
Inheritance	77
Why Use OOP?	78
OOP Is Everywhere	78
Eliminate Redundant Code	78
Ease of Debugging.....	79
Ease of Replacement.....	79
Advanced Topics.....	79
Interface	79
Polymorphism.....	79
Value Orientated Programming	80
Summary	82
■ Chapter 6: Learning Swift and Xcode	83
A Newcomer	83
Understanding the Language Symbols.....	84
Implementing Objects in Swift	85
Creating the Project.....	87
Summary	105

■ Chapter 7: Swift Classes, Objects, and Methods	107
Creating a Swift Class	107
Properties	108
Methods.....	109
Using Your New Class.....	111
Creating Your Project	111
Adding Objects	114
Writing the Class	117
Creating the User Interface.....	119
Hooking Up the Code	124
Running the Program.....	128
Taking Type Methods to the Next Level	129
Accessing the Xcode Documentation.....	130
Summary.....	131
■ Chapter 8: Programming Basics in Swift	133
Using let vs. var	133
Understanding Collections	134
Using Arrays	134
Using the Dictionary Class	136
Creating the BookStore Application.....	137
Creating Your Class.....	142
Introducing Properties	143
Accessing Properties.....	144
Finishing the BookStore Program.....	144
Creating the View	144
Adding Properties	147
Adding a Description	149
Creating a Simple Data Model Class	151
Modifying MasterViewController	153
Modifying the DetailViewController	156
Summary.....	157

■ Chapter 9: Comparing Data	159
Revisiting Boolean Logic	159
Using Relational Operators	160
Comparing Numbers	161
Creating an Example Xcode App	162
Using Boolean Expressions	164
Comparing Strings	165
Using the switch Statement	167
Comparing Dates	168
Combining Comparisons	169
Summary	170
■ Chapter 10: Creating User Interfaces	171
Understanding Interface Builder	172
The Model-View-Controller Pattern	172
Human Interface Guidelines	174
Creating an Example iPhone App with Interface Builder	176
Using Interface Builder	181
The Document Outline	181
The Object Library	182
Inspector Pane and Selector Bar	186
Creating the View	186
Using Outlets	189
Using Actions	192
The Class	193
Summary	196
■ Chapter 11: Storing Information	199
Storage Considerations	199
Preferences/UserDefaults	200
Writing Preferences	200
Reading Preferences	201

Databases	202
Storing Information in a Database.....	202
Getting Started with Core Data.....	203
The Model.....	205
Managed Object Context	212
Setting Up the Interface.....	212
Summary	227
■ Chapter 12: Protocols and Delegates	229
Multiple Inheritance	229
Understanding Protocols	231
Protocol Syntax	232
Delegation	232
Protocol and Delegation Example	233
Getting Started	234
How It Works	246
Summary	246
■ Chapter 13: Introducing the Xcode Debugger	247
Getting Started with Debugging	248
Setting Breakpoints	249
Using the Breakpoint Navigator	250
Debugging Basics	252
Working with the Debugger Controls.....	254
Using the Step Controls.....	255
Looking at the Thread Window and Call Stack	257
Debugging Variables	257
Dealing with Code Errors and Warnings.....	259
Errors	259
Warnings.....	260
Summary	261

■ Chapter 14: A Swift iPhone App	263
Let's Get Started.....	263
Switches	276
Alert Controllers.....	277
App Summary.....	284
■ Chapter 15: Apple Watch and WatchKit.....	285
Considerations When Creating a watchOS App	285
Creating an Apple Watch App	286
Adding More Functionality	302
Summary	309
Index.....	311

About the Authors



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.



Brad Lees has more than a decade of experience in application development and server management. He has specialized in creating and initiating software programs in real-estate development systems and financial institutions. His career has been highlighted by his positions as information systems manager at The Lyle Anderson Company; product development manager for Smarsh; vice president of application development for iNation; and IT manager at The Orcutt/Winslow Partnership, the largest architectural firm in Arizona. A graduate of Arizona State University, Brad and his wife, Natalie, reside in Phoenix with their five children.



Gary Bennett teaches iPhone/iPad programming courses online. He has taught hundreds of students how to develop iPhone/iPad apps, and has several very popular apps on the iTunes App Store. Gary's students have some of the best-selling apps on the iTunes App Store. Gary also 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, chief information officer, and resident. As CIO, he helped take VistaCare public in 2002. He also coauthored *iPhone Cool Projects* (Apress, 2009). Gary lives in Scottsdale, Arizona, with his wife, Stefanie, and their four children.

About the Technical Reviewer



Massimo Nardone has more than 22 years of experiences in Security, Web/Mobile development, Cloud, and IT Architecture. His true IT passions are Security and Android.

He has been programming and teaching how to program with Android, Perl, PHP, Java, VB, Python, C/C++, and MySQL for more than 20 years.

He holds a Master of Science degree in Computing Science from the University of Salerno, Italy.

Massimo has worked as a Project Manager, Software Engineer, Research Engineer, Chief Security Architect, Information Security Manager, PCI/SCADA Auditor, and Senior Lead IT Security/Cloud/SCADA Architect for many years.

Technical skills include Security, Android, Cloud, Java, MySQL, Drupal, Cobol, Perl, Web and Mobile development, MongoDB, D3, Joomla, Couchbase, C/C++, WebGL, Python, Pro Rails, Django CMS, Jekyll, Scratch, etc.

Currently he works as Chief Information Security Office (CISO) for Cargotec Oyj.

He worked as visiting lecturer and supervisor for exercises at the Networking Laboratory of the Helsinki University of Technology (Aalto University). He holds four international patents (PKI, SIP, SAML, and Proxy areas).

Massimo has reviewed more than 40 IT books for different publishing companies, and he is the coauthor of *Pro Android Games* (Apress, 2015).

This book is dedicated to Antti Jalonen and his family who are always there when I need them.

Introduction

Over the last eight years, we've heard the following countless times:

- “I've never programmed before, but I have a great idea for an iPhone/iPad/Apple TV 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 may have never programmed before. The book is also written for someone who may have never programmed before using object-oriented programming (OOP) languages. There are many 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 last eight years, we have taught thousands of students at xcelMe.com to be iOS (iPhone/iPad) developers. Many of our students have developed some of the most successful iOS apps in their category in the 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 Swift Playgrounds and then move those concepts to Xcode. Many students are visual 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 reinforcing 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 a 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 a new developer can find it especially 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 very 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 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 keeping 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 can 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 explained at the end of this Introduction
- Don't give up!

The Development Technology Stack

We will walk you through the process of understanding 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 Website
- App Analytics
- iOS SDK
- Swift
- Object Oriented Programming and Logic
- Xcode Integrated Developers Environment (IDE)
- Debugging
- Performance Tuning

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

Required Software, Materials, and Equipment

One of the great things about developing iOS apps is just about everything is free to get develop your app.

- Xcode
- Swift
- macOS 10.12.6 or higher
- iOS SDK
- iOS Simulator

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

Operating System and IDE

Developing iOS apps you have to use Xcode on a Mac. You can download Xcode for free from the Mac App Store. (See Figure 1)



Figure 1. Downloading Xcode from the Mac App Store

Software Development Kits

You will need to register as a developer. You can do this for free at <https://developer.apple.com/ios> (see Figure 2)



What's New in iOS 11

iOS 11 sets a new standard for the world's most advanced mobile operating system. Your apps can now become more intelligent using the power of machine learning with Core ML. You can create incredible augmented reality experiences with ARKit. And you can deliver a more unified and immersive user experience with new multitasking features, including drag and drop for iPad, the new Files app, new camera APIs, new SiriKit domains, Apple Music integration, and more.



Figure 2. Apple's Developer Website (editor, caption not sure why I can't apply that style)

When you are ready to upload your app to the App Store, you will need to pay \$99/year in order to publish it.

Dual Monitors (editor not sure why this “Strong” format is doing this)

We recommend developers have a second monitor connected to their computer. 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 the port of any Mac, with the correct adapter of course, and you have two monitors working independently of one another. See Figure 3. 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.

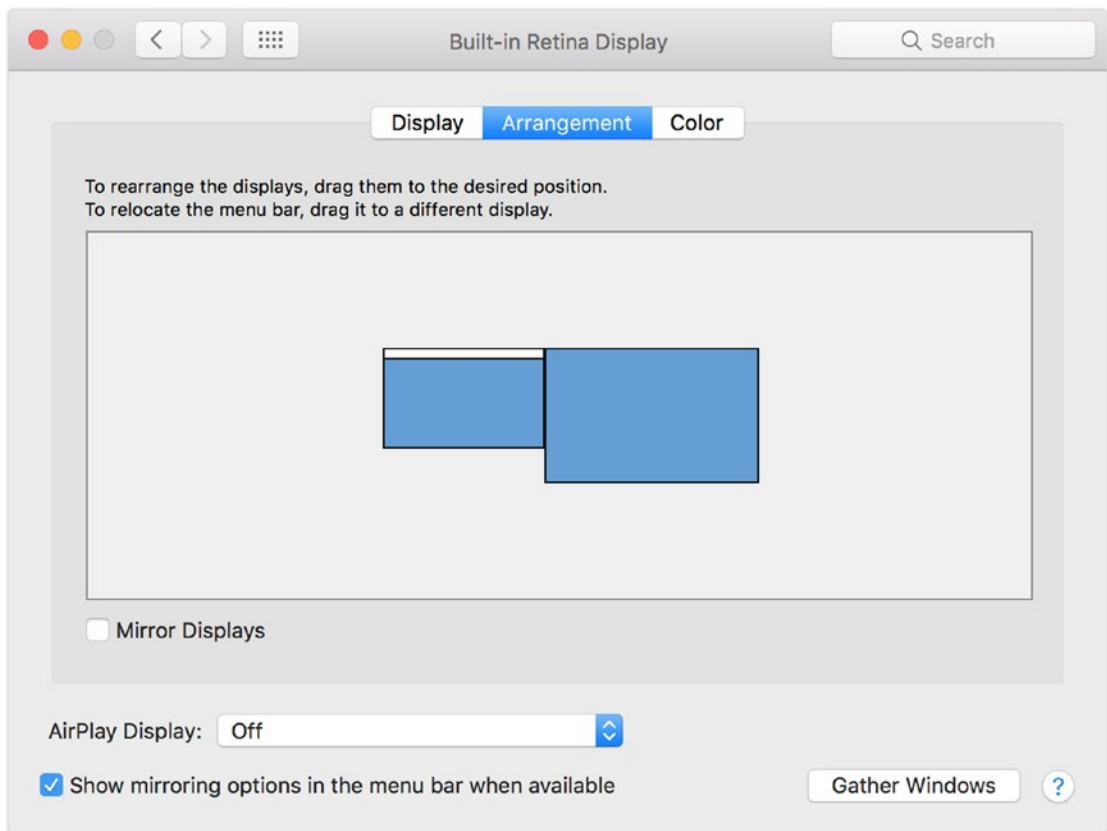


Figure 3. Arranging Dual Monitors on a Mac

Free Live Webinars, Q&A, and YouTube Videos

Nearly every week, 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 <http://www.xcelme.com/latest-videos/>. See Figure 4.

xcelMe
XCEL DIFFERENT

HOME COURSES SCHEDULE CONSULTING ABOUT CONTACT US FAQ FREE VIDEOS

LATEST VIDEOS

Home > Latest Videos >

Free Swift iOS & tvOS Webinars

Every Friday at 10:30am Pacific time xcelMe.com is providing FREE webinars.

Gary Bennett discusses Swift 2.0, tvOS, xCode, Interface Builder, iOS, Maker topics, and answers your programming questions. Webinars are recorded and available on his **YouTube channel**. Make sure you subscribe to his channel to be notified when new videos are uploaded.

To register for the FREE webinar, **click HERE**.
Once registered you will receive an email confirming registration with information you need to join the Webinar.

Recorded Chapter Tutorials

- Using Swift 2 Playgrounds – Recorded 11/16/2015 ([Click Here](#))
- More on Swift 2 Playgrounds – Recorded 11/16/2015 ([Click Here](#))
- One Hour of Code – IBM's New Swift Playground Recorded 12/9/2015 ([Click Here](#))
- Basic Swift 2 Data Types Recorded 12/14/2015 ([click here](#))
- It's all About the Data ([click here](#))
- Making Decisions, Program Flow, and App Design ([click here](#))
- Optionals and Forced Unwrapping ([click here](#))
- Swift Classes, Objects, and Methods ([click here](#))
- Programming Basics in Swift ([click here](#))
- Comparing Data ([click here](#))
- Creating User Interfaces ([click here](#))
- Storing Information ([click here](#))
- Introducing the Xcode Debugger ([click here](#))
- More Delegates and Protocols ([click here](#))


Figure 4. Swift Webinars

At the end of the webinars, we do a Q&A. You can ask a question on the topic discussed or 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. You will also find answers to the exercises and additional exercises to help you learn. See Figure 5.

 **xcelme.com**
xcelMe Training Center And Interactive Developer Forum.

[Board index](#)

[FAQ](#) [Members](#) [Register](#) [Login](#)

It is currently Sat Dec 16, 2017 12:25 pm













FORUM	TOPICS	POSTS	LAST POST
 How To Access Your Course Webinars And How To Use The Forum New students need to download the attached pdf and follow instructions to register for your webinars after you purchase the class. Additionally, there are directions and updates on how to access your course and forum, post questions, navigate the message board, watch training videos, etc. Moderator: gary.bennett	3	12	by zenith9356  Thu Mar 13, 2014 10:24 am
 Book -> Swift 3.0 for Absolute Beginners: iPhone and Mac Programming Made Easy 3rd Edition This forum contains answers readers may have for each chapter as well as any corrections to the book. The forum also contains the Source Code for the book. Moderator: gary.bennett	17	21	by roman_plains  Sun Jun 18, 2017 5:28 pm
 Book -> Swift 2.0 for Absolute Beginners: iPhone and Mac Programming Made Easy 2nd Edition This forum contains answers readers may have for each chapter as well as any corrections to the book. The forum also contains the Source Code for the book. Moderator: gary.bennett	17	96	by zany76  Thu Aug 31, 2017 3:11 pm
 Book -> Developing for Apple TV using tvOS and Swift This forum contains answers readers may have for each chapter as well as any corrections to the book. The forum also contains the Source Code for the book. Moderator: gary.bennett	10	12	by mdstebel  Mon Jun 13, 2016 11:26 am
 Book -> Objective-C for Absolute Beginners: (2nd Edition) iPhone and Mac Programming Made Easy This forum contains all the assignments and questions readers may have for each chapter. Moderator: gary.bennett	20	224	by Drago  Mon Jun 16, 2014 9:27 pm
 Free Live Webinars for iPhone Developers This forum lists the schedule for upcoming live webinars for iPhone developers. Webinars are live and have limited seats. Current and former students get first notifications. Seats for all others is first-come-first-serve. The sessions are recorded and will be made available to current and former students on this forum. Moderator: gary.bennett	1	9	by Miptigningwaw  Tue Nov 29, 2011 3:48 am

Figure 5. Reader forum for accessing answers to exercises and posting questions for authors

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, 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. The algorithm might look something like this:

1. Take the bread out of the bag.
2. Place a slice of 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 and should 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 write the design requirements from their screen layouts.

Note If you take anything away from this chapter, let it be 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 are easily accommodated by the development process. See Figures 1-1 and 1-2.

Figure 1-1 shows a mock-up of a rental report 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 rental report app appears when completed. Notice how mock-up tools enable you to model the app to the real thing.

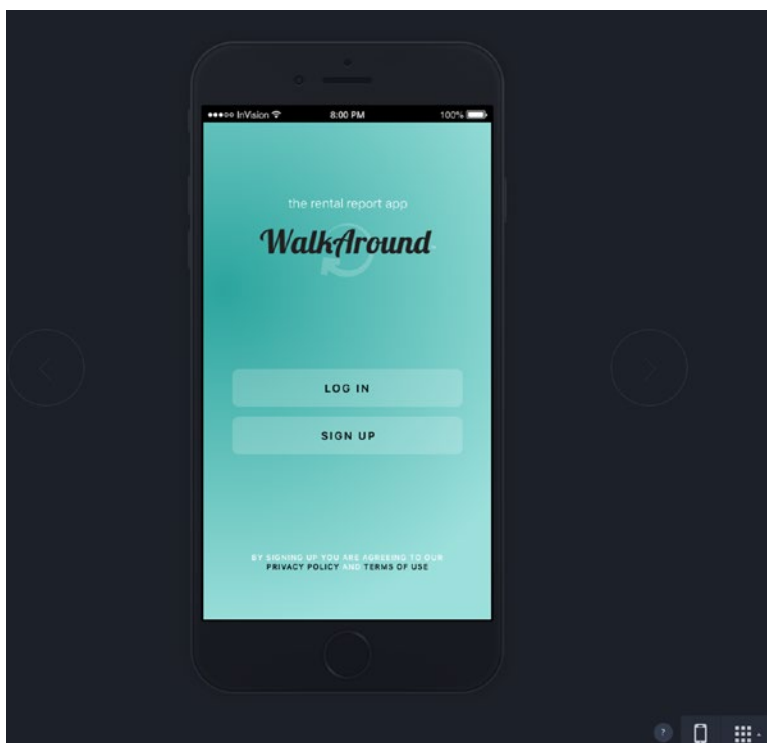


Figure 1-1. This is a UI mock-up of the Log In screen for an iPhone mobile rental report app before development begins. This UI design mock-up was completed using InVision.

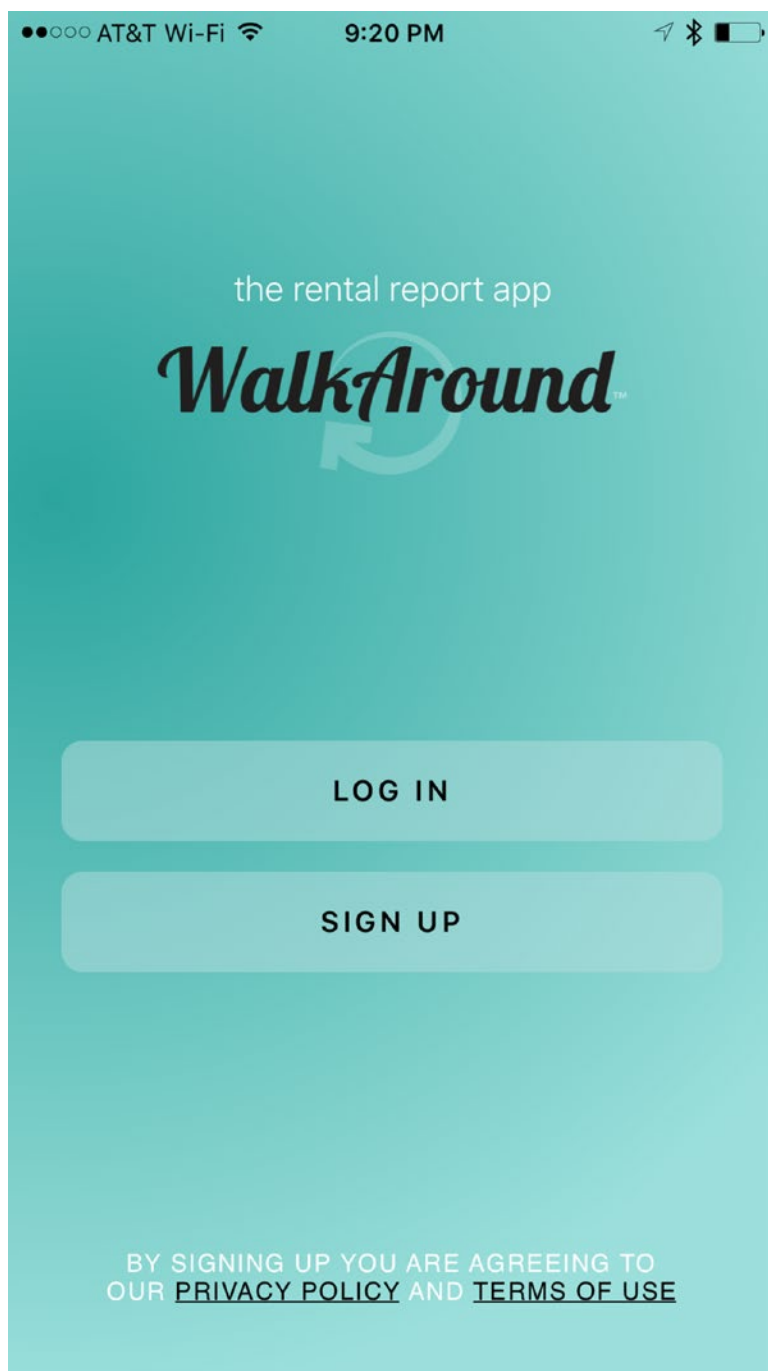


Figure 1-2. This is the completed iPhone rental report app. This app is called WalkAround.

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.

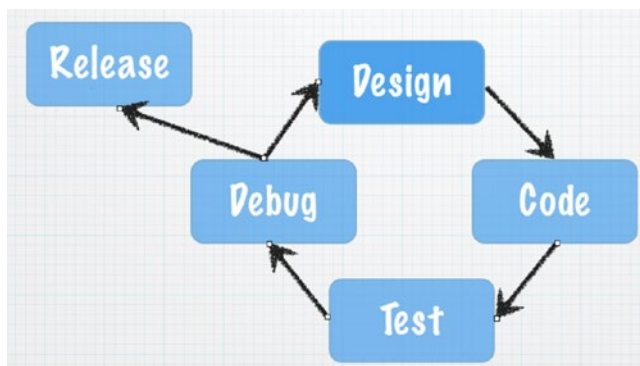


Figure 1-3. The typical software development cycle

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.

At some point, the application that everyone has been working so hard on must be shipped to the App Store. Many considerations are taken into account as to when in the cycle this happens:

- Cost of development
- Budget
- Stability of the application
- Return on investment

There is always the give-and-take between developers and management. Developers want the app to be perfect, and management wants to start realizing revenue from the investment as soon as possible. If the release date were left up to the developers, the app would likely never ship to the App Store. Developers would continue to tweak the app forever, making it faster, more efficient, and more usable. At some point, however, the code needs to be pried from the developers' hands and uploaded to the App Store so it can do what it was meant to do.

Introducing Object-Oriented Programming

As discussed in detail in the introduction, playgrounds enable you to focus on **object-oriented programming (OOP)** without having to cover all the Swift programming syntax and complex Xcode development environment in one big step. Instead, you can focus on learning the basic principles of OOP and using those principles quickly to write your first programs.

For decades, developers have been trying to figure out a better way to develop code that is reusable, manageable, and easily maintained over the life of a project. OOP was designed to help achieve code reuse and maintainability while reducing the cost of software development.

OOP can be viewed as a collection of objects in a program. Actions are performed on these objects to accomplish the design requirements.

An **object** is anything that can be acted on. For example, an airplane, person, or screen/view on the iPad can all be objects. You may want to act on the plane by making the plane bank. You may want the person to walk or to change the color of the screen of an app on the iPad.

Playgrounds execute your code as you complete each line, such as the one shown in Figure 1-4. When you run your playground applications, the user can apply actions to the objects in your application. Xcode is an **integrated development environment** (IDE) that enables you to run your application from within your programming environment. You can test your applications on your computer first before running them on your iOS devices by running the apps in Xcode's simulator, as shown in Figure 1-5.

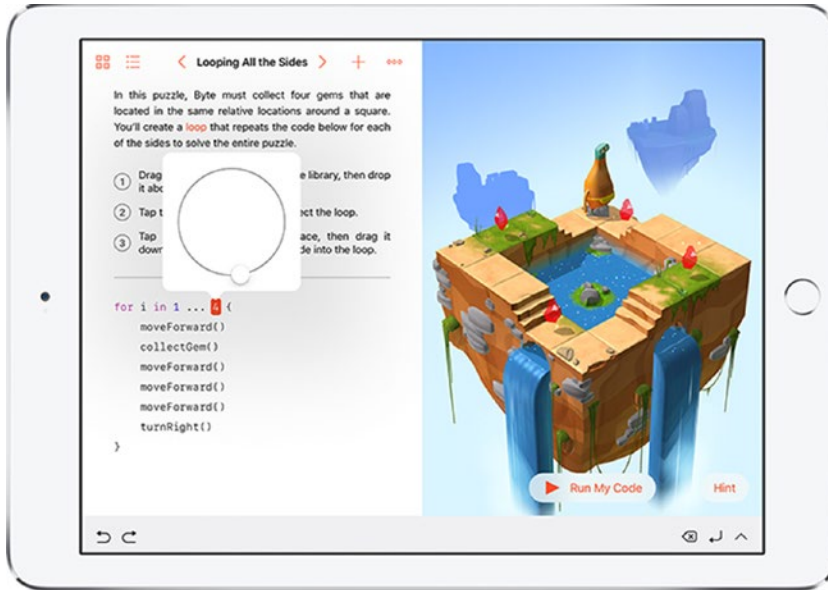


Figure 1-4. There are multiple objects in this playground view

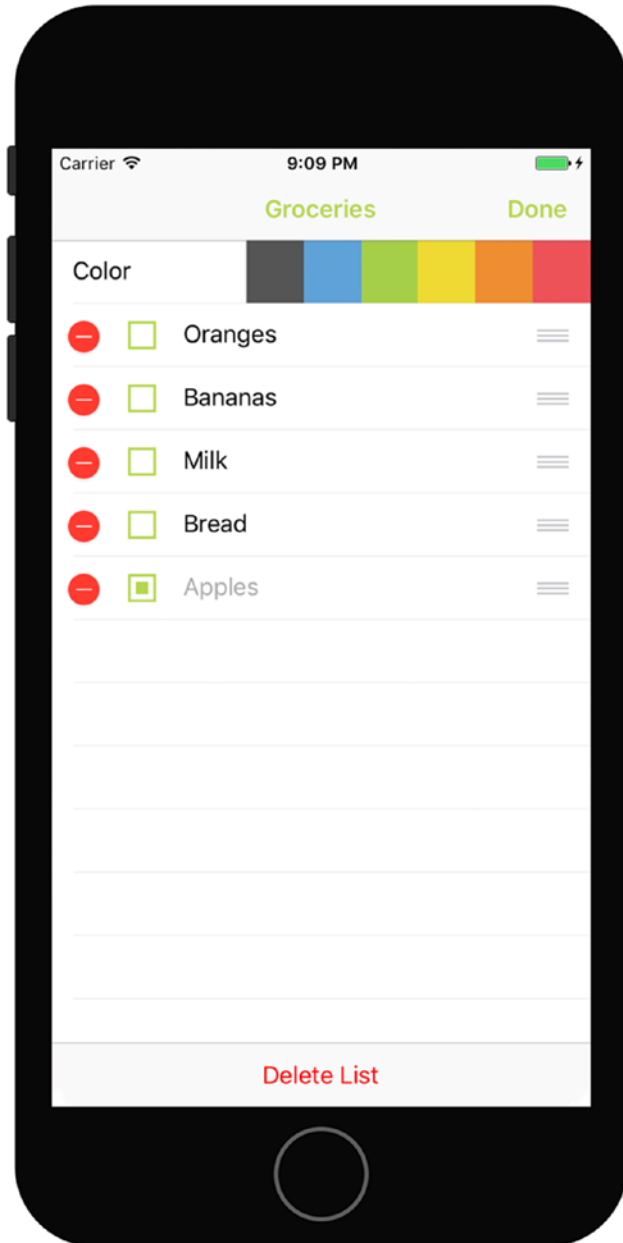


Figure 1-5. This sample iPhone app contains a table object to organize a list of groceries. Actions such as “rotate left” or “user did select row 3” can be applied to this object.

Actions that are performed on objects are called **methods**. Methods manipulate objects to accomplish what you want your app to do. For example, for a jet object, you might have the following methods:

```
goUp  
goDown
```

```
bankLeft  
turnOnAfterburners  
lowerLandingGear
```

The table object in Figure 1-5 is actually called `UITableView` when you use it in a program, and it could have the following methods:

```
numberOfRowsInSection  
cellForRowAtIndexPath  
canEditRowAtIndexPath  
commitEditingStyle  
didSelectRowAtIndexPath
```

Most objects have data that describes those objects. This data is defined as *properties*. Each property describes the associated object in a specific way. For example, the jet object's properties might be as follows:

```
altitude = 10,000 feet  
heading = North  
speed = 500 knots  
pitch = 10 degrees  
yaw = 20 degrees  
latitude = 33.575776  
longitude = -111.875766
```

For the `UITableView` object in Figure 1-5, the following might be the properties:

```
backgroundColor = White  
selectedRow = 3  
animateView = No
```

An object's properties can be changed at any time when your program is running, when the user interacts with the app, or when the programmer designs the app to accomplish the design requirements. The values stored in the properties of an object at a specific time are collectively called the **state of an object**.

State is an important concept in computer programming. When teaching students about state, we ask them to go over to a window and find an airplane in the sky. We then ask them to snap their fingers and make up some of the values that the plane's properties might have at that specific time. Those values might be as follows:

```
altitude = 10,000 feet  
latitude = 33.575776  
longitude = -111.875766
```

Those values represent the *state* of the object at the specific time that they snapped their fingers.

After waiting a couple of minutes, we ask the students to find that same plane, snap their fingers again, and record the plane's possible state at that specific point in time.