John Ray

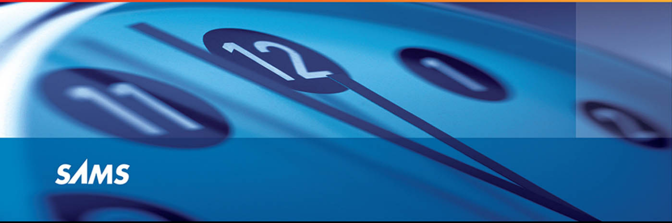
In **Full Color**

Figures and code appear as they do in Xcode 7.x

Covers **iOS 9, Xcode 7.x, iPhone, iPad, and More!**

Additional files and updates available online

Sams Teach Yourself iOS® 9 Application Development in 24 Hours



SAMS

John Ray

Sams Teach Yourself iOS® 9 Application Development in 24 Hours

SAMS 800 East 96th Street, Indianapolis, Indiana, 46240 USA

Sams Teach Yourself iOS9® Application Development in 24 Hours

ISBN-13: 978-0-672-33767-3
ISBN-10: 0-672-33767-3

Library of Congress Control Number: 2015917495

First Printing February 2016

Trademarks

Warning and Disclaimer

Special Sales

Editor-in-Chief
Greg Wiegand

Acquisitions Editor
Laura Norman

Development Editor
Keith Cline

Managing Editor
Kristy Hart

Senior Project Editor
Lori Lyons

Copy Editor
Keith Cline

Indexer
Publishing Works

Proofreader
Laura Hernandez

Technical Editor
Anne Groves

Editorial Assistant
Sandra Fugate

Cover Designer
Mark Shirar

Compositor
Nonie Ratcliff

Contents at a Glance

Introduction 1
HOUR 1 Preparing Your System and iDevice for Development 5
2 Introduction to Xcode and the iOS Simulator 29
3 Discovering Swift and the iOS Playground 73
4 Inside Cocoa Touch 123
5 Exploring Interface Builder 155
6 Model-View-Controller Application Design 193
7 Working with Text, Keyboards, and Buttons 223
8 Handling Images, Animation, Sliders, and Steppers 259
9 Using Advanced Interface Objects and Views 291
10 Getting the User's Attention 329
11 Implementing Multiple Scenes and Popovers 361
12 Making Choices with Toolbars and Pickers 413
13 Advanced Storyboards Using Navigation and Tab Bar Controllers 457
14 Navigating Information Using Table Views and Split View Controllers 497
15 Reading and Writing Application Data 539
16 Building Responsive User Interfaces 585
17 Using Advanced Touches and Gestures 627
18 Sensing Orientation and Motion 663
19 Working with Rich Media 693
20 Interacting with Other iOS Services 743
21 Implementing Location Services 781
22 Building Background-Ready Applications 813
23 Universal Applications and Size Classes 853
24 Application Tracing, Monitoring, and Debugging 877
Index 903
A Introducing Xcode Source Control ..... Online

Note: Appendix A is a bonus online chapter. To access it, go to www.informit.com/title/9780672337673 and click the Downloads tab.

Table of Contents

Introduction **1**

Who Can Become an iOS Developer? 2
Who Should Use This Book? 2
What Is (and Isn't) in This Book? 3

Hour 1: Preparing Your System and iDevice for Development **5**

Welcome to the iOS Platform 5
Becoming an iOS Developer 10
Running an iOS App 15
Developer Technology Overview 22
Further Exploration 24
Summary 24
Q&A 24
Workshop 25
Activities 27

Hour 2: Introduction to Xcode and the iOS Simulator **29**

Using Xcode 29
Using the iOS Simulator 61
Further Exploration 69
Summary 69
Q&A 69
Workshop 70
Activities 72

Hour 3: Discovering Swift and the iOS Playground **73**

Object-Oriented Programming and Swift 73
The Terminology of Object-Oriented Development 75
Exploring the Swift File Structure 78
Swift Programming Basics 84
Memory Management and Automatic Reference Counting 110

Introducing the iOS Playground ... 111
Further Exploration ... 118
Summary ... 119
Q&A ... 119
Workshop ... 120
Activities ... 122

Hour 4: Inside Cocoa Touch **123**
What Is Cocoa Touch? ... 123
Exploring the iOS Technology Layers ... 125
Tracing the iOS Application Life Cycle ... 132
Cocoa Fundamentals ... 134
Exploring the iOS Frameworks with Xcode ... 142
Further Exploration ... 149
Summary ... 149
Q&A ... 150
Workshop ... 150
Activities ... 153

Hour 5: Exploring Interface Builder **155**
Understanding Interface Builder ... 155
Creating User Interfaces ... 162
Customizing the Interface Appearance ... 172
Connecting to Code ... 178
Further Exploration ... 188
Summary ... 189
Q&A ... 189
Workshop ... 190
Activities ... 192

Hour 6: Model-View-Controller Application Design **193**
Understanding the MVC Design Pattern ... 193
How Xcode Implements MVC ... 195
Using the Single View Application Template ... 200
Further Exploration ... 218

Summary 218
Q&A 219
Workshop 219
Activities 222

Hour 7: Working with Text, Keyboards, and Buttons 223
Basic User Input and Output 223
Using Text Fields, Text Views, and Buttons 225
Further Exploration 253
Summary 254
Q&A 255
Workshop 255
Activities 257

Hour 8: Handling Images, Animation, Sliders, and Steppers 259
User Input and Output 259
Creating and Managing Image Animations, Sliders, and Steppers 261
Further Exploration 285
Summary 285
Q&A 286
Workshop 286
Activities 289

Hour 9: Using Advanced Interface Objects and Views 291
User Input and Output (Continued) 291
Using Switches, Segmented Controls, and Web Views 298
Using Scrolling and Stack Views 315
Further Exploration 324
Summary 325
Q&A 325
Workshop 326
Activities 328

Hour 10: Getting the User's Attention 329
Alerting the User 329
Exploring User Alert Methods 340

Further Exploration 355
Summary 355
Q&A 356
Workshop 356
Activities 359

Hour 11: Implementing Multiple Scenes and Popovers 361
Introducing Multiscene Storyboards 362
Using Segues 388
Popovers, Universal Applications, and iPhones 402
Further Exploration 407
Summary 408
Q&A 408
Workshop 409
Activities 411

Hour 12: Making Choices with Toolbars and Pickers 413
Understanding the Role of Toolbars 413
Exploring Pickers 416
Using the Date Picker 424
Using a Custom Picker 437
Further Exploration 452
Summary 453
Q&A 453
Workshop 454
Activities 456

Hour 13: Advanced Storyboards Using Navigation and Tab Bar Controllers 457
Advanced View Controllers 457
Exploring Navigation Controllers 459
Understanding Tab Bar Controllers 464
Using a Navigation Controller 470
Using a Tab Bar Controller 481
Further Exploration 490

Summary 491
Q&A 492
Workshop 492
Activities 495

Hour 14: Navigating Information Using Table Views and Split View Controllers 497
Understanding Tables 497
Exploring the Split View Controller 506
A Simple Table View Application 509
Creating a Master-Detail Application 519
Further Exploration 533
Summary 534
Q&A 534
Workshop 535
Activities 537

Hour 15: Reading and Writing Application Data 539
iOS Applications and Data Storage 539
Data Storage Approaches 542
Creating Implicit Preferences 551
Implementing System Settings 558
Implementing File System Storage 571
Further Exploration 579
Summary 579
Q&A 580
Workshop 580
Activities 583

Hour 16: Building Responsive User Interfaces 585
Responsive Interfaces 585
Using Auto Layout 590
Programmatically Defined Interfaces 615
Further Exploration 622
Summary 622
Q&A 622

Workshop 623
Activities 625

Hour 17: Using Advanced Touches and Gestures 627
Multitouch Gesture Recognition 627
3D Touch Peek and Pop 630
Using Gesture Recognizers 633
Implementing 3D Touch Gestures 654
Further Exploration 658
Summary 659
Q&A 660
Workshop 660
Activities 662

Hour 18: Sensing Orientation and Motion 663
Understanding Motion Hardware 663
Accessing Orientation and Motion Data 666
Sensing Orientation 670
Detecting Acceleration, Tilt, and Rotation 675
Further Exploration 687
Summary 688
Q&A 688
Workshop 688
Activities 691

Hour 19: Working with Rich Media 693
Exploring Rich Media 693
The Media Playground Application 709
Further Exploration 736
Summary 737
Q&A 737
Workshop 738
Activities 741

Hour 20: Interacting with Other iOS Services **743**

Extending iOS Service Integration 743
Using Contacts, Email, Social Networking, Safari, and Maps 761
Further Exploration 776
Summary 777
Q&A 777
Workshop 777
Activities 780

Hour 21: Implementing Location Services **781**

Understanding Core Location 781
Creating a Location-Aware Application 789
Using the Magnetic Compass 799
Further Exploration 808
Summary 809
Q&A 809
Workshop 809
Activities 812

Hour 22: Building Background-Ready Applications **813**

Understanding iOS Backgrounding 813
Disabling Backgrounding 821
Handling Background Suspension 823
Implementing Local Notifications 824
Using Task-Specific Background Processing 828
Completing a Long-Running Background Task 833
Performing a Background Fetch 839
Adding 3D Touch Quick Actions 844
Further Exploration 848
Summary 848
Q&A 848
Workshop 849
Activities 851

Hour 23: Universal Applications and Size Classes **853**

Universal Application Development ... 853

Size Classes ... 857

Further Exploration ... 873

Summary ... 873

Q&A ... 874

Workshop ... 874

Activities ... 875

Hour 24: Application Tracing, Monitoring, and Debugging **877**

Instant Feedback with `NSLog` ... 878

Using the Xcode Debugger ... 881

Further Exploration ... 898

Summary ... 898

Q&A ... 899

Workshop ... 899

Activities ... 901

Index **903**

Appendix A: Introducing Xcode Source Control **Online**

Note: Appendix A is a bonus online chapter. To access it, go to www.informit.com/title/9780672337673 and click the Downloads tab.

About the Author

John Ray currently serves as the Director of the Office of Research Information Systems at The Ohio State University. He has written numerous books for Macmillan/Sams/Que, including *Using TCP/IP: Special Edition*, *Teach Yourself Dreamweaver MX in 21 Days*, *Mac OS X Unleashed*, *My OS X – El Capitan Edition*, and *Sams Teach Yourself iOS 8 Development in 24 Hours*. As a Macintosh user since 1984, he strives to ensure that each project presents the Macintosh with the equality and depth it deserves. Even technical titles such as Using TCP/IP contain extensive information about the Macintosh and its applications and have garnered numerous positive reviews for their straightforward approach and accessibility to beginner and intermediate users.

You can visit his website at http://teachyourselfios.com or follow him on Twitter at @johnemeryray or #iOSIn24.

Dedication

This book is dedicated to the stray cat living in my garage.
It appears to like the cat food I bought.

Acknowledgments

Thank you to the group at Sams Publishing—Laura Norman, Keith Cline, Mark Renfrow—and my Tech Editor, Anne Groves, for helping me survive another year of OS and Xcode updates. The evolution of Swift (and the Xcode tools) has been incredibly rapid, and the ride for developers is definitely a bit bumpy. The Pearson team does a great job of keeping me organized, honest, and on target.

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

We welcome your comments. You can email or write to let us know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that we cannot help you with technical problems related to the topic of this book.

When you write, please be sure to include this book's title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

Email: feedback@samspublishing.com

Mail: Sams Publishing
ATTN: Reader Feedback
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

Register your copy of *Sams Teach Yourself iOS 9 Application Development* at informit.com for convenient access to downloads, updates, and corrections as they become available. To start the registration process, go to informit.com/register and log in or create an account*. Enter the product ISBN, 9780672337673, and click Submit. Once the process is complete, you will find any available bonus content under Registered Products.

*Be sure to check the box that you would like to hear from us in order to receive exclusive discounts on future editions of this product.

Introduction

When you pick up an iOS device and use it, you feel connected. Whether it be an iPad, an iPhone, or an iPod, the interface acts as an extension to your fingers; it is smooth, comfortable, and invites exploration. Other competing devices offer similar features, and even sport gadgets such as wraparound screens and trackpads, but they cannot match the user experience that is iOS.

iOS and its associated development tools have changed rapidly over the past few years. iOS 7 brought us a new user interface that used depth and translucency to keep users connected to their content and aware of the context in which they are accessing it. iOS 8 surprised everyone with a brand new language for developing apps: Swift. Alongside the introduction of iOS 9, Swift became an open source language, solidifying it as the future of development on Apple platforms and beyond.

Swift marks a dramatic change in the history of iOS and OS X development. With Swift, Apple has effectively retired the Objective-C language—used on Apple and NeXT platforms for over 25 years. Swift offers a friendlier development platform with more modern language features and tools. While in development for more than 4 years at Apple, by the time this book reaches you, Swift will have existed as a public programming language for a little over a year.

Unfortunately, there are some caveats to writing about a young language. Swift is rapidly evolving, and changes with each release of Apple's development environment: Xcode. Code that is written in one version of Xcode sometimes breaks in the next. In the version of Swift shipping with iOS 9, for example, the language took syntax used for creating loops and changed its purpose entirely. Suddenly, code that was only a few months old stopped working.

Swift presents challenges, but I also think you'll find that programming in Swift is *fun* (yes, really) and intuitive.

When creating Swift and the iOS development platform, Apple considered the entire application lifecycle. From the interface design tools, to the code that makes it function, to the presentation to the user, everything is integrated and works together seamlessly. As a developer, does this mean that there are rules to follow? Absolutely. But, by following these rules, you can create applications that are interactive works of art for your users to love—not software they will load and forget.

Through the App Store, Apple has created the ultimate digital distribution system for iOS applications. Programmers of any age or affiliation can submit their applications to the App Store for just the cost of a modest yearly Developer Membership fee. Games, utilities, and full-feature applications have been built for everything from pre-K education to retirement living. No matter what the content, with a user base as large as the iPhone, iPod touch, and iPad, an audience exists.

My hope is that this book brings iOS development to a new generation of developers. *Sams Teach Yourself iOS 9 Application Development in 24 Hours* provides a clear and natural progression of skills development, from installing developer tools and registering your device with Apple, to debugging an application before submitting it to the App Store. It's everything you need to get started, in 24 one-hour lessons.

Who Can Become an iOS Developer?

If you have an interest in learning, time to invest in exploring and practicing with Apple's developer tools, and an Intel Macintosh computer running Yosemite, El Capitan, or later, you have everything you need to begin creating software for iOS. Starting with Xcode 7, Apple even lets you run your applications on your own devices (no developer membership required)!

Developing an app won't happen overnight, but with dedication and practice, you can be writing your first applications in a matter of days. The more time you spend working with the Apple developer tools, the more opportunities you'll discover for creating new and exciting projects.

You should approach iOS application development as creating software that *you* want to use, not what you think others want. If you're solely interested in getting rich quick, you're likely to be disappointed. (The App Store is a crowded marketplace—albeit one with a lot of room—and competition for top sales is fierce.) However, if you focus on building useful and unique apps, you're much more likely to find an appreciative audience.

Who Should Use This Book?

This book targets individuals who are new to development for iOS and have experience using the Macintosh platform. No previous experience with Swift, Cocoa, or the Apple developer tools is required. Of course, if you do have development experience, some of the tools and techniques may be easier to master, but the author does not assume that you've coded before.

That said, some things are expected of you, the reader. Specifically, you must be willing to invest in the learning process. If you just read each hour's lesson without working through the tutorials, you will likely miss some fundamental concepts. In addition, you need to spend time reading the Apple developer documentation and researching the topics presented in this book. A vast amount of information on iOS development is available, but only limited space is available in this book. Therefore, this book covers what you need to forge your own path forward.

What Is (and Isn’t) in This Book?

The material in this book specifically targets iOS release 9.1 and later on Xcode 7.1 and later. Much of what you’ll learn is common to all the iOS releases, but this book also covers several important areas that have only come about in recent iOS releases, such as gesture recognizers, embedded video playback with 3D Touch, AirPlay, Core Image, social networking, multitasking, universal (iPhone/iPad) applications, auto layout, size classes, and more!

Unfortunately, this is not a complete reference for the iOS application programming interfaces (APIs), nor do I explicitly cover AppleTV or Apple Watch development; some topics just require much more space than this book allows. That said, this book should provide ample exposure to the tools and techniques that any iOS developer needs to be successful. In addition, the Apple developer documentation is available directly within the free tools you install in Hour 1, “Preparing Your System and iDevice for Development.” In many hours, you’ll find a section titled “Further Exploration.” This identifies additional related topics of interest. Again, a willingness to explore is an important quality in becoming a successful developer.

Each coding lesson is accompanied by project files that include everything you need to compile and test an example or, preferably, follow along and build the application yourself. Be sure to download the project files from this book’s website at http://teachyourselfios.com. If you have issues with any projects, view the posts on this site to see whether a solution has been identified.

In addition to the support website, you can follow along on Twitter! Search for #iOSIn24 on Twitter to receive official updates and tweets from other readers. Use the hashtag #iOSIn24 in your tweets to join the conversation. To send me messages via Twitter, begin each tweet with @johnemeryray.

This page intentionally left blank

This page intentionally left blank

HOUR 5

Exploring Interface Builder

What You'll Learn in This Hour:

- ▶ Where Xcode's Interface Builder fits in the development process
- ▶ The role of storyboards and scenes
- ▶ How to build a user interface using the Object Library
- ▶ Common attributes that can be used to customize interface elements
- ▶ Ways to make your interface accessible to the visually impaired
- ▶ How to link interfaces to code with outlets and actions

Over the past few hours, you've become familiar with the core iOS technologies, Xcode projects, and the iOS Simulator. Although these are certainly important skills for becoming a successful developer, there's nothing quite like laying out your first iOS application interface and watching it come to life in your hands.

This hour introduces you to Interface Builder: the remarkable user interface (UI) editor integrated into Xcode. Interface Builder provides a visual approach to application interface design that is fun, intuitive, and deceptively powerful.

Understanding Interface Builder

Let's get it out of the way up front: Yes, Interface Builder (or IB for short) does help you create interfaces for your applications, but it isn't just a drawing tool for graphical user interfaces (GUIs); it helps you symbolically build application functionality without writing code. This translates to fewer bugs, shorter development time, and easier-to-maintain projects.

If you read through Apple's developer documentation, you'll see IB referred to as an *editor* within Xcode. This is a bit of an oversimplification of a tool that previously existed as a standalone application in the Apple Developer Suite. An understanding of IB and its use is as fundamentally important to iOS development as Swift. Without IB, creating the most basic interactive applications would be an exercise in frustration.

This hour focuses on navigating IB and will be key to your success in the rest of the book. In Hour 6, “Model-View-Controller Application Design,” you combine what you’ve learned about Xcode projects, the code editor, IB, and the iOS Simulator for the first time. So, stay alert and keep reading.

The IB Approach

Using Xcode and the Cocoa toolset, you can program iOS interfaces by hand—instantiating interface objects, defining where they appear on the screen, setting any attributes for the object, and finally, making them visible. For example, in Hour 2, “Introduction to Xcode and the iOS Simulator,” you entered this listing into Xcode to make your iOS device display the text `Hello Xcode` on the screen:

```
var myMessage: UILabel
myMessage=UILabel(frame:CGRectMake(30.0,50.0,300.0,50.0))
myMessage.font=UIFont.systemFontOfSize(48.0)
myMessage.text="Hello Xcode"
myMessage.textColor=UIColor(patternImage: UIImage(named:"Background")!)
view.addSubview(myMessage)
```

Imagine how long it would take to build interfaces with text, buttons, images, and dozens of other controls, and think of all the code you would need to wade through just to make small changes.

Over the years, there have been many different approaches to graphical interface builders. One of the most common implementations is to enable the user to “draw” an interface but, behind the scenes, create all the code that generates that interface. Any tweaks require the code to be edited by hand (hardly an acceptable situation).

Another tactic is to maintain the interface definition symbolically but attach the code that implements functionality directly to interface elements. This, unfortunately, means that if you want to change your interface or swap functionality from one UI element to another, you have to move the code as well.

IB works differently. Instead of autogenerating interface code or tying source listings directly to interface elements, IB builds live objects that connect to your application code through simple links called *connections*. Want to change how a feature of your app is triggered? Just change the connection. As you’ll learn a bit later this hour, changing how your application works with the objects you create in IB is, quite literally, a matter of connecting or reconnecting the dots as you see fit.

The Anatomy of an IB Storyboard

Your work in IB results in an XML file called a *storyboard*, containing a hierarchy of objects for each unique screen that your application is going to display. The objects could be interface elements—buttons, toggle switches, and so forth—but might also be other noninterface objects that you will need to use. The collection of objects for a specific display is called a *scene*. Storyboards can hold as many scenes as you need, and even link them together visually via *segues*.

For example, a simple recipe application might have one scene that consists of a list of recipes the user can choose from. A second scene may contain the details for making a selected recipe. The recipe list could be set to segue to the detail view with a fancy fade-out/fade-in effect when the name of a recipe is touched. All of this functionality can be described visually in an application's storyboard file.

Storyboards aren't just about cool visuals, however. They also help you create usable objects without having to allocate or initialize them manually. When a scene in a storyboard file is loaded by your application, the objects described in it are instantiated and can be accessed by your code.

NOTE

Instantiation, just as a quick refresher, is the process of creating an instance of an object that you can work with in your program. An instantiated object gains all the functionality described by its class. Buttons, for example, automatically highlight when clicked, content views scroll, and so on.

The Storyboard Document Outline

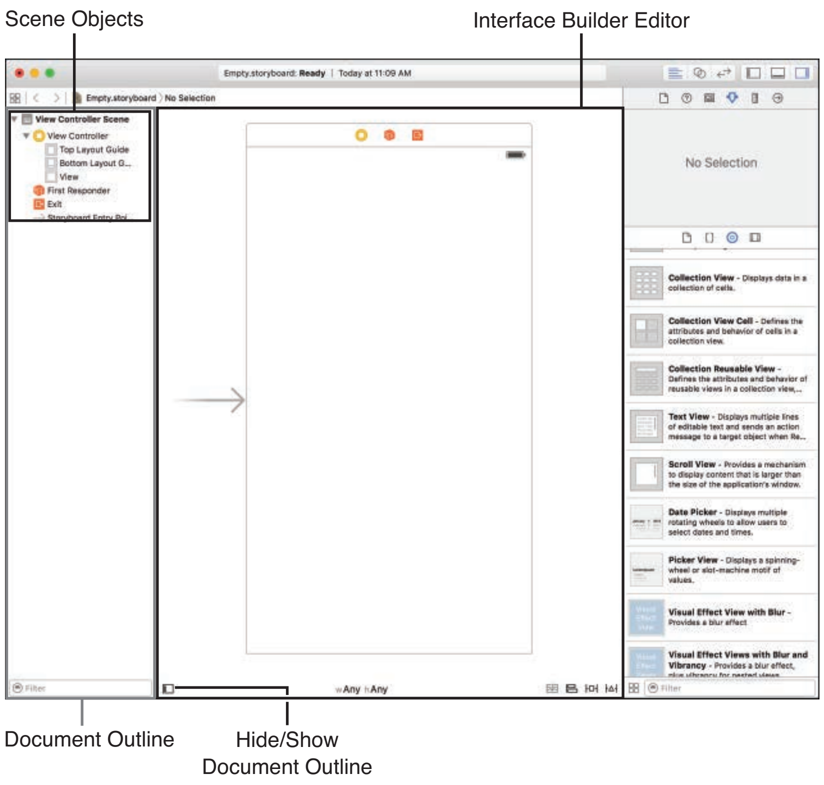
What do storyboard files look like in IB? Open the Hour 5 Projects folder and double-click the file Empty.storyboard to open IB and display a barebones storyboard file with a single scene. You will need to choose View, Show Toolbar from the menu bar to make the workspace look like a normal project. Once the file is open and the toolbar visible, we can get our bearings.

The contents of the file are shown visually in the IB editor area, and hierarchically by scene in the document outline area located in the column to the left of the editor area (see Figure 5.1).

TIP

If you do not see the document outline area in your Xcode workspace, choose Editor, Show Document Outline from the menu bar. You can also click the disclosure arrow button in the lower-left corner of the Xcode editor area.

Note that there is only a single scene in the file: View Controller Scene. Single-scene storyboards will be the starting place for much of your interface work in this book because they provide plenty of room for collecting user input and displaying output. We explore multiscene storyboards beginning in Hour 11, "Implementing Multiple Scenes and Popovers."



Collection View - Displays data in a collection of cells.

Collection View Cell - Defines the attributes and behavior of cells in a collection view.

Collection Reusable View - Defines the attributes and behavior of reusable views in a collection view,...

Text View - Displays multiple lines of editable text and sends an action message to a target object when Re...

Scroll View - Provides a mechanism to display content that is larger than the size of the application's window.

Date Picker - Displays multiple rotating wheels to allow users to select dates and times.

Picker View - Displays a spinning-wheel or slot-machine motif of values.

Visual Effect View with Blur - Provides a blur effect.

Visual Effect Views with Blur and Vibrancy - Provides a blur effect,

FIGURE 5.1
A storyboard scene's objects are represented by icons.

Click the arrow in front of View Controller Scene to show its hierarchy (and then expand the View Controller object within it, as well).

Six icons should be visible in the scene: View Controller, View (within View Controller), Top Layout Guide (within View Controller), Bottom Layout Guide (within View Controller), First Responder, and Exit. With the exception of View, these are special icons used to represent unique noninterface objects in our application; these will be present in most of the storyboard scenes that you work with:

- **View Controller:** The View Controller icon denotes the object that loads and interacts with a storyboard scene in your running application. This is the object that effectively instantiates all the other objects described within a scene. You'll learn more about the relationship between UIs and view controllers in Hour 6.

- **Top Layout Guide:** A guide line that marks the "top" of your content area (usually the bottom of the iOS status bar). You can use this guide to keep your UI objects below the portions of the display managed by iOS. This is part of the Auto Layout system that we discuss later this hour, and in depth in Hour 16, "Building Responsive User Interfaces."
- **Bottom Layout Guide:** A guide line that marks the "bottom" of your content area. This is usually the bottom of the view itself. Like the Top Layout Guide, this is used to help position your user interface. Again, this is an Auto Layout tool that we won't really need until much later in the book.
- **View:** The View icon is an instance of the object `UIView` and represents the visual layout that will be loaded by the view controller and displayed on the iOS device's screen. Views are hierarchical in nature. Therefore, as you add controls to your interface, they are contained within the view. You can even add views within views to cluster controls or create visual elements that can be shown or hidden as a group.
- **First Responder:** The First Responder icon stands for the object that the user is currently interacting with. When a user works with an iOS application, multiple objects could potentially respond to the various gestures or keystrokes that the user creates. The first responder is the object currently in control and interacting with the user. A text field that the user is typing into, for example, would be the first responder until the user moves to another field or control.
- **Exit:** The Exit icon serves a very specific purpose that will come into play only in multi-scene applications. When you are creating an app that moves the user between a series of screens, the Exit icon provides a visual means of jumping back to a previous screen. If you have built five scenes that link from one to another and you want to quickly return to the first scene from the fifth, you'll link from the fifth scene to the first scene's Exit icon. We test this out in Hour 11.
- **Storyboard Entry Point:** This icon is just an indicator that this scene is what your application is going to display when it launches. Storyboards can have many scenes, but they need to *start* somewhere. That somewhere is the entry point.

NOTE

The storyboard shown in this example is about as "vanilla" as you can get. In larger applications with multiple scenes, you may want to either name your view controller class to better describe *what* it is actually controlling or set a descriptive label, such as Recipe Listing.

Using unique view controller names/labels also benefits the naming of scenes. IB automatically sets scene names to the name of the view controller or its label (if one is set) plus the suffix *scene*. If you label your view controller as Recipe Listing, for example, the scene name changes to Recipe

Listing Scene. We'll worry about multiple scenes later in the book; for now, our projects will contain a generic class called View Controller that will be in charge of interacting with our single view controller scene.

As you build your UIs, the list of objects within your scenes will grow accordingly. Some UIs may consist of dozens of different objects, leading to rather busy and complex scenes, as demonstrated in Figure 5.2.

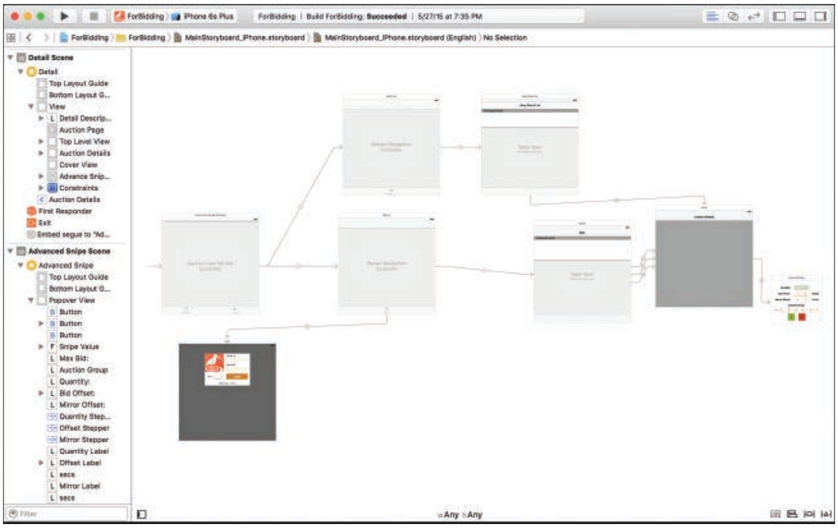


FIGURE 5.2
Storyboard scenes and their associated views can grow quite large and complex.

You can collapse or expand your hierarchy of views within the document outline area to help manage the information overload that you are bound to experience as your applications become more advanced.

NOTE

At its most basic level, a view (`UIView`) is a rectangular region that can contain content and respond to user events (touches and so forth). All the controls (buttons, fields, and so on) that you'll add to a view are, in fact, subclasses of `UIView`. This isn't necessarily something you need to be worried about, except that you'll be encountering documentation that refers to buttons and other interface elements referred to as *subviews* and the views that contain them as *superviews*.

Just keep in mind that pretty much everything you see onscreen can be considered a view and the terminology will seem a little less alien.

Working with the Document Outline Objects

The document outline area shows icons for objects in your application, but what good are they? Aside from presenting a nice list, do they provide any functionality?

Absolutely! Each icon gives you a visual means of referring to the objects they represent. You interact with the icons by dragging to and from them to create the connections that drive your application's features.

Consider an onscreen control, such as a button, that needs to trigger an action in your code. By dragging from the button to the View Controller icon, you can create a connection from the GUI element to a method that you want it to activate. You can even drag from certain objects directly to your code, quickly inserting a variable or method that will interact with that object.

Xcode provides developers with a great deal of flexibility when working with objects in IB. You can interact with the actual UI elements in the IB editor, or with the icons that represent them in the document outline area. In addition, any object that isn't directly visible in the UI (such as the first responder and view controller objects) can be found in an icon bar directly above the UI design in the editor. This is known as the scene dock, and is visible in Figure 5.3.

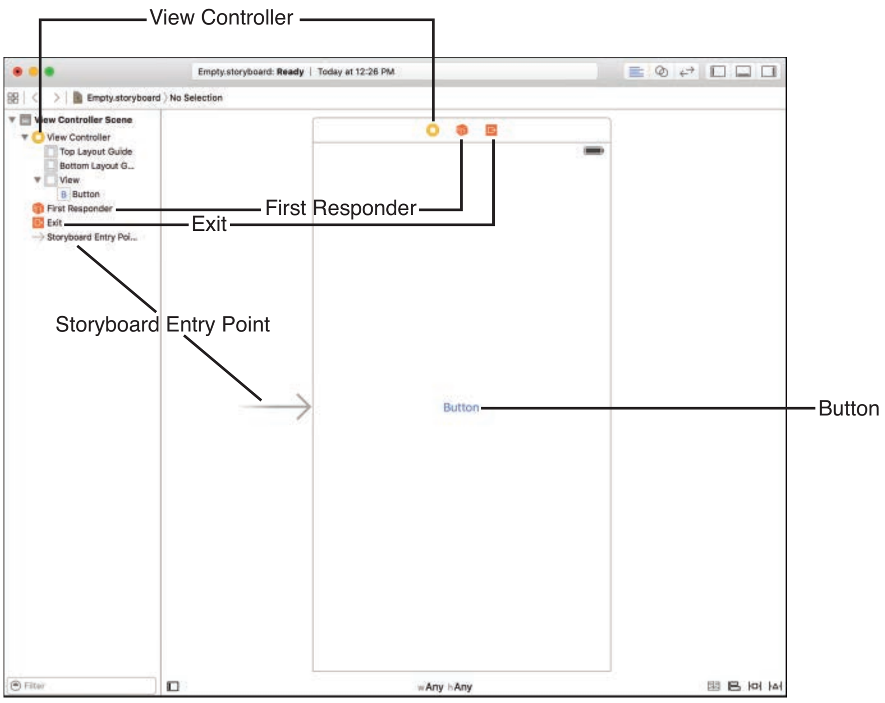


FIGURE 5.3
You will interact with objects either in the editor or in the document outline area.

NOTE

If the scene dock above your view does not show any icons and is displaying the text *View Controller* instead, just click it. The dock defaults to the name of a scene's view controller until it is selected.

We go through a hands-on example later this hour so that you can get a feel for how interacting with and connecting objects works. Before we do that, however, let's look at how you go about turning a blank view into an interface masterpiece.

Creating User Interfaces

In Figures 5.1 and 5.2, you've seen an empty view and a fully fleshed-out interface. Now, how do we get from one to the other? In this section, we explore how interfaces are created with IB. In other words, it's time for the fun stuff.

If you haven't already, open the Empty.storyboard file included in this hour's Projects folder. Make sure the document outline area is visible and that the view can be seen in the editor; you're ready to start designing an interface.

What's the "Default" iPhone?

When you're editing iPhone interfaces and creating new projects in Xcode, you'll be working, by default, with a "generic" screen that doesn't match any shipping iPhone or iPad. This is because Apple wants you to build interfaces that work for any device. I want you to do the same, but I want you to learn development basics first! We cover how to create interfaces that properly resize for any device in Hour 16. For the majority of projects in this book, we'll be setting a default simulated device so that we're working on a canvas that is a bit more familiar than a nonspecific rectangle. I'll use a 4.7-inch iPhone (6/6s) as my default screen for layouts—as you'll see when we start building applications from scratch.

Again, we *do* cover how to accommodate any size you want later in the book. Our focus now is on getting started building apps rather than fine-grained interface layout.

The Object Library

Everything that you add to a view, from buttons and images to web content, comes from the Object Library. You can view the library by choosing View, Utilities, Show Object Library from the menu bar (Control-Option-Command-3). If it isn't already visible, the utility area of the Xcode interface opens, and Object Library is displayed in the lower right.

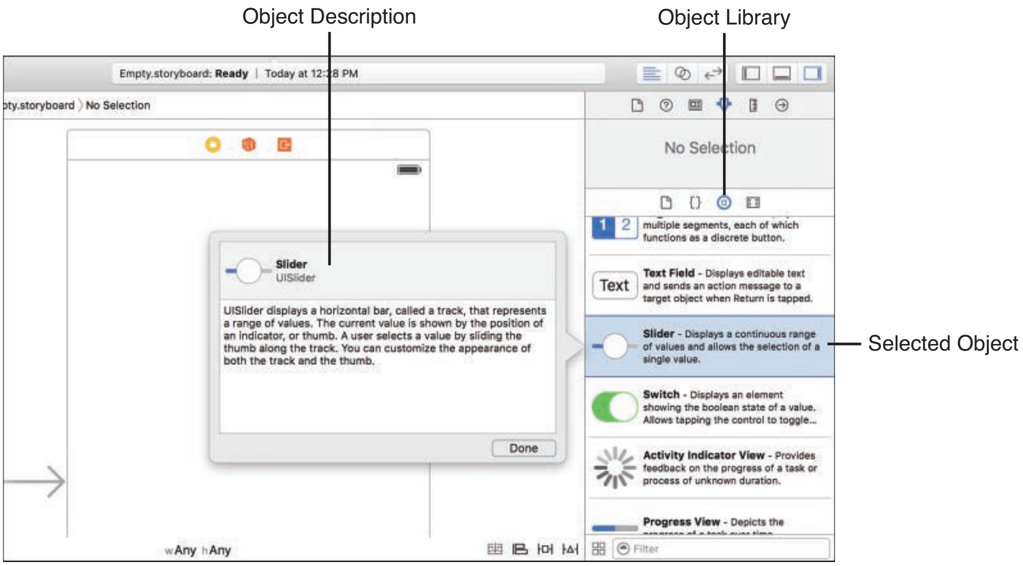
CAUTION

Libraries, Libraries, Everywhere!

Xcode has more than one library. The Object Library contains the UI elements you'll be adding in IB, but there are also File Template, Code Snippet, and Media libraries that can be activated by clicking the icons immediately above the Library area.

If you find yourself staring at a library that doesn't seem to show what you're expecting, click the icon of a square surrounded by a circle above the library or reselect the Object Library from the View, Utilities menu to make sure that you're in the right place.

When you click and hover over an element in the library, a popover is displayed with a description of how the object can be used in the interface, as shown in Figure 5.4. This provides a convenient way of exploring your UI options without having to open the Xcode documentation.



UISlider displays a horizontal bar, called a track, that represents a range of values. The current value is shown by the position of an indicator, or thumb. A user selects a value by sliding the thumb along the track. You can customize the appearance of both the track and the thumb.

multiple segments, each of which functions as a discrete button.

Text Field - Displays editable text and sends an action message to a target object when Return is tapped.

Slider - Displays a continuous range of values and allows the selection of a single value.

Switch - Displays an element showing the boolean state of a value. Allows tapping the control to toggle...

Activity Indicator View - Provides feedback on the progress of a task or process of unknown duration.

Progress View - Depicts the

FIGURE 5.4
The library contains a palette of objects that can be added to your views.

Using view button (four squares) at the bottom-left of the library, you can switch between list and icon views of the available objects. If you know the name of an object but can't locate it in the list, use the filter field below the library to quickly find it.

Adding Objects to a View

To add an object to a view, just click and drag from the library to the view. For example, find the label object (`UILabel`) in the Object Library and drag it into the center of the view in the editor.

The label should appear in your view and read `Label`. Double-click the label and type **Hello**. The text will update, as shown in Figure 5.5, just as you would expect.

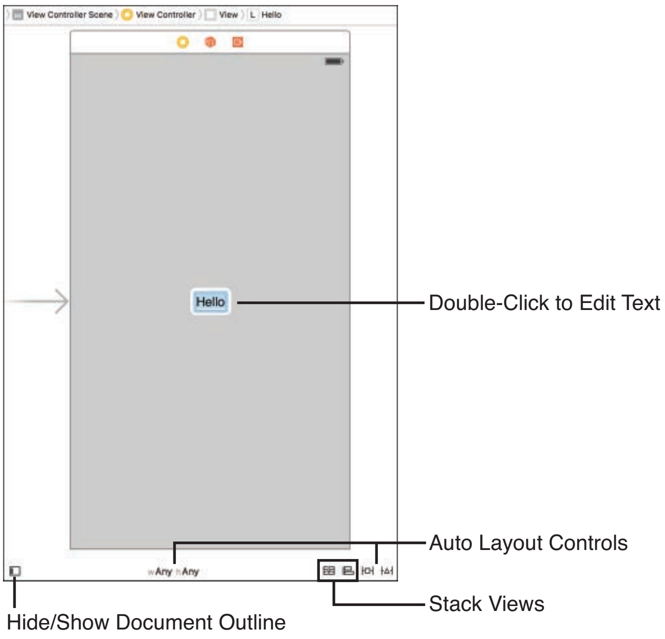


FIGURE 5.5
If an object contains text, in many cases, just double-click to edit it.

With that simple action, you've almost entirely replicated the functionality implemented by the code fragment presented earlier in the lesson. Try dragging other objects from the Object Library into the view (buttons, text fields, and so on). With few exceptions, the objects should appear and behave just the way you'd expect.

To remove an object from the view, click to select it, and then press the Delete key. You may also use the options under the Edit menu to copy and paste between views or duplicate an element several times within a view.

NOTE

Notice the tools along the bottom of the editing area? These are largely related to view positioning and Auto Layout (more on that shortly), with one exception:

The button in the lower left hides and shows the document outline, giving you more space to work. You'll want to make use of this button *a lot*, especially if you're working on a laptop.

The middle set of menus (typically labeled “w Any” and “h Any”) are used when designing interfaces to fit a variety of devices, while the buttons on the right are used to insert stack views and reveal menus for managing Auto Layout Alignment, Pinning, and Constraint issues.

Working with the IB Editing Tools

Instead of having you rely on your visual acuity to position objects in a view, Apple has included some useful tools for fine-tuning your interface design. If you’ve ever used a drawing program like OmniGraffle or Adobe Illustrator, you’ll find many of these familiar.

Guides

As you drag objects in a view, you’ll notice guides (shown in Figure 5.6) appearing to help with the positioning. These blue, dotted lines are displayed to align objects along the margins of the view, to the centers of other objects in the view, and to the baseline of the fonts used in the labels and object titles.

As an added bonus, guides automatically appear to indicate the approximate spacing requirements of Apple’s interface guidelines. If you’re not sure why it’s showing you a particular margin guide, it’s likely that your object is in a position that IB considers “appropriate” for something of that type and size.

TIP

You can manually add your own guides by choosing Editor, Guides, Add Horizontal Guide or by choosing Editor, Guides, Add Vertical Guide. You can position manually added guides anywhere in your view by dragging them; they will appear orange in color. When you drag an object close to the guide, the guide highlights in blue to show when they are aligned.

TIP

To fine-tune an object’s position within a view, select it, and then use the arrow keys to position it left, right, up, or down, 1 point at a time. You can also zoom in and out of a view by pinching on a Magic Trackpad, or Control-clicking (or right-clicking) in the Interface Builder editor view and choosing an appropriate zoom level from the menu that appears.

Selection Handles

In addition to the guides, most objects include selection handles to stretch an object horizontally, vertically, or both. Using the small boxes that appear alongside an object when it is selected, just click and drag to change its size, as demonstrated using a text field in Figure 5.7.

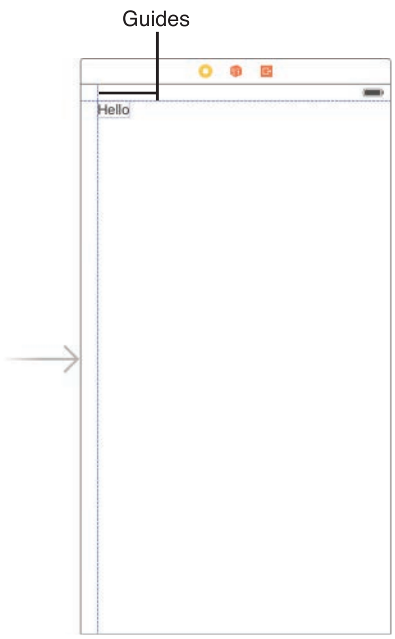


FIGURE 5.6
Guides help position your objects within a view.

Note that some objects constrain how you can resize them; this preserves a level of consistency within iOS application interfaces.

TIP

In busy interfaces, it can be difficult to figure out what object you've selected just by the appearance of selection handles. If you'd prefer Xcode also darken the selected object, choose Editor, Canvas, Show Selection Highlights.

Arrangement and Alignment

When you're working with UI objects in Interface Builder, you'll likely start to feel like you're working in a drawing program. Two commands common in object-based drawing applications are Arrange (where you can position objects in front of or behind one another) and Align (where you can make misplaced objects line up). You'll find both of these options also exist in Interface Builder.

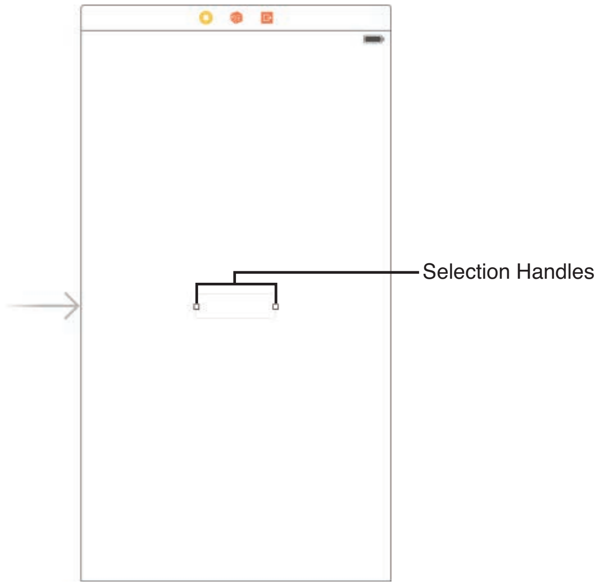


FIGURE 5.7
Use the selection handles around the perimeter of an object to change its size.

To arrange objects, you can choose from the Editor, Arrange menu. Use the Arrange selections (Send to Back, Send Forward, and so on) to move UI elements behind or in front of other elements.

To quickly align several objects within a view, select them by clicking and dragging a selection rectangle around them or by holding down the Shift key and then choosing Editor, Align and an appropriate alignment type from the menu.

For example, try dragging several buttons into your view, placing them in a variety of different positions. To align them based on their horizontal center (a line that runs vertically through each button's center), select the buttons, and then choose Editor, Align, Horizontal Centers. Figure 5.8 shows the before and after results.

The Size Inspector

Another tool that you may want to use for controlling your design is the Size Inspector. IB has a number of inspectors for examining the attributes of an object. As the name implies, the Size Inspector provides information about sizes, but also position and alignment. To open the Size Inspector, first select the object (or objects) that you want to work with, and then click the ruler icon at the top of the utility area in Xcode. Alternatively, choose View, Utilities, Show Size Inspector or press Option-Command-5 (see Figure 5.9).

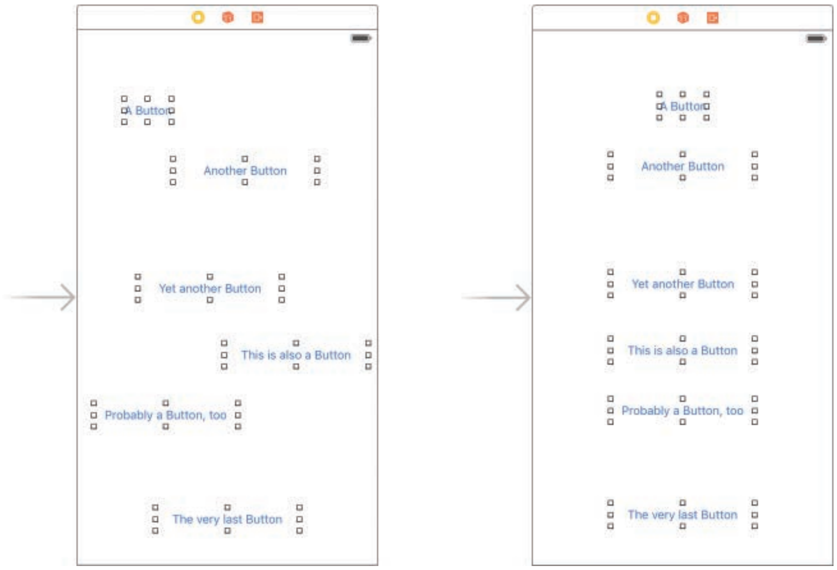
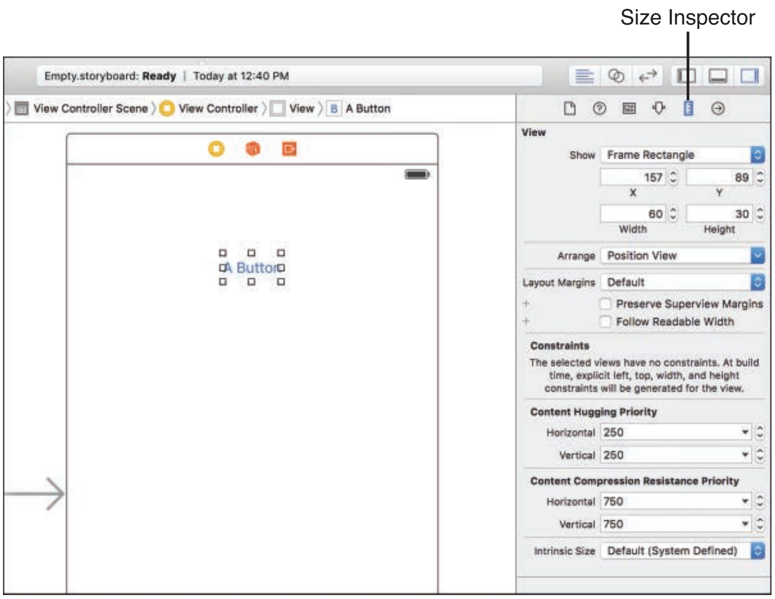


FIGURE 5.8
Use the Align menu to quickly align a group of items to an edge or center.


Size Inspector
Empty.storyboard: Ready | Today at 12:40 PM
View Controller Scene > View Controller > View > A Button
A Button

View

Show: Frame Rectangle

X: 157 Y: 89

Width: 60 Height: 30

Arrange: Position View

Layout Margins: Default

Preserve Superview Margins

Follow Readable Width

Constraints

The selected views have no constraints. At build time, explicit left, top, width, and height constraints will be generated for the view.

Content Hugging Priority

Horizontal: 250

Vertical: 250

Content Compression Resistance Priority

Horizontal: 750

Vertical: 750

Intrinsic Size: Default (System Defined)

FIGURE 5.9
The Size Inspector enables you to adjust the size and position of one or more objects.

Using the fields at the top of the inspector, you can view or change the size and position of the object by changing the coordinates in the Height/Width and X/Y fields.

NOTE

At the top of Size Inspector's View settings, you'll often see a drop-down menu where you can choose between Frame Rectangle and Alignment Rectangle. These two settings will usually be similar, but there is a slight difference. The frame values represent the exact area a "raw" object occupies onscreen, whereas the alignment values take into account spacing around the object for drop shadows and the like.

The Arrange drop-down menu gives you quick layout arrangements for the selected object (or objects). Using this menu, you can center the object, align it with other objects, or size it to take up the width or height of the view that holds it. You can do all of this with the main Interface Builder tools as well; this menu is just another place to make quick tweaks to your layout.

Below the Arrange menu are options for controlling layout margins. Layout margins are the amount of space around an object in your design. By default, layout margins are 8 points on the top, bottom, left, and right of each object. You can explicitly set layout margins to include more (or less) space around an object. These margins, however, are only used when using Auto Layout. Bear with me, I'm about to tell you what Auto Layout is.

Notice that the Size Inspector includes a section at the bottom labeled Constraints. Constraints are part of the Auto Layout system that we will be using to create resizable user interfaces in Hour 16. Because you're likely to run into a few references to Auto Layout before we get there, let's take a few minutes to get an idea of what this beast is.

TIP

Hold down the Option key after selecting an object in IB. As you move your mouse around, you'll see the distance between the selected object and other objects that you point to.

The Auto Layout System

While the guides, Size Inspector, and other tools are helpful for laying out interfaces—even interfaces that can adapt to view changes—iOS applications can take advantage of a new powerful tool for managing layouts: the Auto Layout system. Auto Layouts are enabled by default on new projects and make it possible to write applications that adapt to a number of different screen sizes and orientations without needing to modify a single line of code. Do you want to write software to take advantage of all the available iOS device screen sizes? How about layouts that rearrange themselves when you move from portrait to landscape orientations? You'll want Auto Layouts!

Understanding Constraints

Auto Layout works by building a series of *constraints* for your onscreen objects. The constraints define distances between objects and how flexible these relationships are.

For example, open the Constraints.storyboard file included in the Projects folder. This storyboard contains a view with a single label positioned in the upper center. Expand the View Controller scene in the document outline so that you can see all the objects it contains. Notice that at the same level in the hierarchy as the label, a Constraints object is showing up, as shown in Figure 5.10.

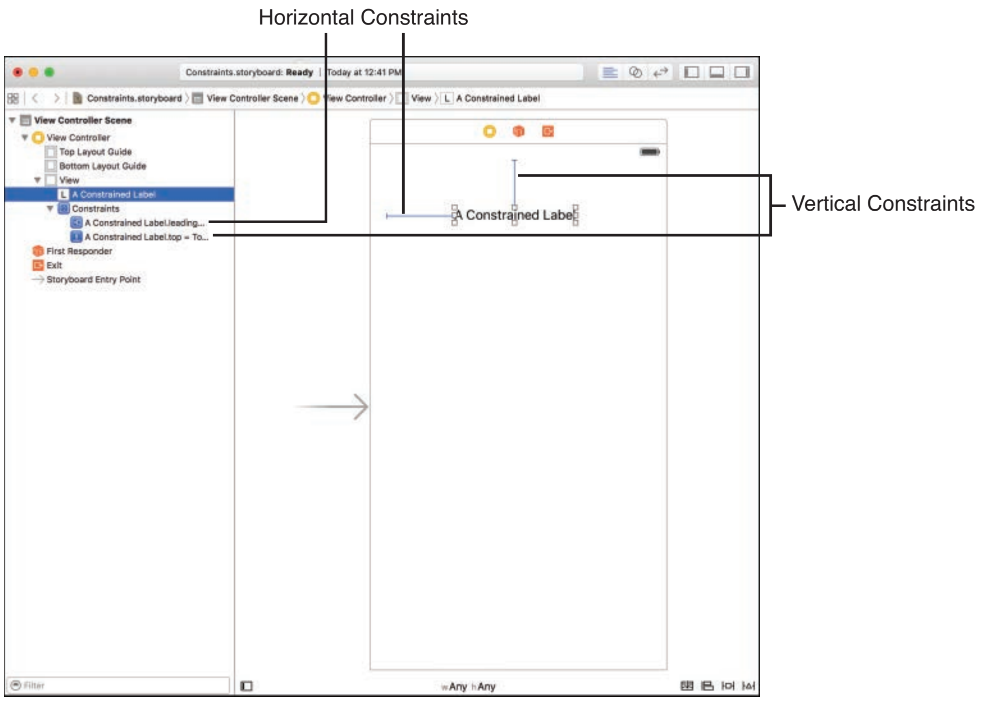


FIGURE 5.10
The Constraints object represents the positioning relationships within a view.

Within the Constraints object are two constraints: horizontal space and vertical space constraints. The horizontal constraint states that the left or right side of the label will be a certain number of points from the left or right edge of the view. These are known as *leading* and *trailing* constraints, respectively. A vertical constraint is the distance from the top or bottom of the view to the top or bottom of the label. Intuitively, these are called the *top* and *bottom* constraints.

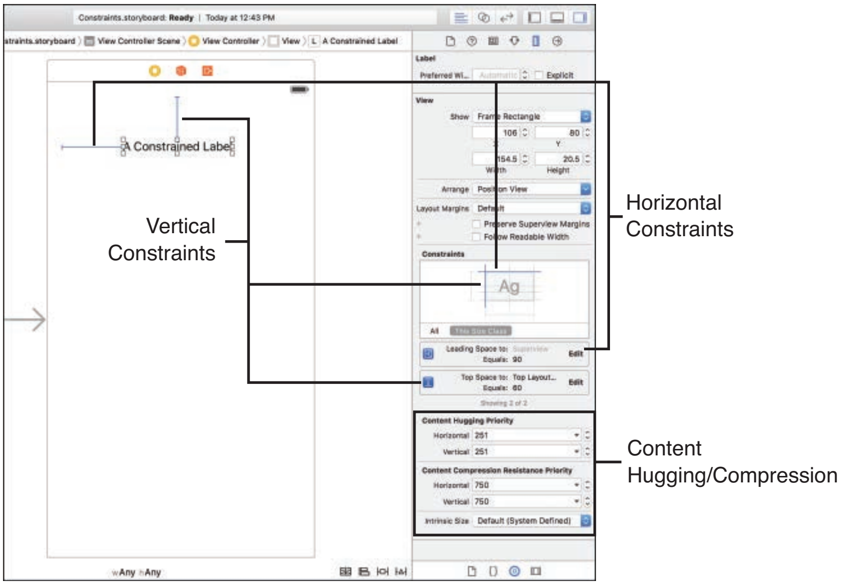
Constraints, however, are more than just entries that tie an object to the view it is within. They can be flexible, ensuring that an object maintains *at least* or *at most* a certain distance from another object, or even that two objects, when resized, maintain the same distance between one another.

Constraints that set a specific size or distance between objects are called *pinning*. The flexibility (or inflexibility of a constraint) is managed by configuring a *relationship*.

Content Hugging and Content Compression Resistance

Now that you're viewing an object with constraints, the Size Inspector updates to show a bit more information than we saw earlier. Click the label in the Constraints storyboard file and make sure that the Size Inspector is visible (Option-Command-5), as shown in Figure 5.11.

The constraints affecting the label itself are shown near the bottom of the Size Inspector information, but there are additional settings now visible for Content Hugging (how friendly!) and Content Compression.


Constraints.storyboard: Ready | Today at 12:43 PM
straints.storyboard 〉 View Controller Scene 〉 View Controller 〉 View 〉 A Constrained Label
A Constrained Label
Vertical Constraints
Horizontal Constraints
Content Hugging/Compression
wAny hAny

Label
Preferred Wi... Automatic ☐ Explicit

View
Show: Frame Rectangle
X: 106 Y: 80
Width: 154.5 Height: 20.5
Arrange: Position View
Layout Margins: Default
☐ Preserve Superview Margins
☐ Follow Readable Width

Constraints
Ag
All | This Size Class
Leading Space to: Superview Equals: 90 Edit
Top Space to: Top Layout... Equals: 60 Edit
Showing 2 of 2

Content Hugging Priority
Horizontal: 251
Vertical: 251

Content Compression Resistance Priority
Horizontal: 750
Vertical: 750

Intrinsic Size: Default (System Defined)

FIGURE 5.11
The Size Inspector shows information about how Auto Layout will affect an object.

These settings control how closely the sides of an object "hug" the content in the object and how much the content can be compressed or clipped. An object that can expand horizontally but not vertically would set horizontal hugging as a low priority and vertical hugging as a very high priority. Similarly, if the content of the object (say a label) should not be compressed or clipped at all, the content compression resistance settings for both horizontal and vertical compression could be set to a very high priority.

NOTE

Another Auto Layout option you'll encounter in the Size Inspector is an Intrinsic Size setting. The intrinsic size is just the size of the object as determined by your layout. If you're creating custom views, IB will have no idea how large the view actually is, so you'll need to use the Intrinsic Size drop-down to set a "placeholder" size. This is a relatively advanced topic and not something you're likely to encounter in day-to-day development.

I Miss the Old Autosizing Features! Boo Hoo

If you prefer to forego the new Auto Layout tools in Xcode, you can revert your storyboard to the "old" layout approach by first selecting the File Inspector (Option-Command-1) while viewing your storyboard. Next, *uncheck* the Use Autolayout check box within the Interface Builder Document section of settings. Everything will operate exactly as it did prior to Xcode 4.5.

Hour 16 focuses on the new Auto Layout tools, however, so I suggest leaving Auto Layout active.

Auto Layout and Automatic Constraints

When Apple introduced Auto Layout in Xcode 4.5, suddenly constraints were *everywhere*. Any label, button, or object you positioned in your user interface immediately had constraints appear. Each object requires at least two constraints (horizontal positioning and vertical positioning) to determine its location in the interface and (often) two to determine height and width. Add 10 objects, and suddenly you've got at least 20 to 40 constraints (and 40 blue lines all over your view).

In later releases of Xcode, Apple made this blue crosshatched nightmare go away. Now, when you position objects in your UI, you won't see any constraints until you manually add them. That doesn't mean they aren't there; Xcode automatically adds constraints when you build your project. For beginners like us, this is perfect. We can lay out our UIs and not worry about constraints until we absolutely need to do something "clever" with object positioning. As already mentioned, we'll "get clever" with Auto Layout in Hour 16, but for the time being, you can pretend it doesn't even exist.

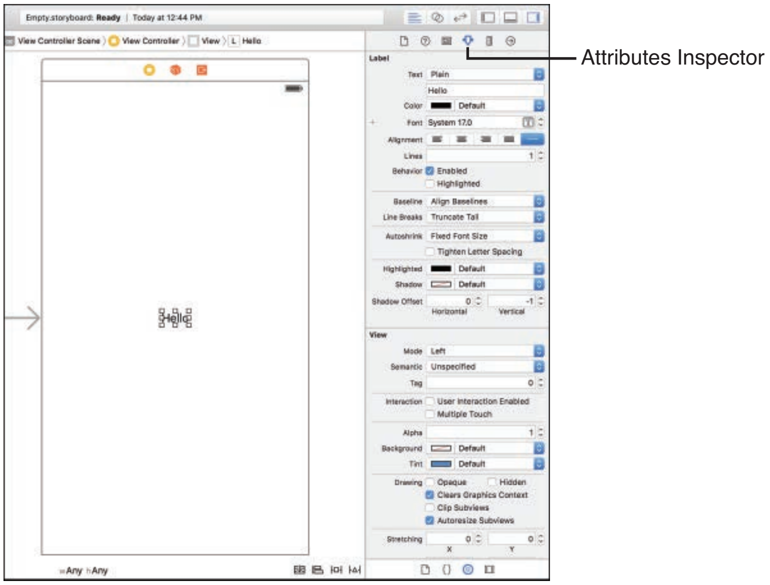
Customizing the Interface Appearance

How your interface appears to the end user isn't just a combination of control sizes, positions, and constraints. For many kinds of objects, literally dozens of different attributes can be adjusted. Although you could certainly configure things such as colors and fonts in your code, it's easier to just use the tools included in IB.

Using the Attributes Inspector

The most common place you'll tweak the way your interface objects appear is through the Attributes Inspector, available by clicking the slider icon at the top of the utility area. You can also choose View, Utilities, Show Attributes Inspector (Option-Command-4) if the utility area isn't currently visible. Let's run through a quick example to see how this works.

Turn back to the Empty.storyboard file with the label you've added (or just use the Constraints. storyboard label). Select the label, and then open the Attributes Inspector, shown in Figure 5.12.


Empty.storyboard: Ready | Today at 12:44 PM
View Controller Scene > View Controller > View > L Hello
Hello
wAny hAny
Attributes Inspector

Label
Text: Plain
Hello
Color: Default
Font: System 17.0
Alignment
Lines: 1
Behavior: [✓ Enabled] Highlighted
Baseline: Align Baselines
Line Breaks: Truncate Tail
Autoshrink: Fixed Font Size
Tighten Letter Spacing
Highlighted: Default
Shadow: Default
Shadow Offset: 0 Horizontal, -1 Vertical

View
Mode: Left
Semantic: Unspecified
Tag: 0
Interaction: User Interaction Enabled, Multiple Touch
Alpha: 1
Background: Default
Tint: Default
Drawing: Opaque, Hidden, [✓ Clears Graphics Context], Clip Subviews, [✓ Autoresize Subviews]
Stretching: 0 X, 0 Y

FIGURE 5.12
To change how an object looks and behaves, select it, and then open the Attributes Inspector.

The top portion of the Attributes Inspector contains attributes for the specific object. In the case of the text object, this includes settings such as font, size, color, and alignment (everything you'd expect to find for editing text).

In the bottom portion of the inspector are additional inherited attributes. Remember that onscreen elements are a subclass of a view. Therefore, all the standard view attributes are also available for the object and for your tinkering enjoyment. In many cases, you'll want to leave these alone, but settings such as background and transparency can come in handy.

TIP

Don't get hung up on trying to memorize every attribute for every control now. I cover interesting and important attributes when they are needed throughout the book.

Feel free to explore the many different options available in the Attributes Inspector to see what can be configured for different types of objects. There is a surprising amount of flexibility to be found within the tool.

NOTE

The attributes you change in IB are simply properties of the object's class. To help identify what an attribute does, use the documentation tool in Xcode to look up the object's class and review the descriptions of its properties.

Setting Accessibility Attributes

For many years, the "appearance" of an interface meant just how it looks visually. Today, the technology is available for an interface to vocally describe itself to the visually impaired. iOS includes Apple's screen-reader technology: Voiceover. Voiceover combines speech synthesis with a customized interface to aid users in navigating applications.

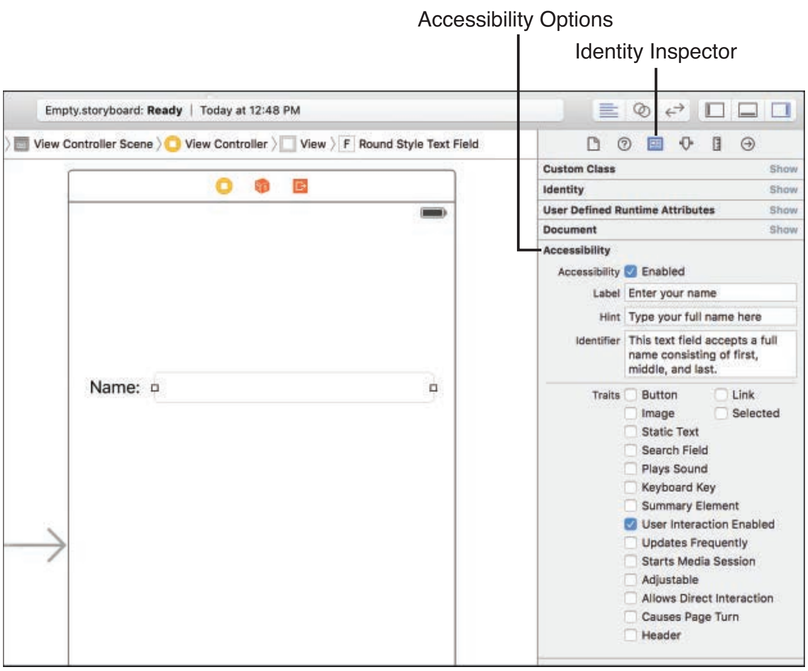
Using Voiceover, users can touch interface elements and hear a short description of what they do and how they can be used. Although you gain much of this functionality "for free" (the iOS Voiceover software will read button labels, for example), you can provide additional assistance by configuring the accessibility attributes in IB.

To access the Accessibility settings, you need to open the Identity Inspector by clicking the window icon at the top of the utility area. You can also choose View, Utilities, Show Identity Inspector or press Option-Command-3. The Accessibility options have their own section within the Identity Inspector, as shown in Figure 5.13.

You can configure four sets of attributes within this area:

- **Accessibility:** If enabled, the object is considered accessible. If you create any custom controls that must be seen to be used, this setting should be disabled.
- **Label:** A simple word or two that serves as the label for an item. A text field that collects the user's name might use "your name," for example.
- **Hint:** A short description, if needed, on how to use the control. This is needed only if the label doesn't provide enough information on its own.
- **Identifier:** Similar to the Label attribute, the Identifier should contain a more detailed description of the control (for example, "A text field for your first and last name").

- **Traits:** This set of check boxes is used to describe the features of the object—what it does and what its current state is.


Accessibility Options
Identity Inspector
Empty.storyboard: Ready | Today at 12:48 PM
View Controller Scene > View Controller > View > F Round Style Text Field
Custom Class Show
Identity Show
User Defined Runtime Attributes Show
Document Show
Name:

Accessibility

Accessibility ✓ Enabled

Label: Enter your name

Hint: Type your full name here

Identifier: This text field accepts a full name consisting of first, middle, and last.

Traits:
- ☐ Button
- ☐ Link
- ☐ Image
- ☐ Selected
- ☐ Static Text
- ☐ Search Field
- ☐ Plays Sound
- ☐ Keyboard Key
- ☐ Summary Element
- ✓ User Interaction Enabled
- ☐ Updates Frequently
- ☐ Starts Media Session
- ☐ Adjustable
- ☐ Allows Direct Interaction
- ☐ Causes Page Turn
- ☐ Header

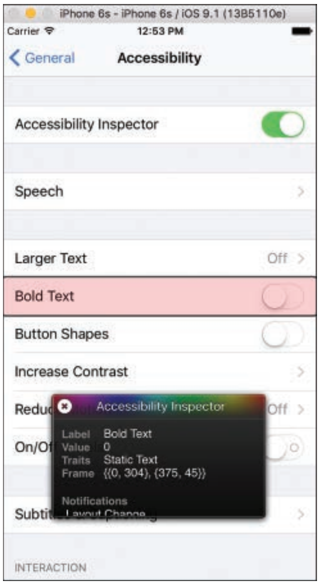
FIGURE 5.13
Use the Accessibility section in the Identity Inspector to configure how Voiceover interacts with your application.

TIP

For an application to be available to the largest possible audience, take advantage of accessibility tools whenever possible. Even objects such as the text labels you've used in this lesson should have their traits configured to indicate that they are static text. This helps potential users know that they can't interact with them.

Enabling the iOS Accessibility Inspector

If you are building accessible interfaces, you may want to enable the Accessibility Inspector in the iOS Simulator. To do this, start the Simulator and click the Home button to return to the home screen. Start the Settings application and navigate to General, Accessibility, and then use the switch to turn the Accessibility Inspector on, as shown in Figure 5.14.



iPhone 6s - iPhone 6s / iOS 9.1 (13B5110e)
Carrier 12:53 PM
General Accessibility
Accessibility Inspector
Speech
Larger Text Off
Bold Text
Button Shapes
Increase Contrast
Redu Off
On/O
Subti
INTERACTION

Accessibility Inspector
Label Bold Text
Value 0
Traits Static Text
Frame {{0, 304}, {375, 45}}
Notifications
Layout Change

FIGURE 5.14
Toggle the iOS Accessibility Inspector on.

The Accessibility Inspector adds an overlay to the Simulator workspace that displays the label, hints, and traits that you've configured for your interface elements. Note that navigating the iOS interface is *very* different when operating in accessibility mode.

Using the X button in the upper-left corner of the inspector, you can toggle it on and off. When off, the inspector collapses to a small bar, and the iPhone Simulator will behave normally. Clicking the X button again turns it back on. To disable the Accessibility Inspector altogether, just revisit the Accessibility setting in the Settings application.

Previewing the Interface

If you've worked with earlier versions of Xcode, you know that you could easily simulate your UI. This feature disappeared in Xcode 4, but has made a happy return in the latest Xcode releases (even if it is a bit difficult to find).

To use the preview feature, you must use the Xcode assistant editor feature that we reviewed in Hour 2. For example, open the PreviewUI.storyboard file included in this hour's Projects directory. This storyboard contains a simple user interface: a label, a field, and a button. Choose View, Show Toolbar from the menu bar so that all the typical window controls are showing. (On a normal project, the toolbar would already be visible.)

To preview the view as it will look on a device, first activate the assistant editor. Next, click the bar above the assistant editor pane where it reads Automatic. A drop-down menu appears, with Preview as the last option, as shown in Figure 5.15.

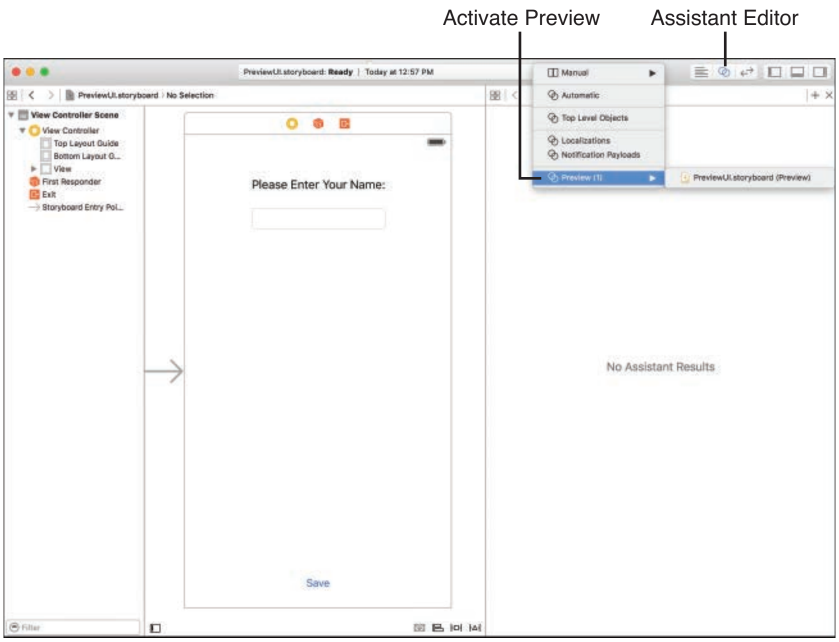


FIGURE 5.15
Select Preview (or choose from the options in its submenu) to activate a preview in the assistant editor.

Select Preview, and the assistant editor refreshes to show a live preview of what your UI will look like on a device. Use the button at the bottom of the preview to toggle between portrait and landscape orientations. In Figure 5.16, I'm previewing the UI on a 4.7-inch iPhone in landscape mode.

To add additional devices to the preview, click the + menu and choose the iOS device you'd like. If you're creating a project that runs on earlier versions of iOS, you can even choose to preview a device running a different OS version.

The preview is added to the right of the existing preview. You can even add multiples of the same device so that you can see a landscape and portrait view at the same time. To remove a preview, click to select it, and then press the Delete key.

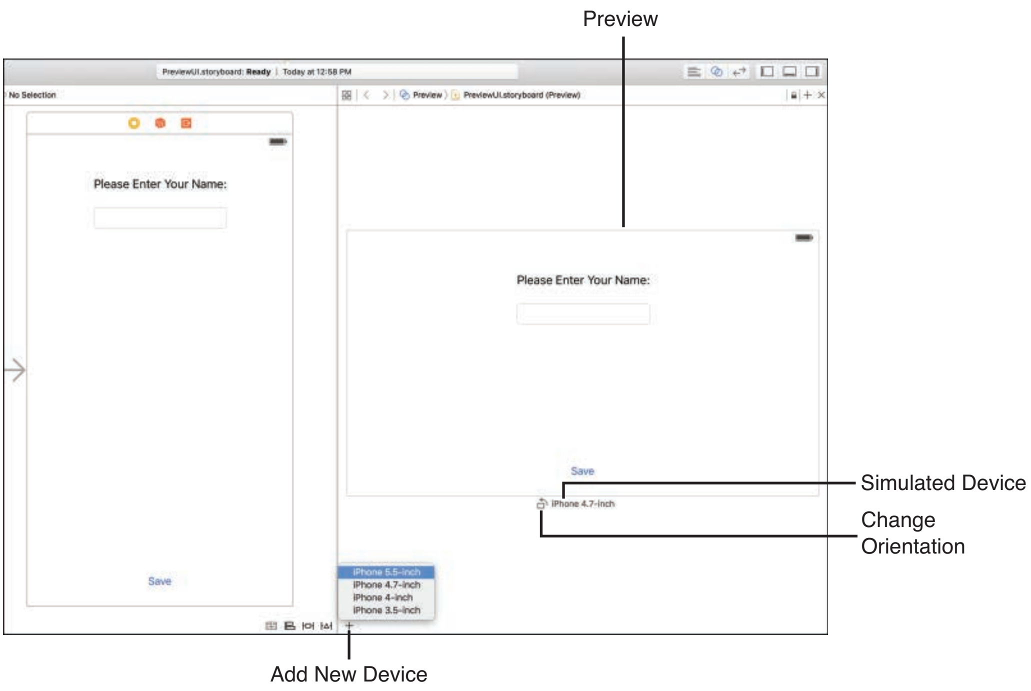


FIGURE 5.16
Preview the UI in different orientations and on different devices.

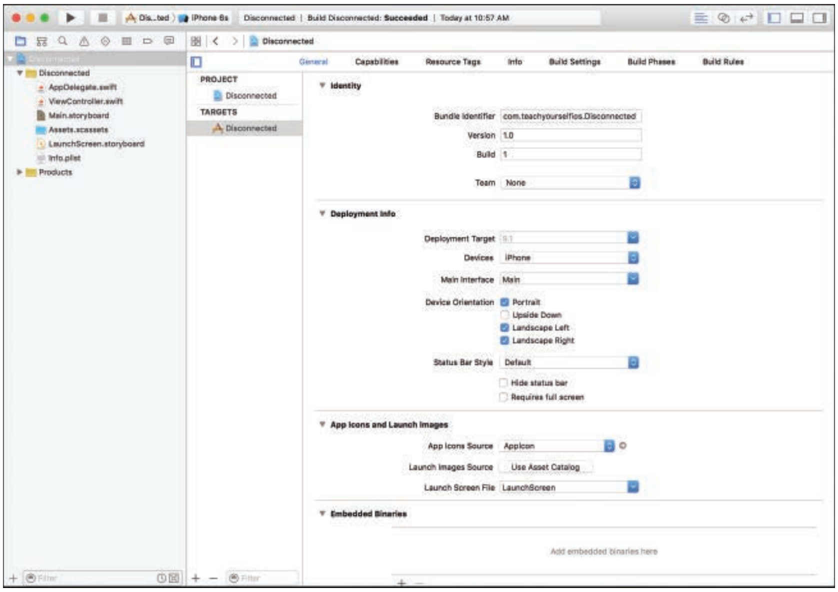
This will come in very handy when we explore Auto Layout more in Hour 16. In addition, if you want to test your application UIs, you can always run your apps in the iOS Simulator, even when they aren't entirely written. Apple's development tools make it possible to see results as you build, instead of having to wait until every single feature is in place.

Connecting to Code

You know how to make an interface, but how do you make it *do* something? Throughout this hour, I've been alluding to the idea that connecting an interface to the code you write is just a matter of "connecting the dots." In this last part of the hour, we do just that: take an interface and connect it to the code that makes it into a functional application.

Opening the Project

To get started, we use the project Disconnected contained within this hour's Projects folder. Open the folder and double-click the Disconnected.xcodeproj file. This opens the project in Xcode, as shown in Figure 5.17.



Identity

Bundle Identifier: com.teachyourselfios.Disconnected
Version: 1.0
Build: 1
Team: None

Deployment Info

Deployment Target: 9.1
Devices: iPhone
Main Interface: Main
Device Orientation: [✓ Portrait] Upside Down [✓ Landscape Left] [✓ Landscape Right]
Status Bar Style: Default
Hide status bar
Requires full screen

App Icons and Launch Images

App Icons Source: AppIcon
Launch Images Source: Use Asset Catalog
Launch Screen File: LaunchScreen

Embedded Binaries

FIGURE 5.17
To begin, open the project in Xcode.

After the project is loaded, expand the project code group (Disconnected) and click the Main.storyboard file. This storyboard file contains the scene and view that this application displays as its interface. Xcode refreshes and displays the scene in IB, as shown in Figure 5.18.

Implementation Overview

The interface contains four interactive elements: a button bar (called a segmented control), a push button, an output label, and a web view (an integrated web browser component). Together, these controls interface with application code to enable a user to pick a flower color, touch the Get Flower button, and then display the chosen color in a text label along with a matching flower photo fetched from the website http://www.floraphotographs.com. Figure 5.19 shows the final result.

NOTE

A Blurry Visual Treat

You may notice one additional (noninteractive) element in this project: a visual effects view. This view lives behind the color controls and labels and blurs out the background behind it. It's used to very easily create the soft blurred backgrounds prevalent in iOS.

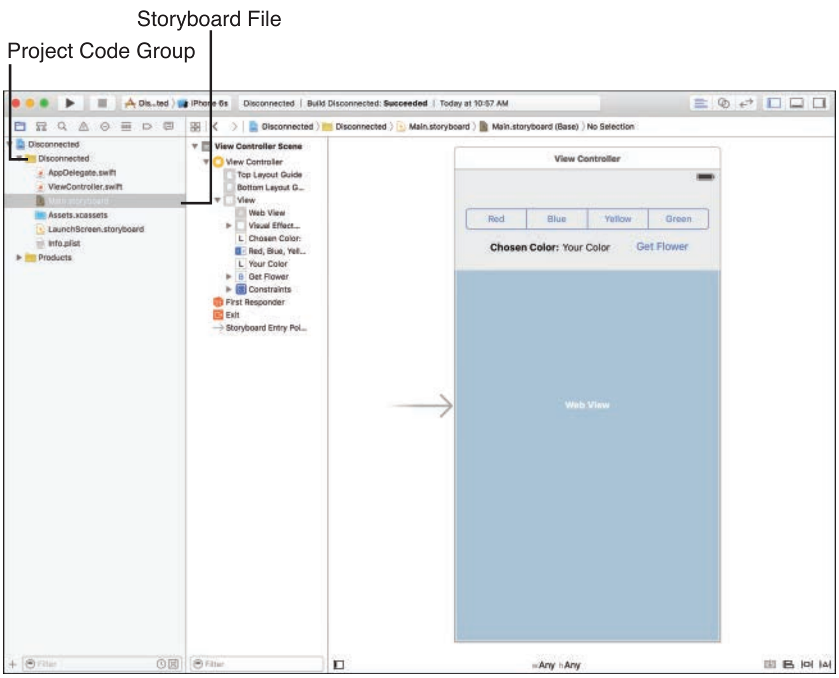


FIGURE 5.18
IB displays the scene and corresponding view for the application.

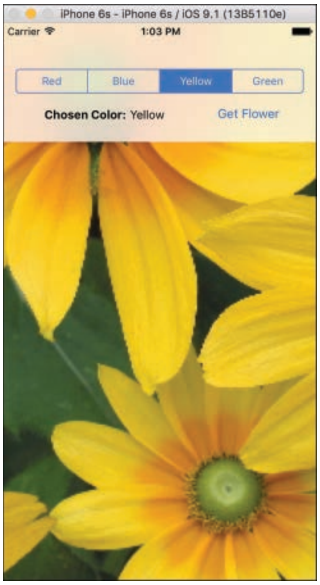


FIGURE 5.19
The finished application enables a user to choose a color and have a flower image returned that matches that color.

Unfortunately, right now the application does nothing. The interface isn't connected to any application code, so it is hardly more than a pretty picture. To make it work, we create connections to outlets and actions that have been defined in the application's code.

Outlets and Actions

An outlet is nothing more than a variable property by which an object can be referenced. For example, if you had created a field in IB intending that it would be used to collect a user's name, you might want to create an outlet for it in your code called `userName`. Using this outlet and a corresponding variable property, you could then access or change the contents of the field.

An action, however, is a method within your code that is called when an event takes place. Certain objects, such as buttons and switches, can trigger actions when a user interacts with them through an event, such as touching the screen. If you define actions in your code, IB can make them available to the onscreen objects.

Joining an element in IB to an outlet or action creates what is generically termed a *connection*. For the Disconnected app to function, we need to create connections to these outlets and actions:

- **`colorChoice`:** An outlet created for the button bar to access the color the user has selected
- **`getFlower`:** An action that retrieves a flower from the Web, displays it, and updates the label with the chosen color
- **`chosenColor`:** An outlet for the label that will be updated by getFlower to show the name of the chosen color
- **`flowerView`:** An outlet for the web view that will be updated by `getFlower` to show the image

Let's make the connections now.

Creating Connections to Outlets

To create a connection from an interface item to an outlet, Control-drag from a scene's View Controller icon (in the document outline area or the icon bar below the view) to either the visual representation of the object in the view or its icon in the document outline area.

Try this with the button bar (segmented control). Pressing Control, click and drag from the view controller in the document outline area to the onscreen image of the bar. A line appears as you drag, enabling you to easily point to the object that you want to use for the connect, as shown in Figure 5.20.

When you release the mouse button, the available connections are shown in a pop-up menu (see Figure 5.21). In this case, you want to pick `colorChoice`.

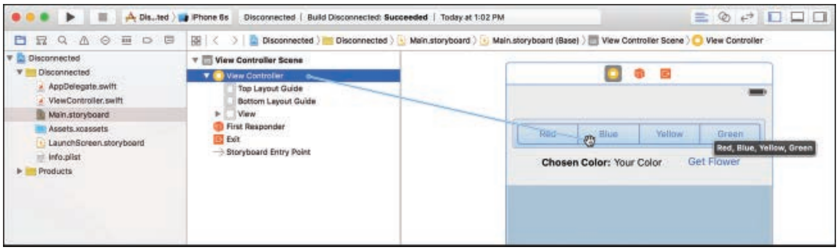


FIGURE 5.20
Control-drag from the view controller to the button bar.

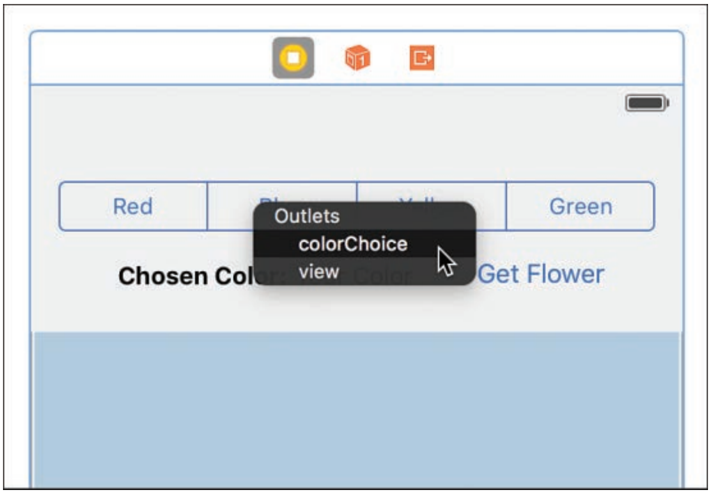


FIGURE 5.21
Choose from the outlets available for the targeted object.

NOTE

IB knows what type of object is allowed to connect to a given outlet, so it displays only the outlets appropriate for the connection you're trying to make.

Repeat this process for the label with the text `"Your Color"`, connecting it to the `chosenColor` outlet, and with the web view, connecting to `flowerView`.

Connecting to Actions

Connecting to actions is a bit different. An object's events trigger actions (methods) in your code. So, the connection direction reverses; you connect from the object invoking an event to the view controller of its scene. Although it is possible to Control-drag and create a connection in the same manner you did with outlets, this isn't recommended because you don't get to specify which event triggers it. Do users have to touch the button? Release their fingers from a button?

Actions can be triggered by *many* different events, so you need to make sure that you're picking exactly the right one, instead of leaving it up to IB. To do this, select the object that will be connecting to the action and open the Connections Inspector by clicking the arrow icon at the top of the Xcode utility area. You can also show the inspector by choosing View, Utilities, Show Connections Inspector (or by pressing Option-Command-6).

The Connections Inspector, in Figure 5.22, shows a list of the events that the object, in this case a button, supports. Beside each event is an open circle. To connect an event to an action in your code, click and drag from one of these circles to the scene's View Controller icon in the document outline area.

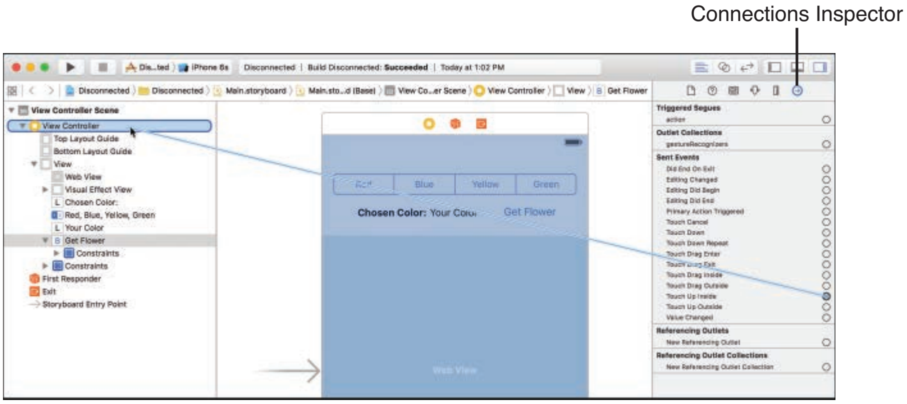


FIGURE 5.22
Use the Connections Inspector to view existing connections and to make new ones.

NOTE

I often refer to creating connections to a scene's view controller or placing interface elements in a scene's view. This is because IB storyboards can contain multiple different scenes, each with its own view controller and view. In the first few lessons, there is only a single scene, and therefore, a single view controller. That said, you should still be getting used to the idea of multiple View Controller icons appearing in the document outline area and having to correctly choose the one that corresponds to the scene you are editing.

For example, to connect the Get Flower button to the `getFlower` method, select the button, and then open the Connections Inspector (Option-Command-6). Drag from the circle beside the Touch Up Inside event to the scene's view controller and release, as demonstrated in Figure 5.22. When prompted, choose the `getFlower` action, shown in Figure 5.23.

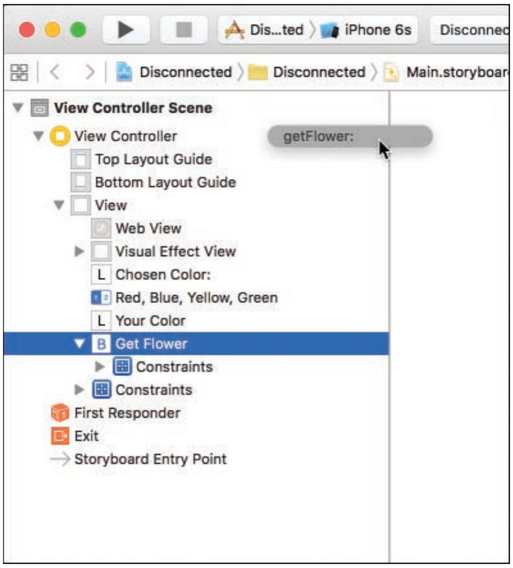


FIGURE 5.23
Choose the action you want the interface element to invoke.

After a connection has been made, the inspector updates to show the event and the action that it calls, demonstrated in Figure 5.24. If you click other already connected objects, you'll notice that the Connections Inspector shows their connections to outlets and to actions.

Well done! You've just linked an interface to the code that supports it. Click Run on the Xcode toolbar to build and run your application in the iOS Simulator or on your personal iDevice.

Connections Without Code

Although most of your connections in IB will be between objects and outlets and actions you've defined in your code, certain objects implement built-in actions that don't require you to write a single line of code.

The web view, for example, implements actions, including `goForward` and `goBack`. Using these actions, you could add basic navigation functionality to a web view by dragging from a button's Touch Up Inside event directly to the web view object (rather than the view controller). As described previously, you are prompted for the action to connect to, but this time, it isn't an action you had to code yourself.

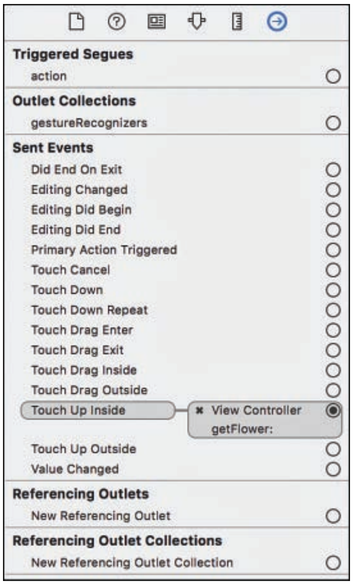


FIGURE 5.24
The Connections Inspector updates to show the actions and outlets that an object references.

Editing Connections with the Quick Inspector

One of the errors that I commonly make when connecting my interfaces is creating a connection that I didn't intend. A bit of overzealous dragging, and suddenly your interface is wired up incorrectly and won't work. To review the connections that are in place, you select an object and use the Connections Inspector discussed previously, or you can open the Quick Inspector by right-clicking any object in the IB editor or document outline area. This opens a floating window that contains all the outlets and actions either referenced or received by the object, as shown in Figure 5.25.

Besides viewing the connections that are in place, you can remove a connection by clicking the X next to a connected object (see Figure 5.24 and 5.25). You can even create new connections using the same "click-and-drag from the circle to an object" approach that you performed with the Connections Inspector. Click the X in the upper-left corner of the window to close the Quick Inspector.

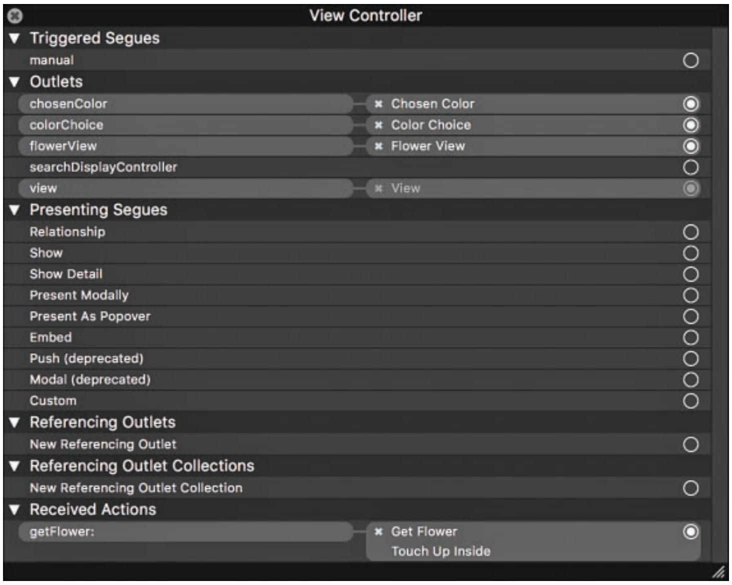


FIGURE 5.25
Right-click to quickly inspect any object connections.

NOTE

Although clicking an object, such as a button, shows you all the connections related to that object, it doesn't show you *everything* you've connected in the IB editor. Because almost all the connections you create will go to and from a scene's view controller, choosing it and then opening the inspector will give you a more complete picture of what connections you've made.

Writing Code with IB

You just created connections from UI objects to the corresponding outlets and actions that have already been defined in code. In the next hour's lesson, you write a full application, including defining outlets and actions and connecting them to a storyboard scene. What's interesting about this process, besides it bringing all the earlier lessons together, is that IB editor writes and inserts the necessary Swift code to define outlets and actions.

Although it is impossible for Xcode to write your application for you, it does create the instance variables and properties for your app's interface objects, as well as "stubs" of the methods your interface will trigger. All you need to do is drag and drop the IB objects into your source code files. Using this feature is completely optional, but it does help save time and avoid syntax errors.

TIP

A method stub (or *skeleton*) is nothing more than a method that has been declared but executes no instructions. You can add stubs to your code where you know what you'll be writing in the future but aren't yet ready to commit it to code. This is useful in the initial design stages of an application because it helps you keep track of the work you have left to do.

Stub methods are also helpful if you have code that needs to use a method that you haven't written. By inserting and referencing stubs for your unwritten methods, your application will compile and run—enabling the code that *is* complete to be tested at any stage of the development process.

Object Identity

As we finish up our introduction to IB, I'd be remiss if I didn't introduce one more feature: the Identity Inspector. You've already accessed this tool to view the accessibility attributes for interface objects, but there is another reason why we need to use the inspector in the future: setting class identities and labels.

As you drag objects into the interface, you're creating instances of classes that already exist (buttons, labels, and so on). Throughout this book, however, we build custom subclasses that we also need to be able to reference with IB's objects. In these cases, we need to help IB by identifying the subclass it should use.

For example, suppose we created a subclass of the standard button class (`UIButton`) that we named `ourFancyButtonClass`. We might drag a button into a scene to represent our fancy button, but when the storyboard file loads, it would just create the same old `UIButton`.

To fix the problem, we select the button we've added to the view, open the Identity Inspector by clicking the window icon at the top of the Xcode utility area or by choosing View, Utilities, Show Identity Inspector (Option-Command-3), and then use the drop-down menu/field to enter the class that we really want instantiated at runtime (see Figure 5.26).

This is something we cover on an as-needed basis; so if it seems confusing, don't worry. We come back to it later in the book.

NOTE

I see that Module Name field you're ignoring! When setting a custom class name, you can also set the name of the module that defines the class. Modules provide a way to organize large numbers of classes into functional groups, but aren't something you'll need for the projects in this book.

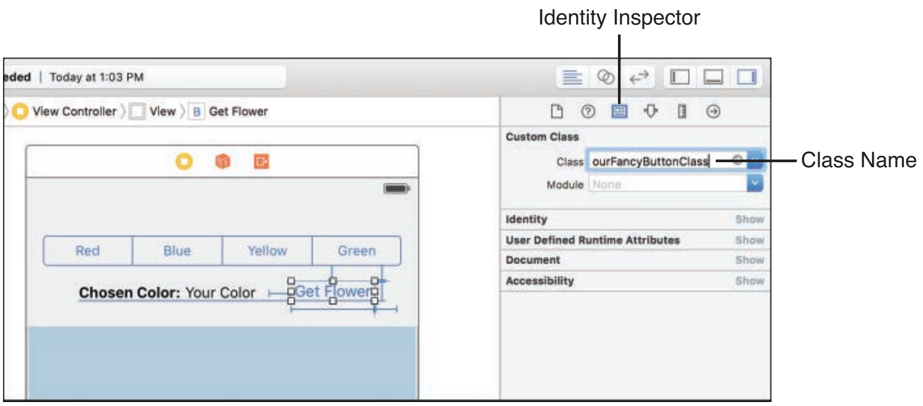


FIGURE 5.26
If you're using a custom class, you need to manually set the identity of your objects in the Identity Inspector.

Further Exploration

The IB editor gives you the opportunity to experiment with many of the different GUI objects you've seen in iOS applications and read about in the previous hours. In the next hour, the Xcode code editor is used in conjunction with IB for your first full project, developed from start to finish.

To learn even more about what you can do with IB, I suggest reading through the following five Apple publications:

- ***Interface Builder Help:*** Accessed by right-clicking the background in the IB editor, the IB help is more than a simple help document. Apple's IB Help walks you through the intricacies of IB using video tutorials and covers some advanced topics that will be important as your development experience increases.
- ***Auto Layout Guide:*** This document presents a good introduction to the Auto Layout system and is an excellent read for anyone wanting to get a jump start on adaptive interface layout techniques.
- ***Xcode Overview: Build a User Interface:*** Part of the larger "Xcode Overview" document, this section offers a nice tutorial on Interface Builder principals.
- ***iOS Human Interface Guidelines:*** The Apple iOS HIG document provides a clear set of rules for building usable interfaces on the iOS device family. This document describes when you should use controls and how they should be displayed, helping you create more polished, professional-quality applications.

- ***Accessibility Programming Guide for iOS:*** If you're serious about creating accessible apps, this is a mandatory read. The *Accessibility Programming Guide* describes the accessibility features in this hour's lesson as well as ways to improve accessibility programmatically and methods of testing accessibility beyond the tips given in this hour.

As a general note, from here on, you do quite a bit of coding in each lesson. So, now is a great time to review the previous hours if you have any questions.

Summary

In this hour, you explored the Xcode IB editor and the tools it provides for building rich graphical interfaces for your iOS applications. You learned how to navigate IB storyboards and access the GUI elements from the Object Library. Using the various inspector tools within IB, you reviewed how GUI elements can be placed within a scene using constraints, how the look and feel of onscreen controls can be customized, and how interfaces can be made accessible to the visually impaired.

More than just a pretty picture, an IB-created interface uses simple outlets and actions to connect to functionality in your code. You used IB's connection tools to turn a nonfunctioning interface into a complete application. By maintaining a separation between the code you write and what is displayed to the user, you can revise your interface to look however you want, without breaking your application. In Hour 6, you examine how to create outlets and actions from scratch in Xcode (and thus gain a full toolset to get started developing).

Q&A

Q. Why do I keep seeing things referred to as NIB/XIB files?

A. The origins of IB trace back to the NeXT Computer, which made use of NIB files to store individual views. These files, in fact, still bore the same name when Mac OS X was released. In recent years, however, Apple renamed the files to have the .xib extension, which has subsequently been mostly replaced by storyboards and scenes. You'll still see a XIB file used for your startup screen in your project (see Hour 2 for details), but, in general, anything that refers to a XIB or NIB file applies to storyboards as well.

Q. Some of the objects in the IB Object Library can't be added to my view. What gives?

A. Not all the items in the Object Library are interface objects. Some represent objects that provide functionality to your application. These can be added to the scene in the document outline area or on the icon bar located below a scene's layout in the IB editor.

Q. I've seen controls in applications that aren't available here. Where are they?

A. Keep in mind that the iOS objects are heavily customizable and frequently used as a starting point for developers to make their own UI classes or subclasses. The end result can vary tremendously from the stock UI appearance.

Workshop

Quiz

1. The default storyboard file is named what?
    - **a.** Main
    - **b.** iPhone
    - **c.** Universal
    - **d.** Default
2. Which inspector enables you to update the appearance of an interface object?
    - **a.** Identity
    - **b.** Appearance
    - **c.** Visual
    - **d.** Attributes
3. To change the height or width of an object, you could use which inspector?
    - **a.** Attributes
    - **b.** Constraints
    - **c.** Volumetric
    - **d.** Size
4. What system gives us a way to describe interfaces that resize and change depending on device screen size and orientation?
    - **a.** Auto Adapt
    - **b.** Attributes
    - **c.** Auto Layout
    - **d.** Content Autosizing

5. To set a custom class on an object, we would turn to which inspector?
    - **a.** Class
    - **b.** Identity
    - **c.** Object
    - **d.** Location
6. Invoking the Quick Inspector on which object is a good way to see most of your scene's connections?
    - **a.** View
    - **b.** View Controller
    - **c.** First Responder
    - **d.** Class
7. Through what is an interface object referenced in code?
    - **a.** Plug
    - **b.** Action
    - **c.** Connection
    - **d.** Outlet
8. Interactive interface elements often connect to code via which of the following?
    - **a.** Classes
    - **b.** Actions
    - **c.** Connections
    - **d.** Outlets
9. To test the accessibility of iOS interfaces, you can activate which of the following tools in the iOS Simulator?
    - **a.** Accessibility Viewer
    - **b.** Accessibility Chain
    - **c.** Accessibility Inspector
    - **d.** Accessibility Wizard

10. What library enables you to find and add interface objects to a scene?

- **a.** Object
- **b.** Media
- **c.** Interface
- **d.** Tool

Answers

1. A. The default storyboard name is simply Main.storyboard.
2. D. Use the Attributes inspector to change visual properties for any object in your interface layout.
3. D. You can use the Size Inspector to fine-tune the height and width of most interface elements.
4. C. Auto Layout is the name of Apple's system for describing size/orientation-independent interfaces.
5. B. The Identity Inspector enables you to set an object to a custom class.
6. B. Select the View Controller object and open the Quick Inspector to view most of the connections within a scene.
7. D. Outlets connect interface objects to a variable property in code.
8. B. Actions provide a connection point between an interface element and an underlying method in code.
9. C. Use the Accessibility Inspector to view the accessibility properties set on interface elements within the iOS Simulator.
10. A. You'll use the Object Library to find, select, and place interface elements in Interface Builder.

Activities

1. Practice using the interface layout tools on the Empty.storyboard file. Add each available interface object to your view, and then review the Attributes Inspector for that object. If an attribute doesn't make sense, remember that you can review documentation for the class to identify the role of each of its properties.
2. Revise the Disconnected project with an accessible interface. Review the finished design using the Accessibility Inspector in the iOS Simulator.

This page intentionally left blank

Index

&& (AND) expression, 103
*** (asterisks), 101**
{ } (curly braces), 45, 80
– (decrementing), 105
== (equal to) expression, 103
! (exclamation points), 748
++ (incrementing), 105
! (negation) expression, 103
!= (not equal) expression, 103
| | (OR) expression, 103
( ) (parenthesis), 45, 103
? (question marks), 94, 748
3D graphics, 8
3D Touch
  peek and pop
    adding, 631-632
    editor screen, 655-657
    implementing, 654-655
    overview, 630-631
    table views, 657-659
    testing, 633
  Quick Actions, 817-819
    defining, 818-819
    events, handling, 846-847
    icons, 819, 844
    .plist file configuration, 844-845
    responding, 819

A

a: date format, 432
Accelerate framework, 131
acceleration data, 665, 683-684
accelerometers, 663-665
  1g of force on devices at rest, 665
  acceleration data, 665, 683-684
  attitude, 665
  measurable axes, 664
access levels (Swift), 84
accessibility attributes (UIs), 174-175
Accessibility Inspector, enabling, 175-176
accessing
  dictionary values, 89
  direct file system access. *See* direct file system access
  media items, 697-698
  music library. *See* music player
  optional values, 94-95
  orientation notifications, 667-668
  popover presentation controller, 383
  Safari web browser, 752-753
  System Sound Services, 338-339
accessors, 81
Accounts framework, 129
accuracy (locations), 783, 786
action sheets, 334-337
  implementing, 350-352
  popovers, 352
actions
  alerts, 333-334, 344-345
  animation loops, 275-276
  BestFriend app, 765
  built-in, 185
  buttons
    creating, 246-247
    press method definition, 206
    triggering, 206
  ColorTilt app, 679
  connections, creating, 183
  CustomPicker app, 444
  date pickers, 430
  defined, 181
  direct file system access preferences, 574-575
  gestures, 642-644
  hideKeyboard, 248
  ImageHop app, 264
  implicit preferences, 554
  MediaPlayground app, 715
  ModalEditor app, 399
  navigation controllers, 479
  switches, 307-309
  tab bar controllers, 487
  view to view controller connections, 215
adaptive segues, disabling, 406
add contact buttons, 243
Add Horizontal Guide/Add Vertical Guide commands (Xcode Editor menu), 166
AddMusic Apple tutorial, 736
Address Book framework, 129
addTextFieldWithConfiguration Handler method, 333
Adobe Flash Professional, 10
AirPlay support, 700
alertBody property, 825
alerts, 331-334
  action sheets, 334-337
    implementing, 350-352
    popovers, 352
  actions, 333-334

buttons, 332-333
example, 331
fields, adding, 349-350
implementing, 331-333, 345-346
multibutton, creating, 346-348
Playground, 336-337
sounds/vibrations, 339-340, 353-355
text fields, adding, 333

Align command (Xcode Editor menu), 167

aligning
constraints, 593, 604
UI objects, 167-168

AllInCode app
button touches, handling, 621
interface, drawing upon launching, 620
objects
defining, 617
initializing, 617-618
project setup, 616-617
rotations, handling, 621
running, 621
screen orientations, enabling, 616
update method, implementing, 619-620

allowsExternalPlayback property, 700

anchors, 376

AND (&&) expression, 103

animal chooser scene, 441-442

animal picker. *See* CustomPicker app

AnimalChooserViewController class, 439

animated image view, implementing, 277-278

animationDuration property, 269, 278

animationImages property, 278

animations
loading, 277-278
looping
actions, 264, 275-276
animated image view, implementing, 277-278
background image, 271-272
blur visual effect, 272-273
default images, setting, 265-266
image copies, creating, 266-267
image views, adding, 264
outlets, 264, 274-275
Playground, 281-282
project, setting up, 263
resources, adding, 263
running, 284
slider range attributes, 268-269
speed output labels, 271
speed slider, adding, 267-268
start/stop button, adding, 271
status bar appearance, 283-284
steppers, adding, 269-270
Quartz Core framework, 129
resources, adding, 263
speed
incrementing, 282-283
setting, 280-281
starting/stopping, 278-280

annotations, 755-757
adding, 755
customizing, 756-757, 770-771
deleting, 756

Any, Any size class, 859, 873

Any Landscape screen orientation constant, 586

Anything But Upside-Down screen orientation constant, 587

Anything screen orientation constant, 587

Any(w), Compact(h) size class, 870-873

Any(w), Regular(h) size classes, configuring, 868-870

API Reference, 144

App Store, 2

App Transport Security (ATS), 311

appearance
bar button items, configuring, 416
buttons, editing, 242-243
date pickers, configuring, 418
frosted glass, 561
keyboard, 236
popovers, troubleshooting, 405
segmented controls, 301
status bars, configuring, 283-284, 775, 796
tables, 498
text fields, editing, 233-235
UIs
accessibility attributes, 174-175
Attributes Inspector, 173-174

append method, 88

Apple Developer Program, 10-13

Apple IDs
creating, 11
Xcode, adding, 16

Apple Maps integration. *See* mapping

Apple Settings application, 540-541

application logic
animation loops
animated image view, implementing, 277-278
setting speed, 280-281
starting/stopping animations, 278-280
animations
incrementing speed, 282-283
status bar appearance, 284
BackgroundColor app
creating preferences, 555
reading preferences, 557
storing preferences, 556-557
BackgroundDownload app, 841-842
ColorTilt app, 679-680
acceleration data, handling, 683-684

attitude data, handling, 682-683
CMMotionManager, initializing, 680-681
interface-orientation changes, disabling, 685-686
motion tracking switch updates, managing, 681-682
rotations, handling, 684-685
Cupertino app
distance to Cupertino, calculating, 795-796
errors, handling, 794
location manager instance, creating, 793-794
location manager preparations, 793
plist file update, 797
status bar appearance, 796
Cupertino Compass app
handling heading updates, 805-807
heading to Cupertino, calculating, 804-805
recent locations, storing, 803-804
requesting heading updates, 802
current date, retrieving, 431
DateCalc app, 431
date differences, calculating, 432-434
date output, updating, 434-435
date/time, displaying, 431-432
FieldButtonFun, 251-252
flower app, 309-314
flower images, loading, 311-313
initial display, updating, 313-314
FlowerColorTable app, 514-518
cells to display, configuring, 516-517
flower arrays, populating, 514
number of sections, returning, 515
row selections, handling, 517-518
section headings, 515
section row counts, returning, 515
Gestures app, 644
image view, replacing, 644-645
pinch recognizer response, 646-649
rotation recognizer response, 649-650
shake gesture response, 653
shake recognizer implementation, 651-653
swipe recognizer response, 646
tap gesture recognizer response, 645-646
HelloNoun, 216-217
LetsNavigate app, 479-480
LetsTab app, 488
count button presses, tracking, 488
counter updates, triggering, 490
display updates with counter values, 488-489
tab bar item badges, updating, 490
ModalEditor app, 400-401
emailField text field, populating, 400
initial scene label to editor scene field, setting, 401
keyboard, hiding, 401
Orientation app, 672-675
orientation changes, responding, 673-675
orientation updates, registering, 673
ReturnMe app, 569-570
SlowCount app, 836
counters, updating, 837
timer, initializing, 836
Survey app, 575-578
displaying survey results, 577-578
keyboard, hiding, 575
storing survey results, 576-577
applicationDidBecomeActive method, 821
applicationDidEnterBackground method, 133, 821
application:didFinishLaunchingWithOptions method, 821, 824
applicationIconBadgeNumber property, 825
application:performActionForShortcutItem method, 846
application:performFetchWithCompletionHandler method, 842
applications
badges, creating, 355
building process, 51
converting to universal, 404-407
data structures
creating, 524
populating, 525-526
device orientation, setting, 56
icons, 57-60
life cycles
background-aware methods, 820-821
diagram, 132
tracing, 133
preferences. *See* preferences
quitting, 218
running
development provisioning profiles, 15-16
device configurations, 16-18
iOS Simulator, 51
Welcome app, 19-21
what happens on devices, 17-18
sandbox, 546-547
Simulator, launching, 62-63
state preservation, 541-542
submitting to App Store, 2
suspending, 133, 814

UIApplication class, 134
universal. *See* universal
applicationWillEnterForeground method, 821, 823
applicationWillResignActive method, 821
applicationWillTerminate method, 821
ARC (Automatic Reference Counting), 110-111
Arrange command (Xcode Editor menu), 167
arranging UI objects, 167, 169
arrays, 85
animated image views, implementing, 278
classes, 136
contents, counting, 89
declaring/initializing, 88
FlowerDetail app
creating, 524
populating, 525-526
index values, 88
iterating over, 106
new items, adding to end, 88
populating, 514
userMessages, 88
asset catalogs, 38-42
application icons, setting, 58-60
contents, viewing, 39
features, 38
files, adding, 39
new, adding, 38
Retina images, 40-42
slicing, 40
sound files, 340
assistant editor mode (Xcode), 48
asterisks (*), 101
ATS (App Transport Security), 311
attitude data, handling, 665, 682-683
attributed text, 235
attributes
bar button items, 416
buttons, editing, 242-243
contentSize, 322
date pickers, 418
navigation bar items, 462-463
slider range, 268-269
stepper range, 270
tab bar items, 466
tables, 500-501, 512
tap gesture recognizers, 638
text fields
editing, 233-235
input traits, 236-237
text views, editing, 238-239
UI
objects, customizing, 172
simulated, 208
web view, 303-305
Attributes Inspector, 173-174
bar button items, 416
buttons, 242-243
constraints, editing, 595-597
custom button images, 244-245
date pickers, 418
navigation bar items, 462-463
pinches, 641
slider ranges, 268-269
stepper ranges, 270
tab bar items, 466
tables, 500-501, 512
text fields, 233-235
text views, 238-239
web views, 304-305
audio
AddMusic Apple tutorial, 736
alerts, 339-340
alerts with vibrations, 353-355
asset catalog storage, 340
AV Audio Player
audio files, playing, 702
customizing, 736
playback completion, handling, 703
playback controls, 702
books filter, 695
file filter, 695
formats supported, 699
frameworks
AV Foundation, 127
Core Audio, 127
loading/playing, 338
music player
empty selections, handling, 733-734
media picker, displaying, 732-733
media picker preparations, 730
playing music, 734-735
playlist, creating, 733
preparations, 730-731
playback
classes, 698
controlling, 723
implementing, 722-723
recorded sounds, loading, 723-724
playing in the background, 828
audio files, adding, 828
AudioToolbox framework, adding, 828
background modes, 832
heading update audio feedback, adding, 830-831
sound file references, initializing, 829
system sound ID variables, 829
recording, 703-705
controlling, 721-722
implementing, 720-721
microphone prompt, 722
resources, adding, 340
system, playing, 353-354
System Sound Services, 337-339
audioPlayer property, 722
audioRecorder property, 720
AudioToolbox framework
adding, 828
importing, 353
authentication framework, 132
authorization
local notifications, 824-825
user locations, 768-769, 782
Auto Layouts.
constraints. *See* constraints, Auto Layouts
Content Compression, 171
Content Hugging, 171
Intrinsic Size setting, 172
responsive interfaces, 588, 590
turning off, 172

autocompleting code, 44-45
Auto-Enable Return Key attribute, 236-237
automatic constraints, 172
Automatic Reference Counting (ARC), 110-111
AV Audio Player
  audio files, playing, 702
  customizing, 736
  playback completion, handling, 703
  playback controls, 702
AV Audio Recorder, 703-705
  controlling, 721-722
  implementing, 720-721
AV Foundation framework, 127, 698-699
AV Kit framework, 127, 698-699
  external device playback/ AirPlay support, 700
  loading/displaying movies, 699-700
  playback completion, handling, 701-702
  playback controls, 701
  screen placement, 700
availableData method, 550
AVAudioPlayer class, 699
AVAudioRecorder class, 698
AVPlayer class, 698
AVPlayerItem class, 698
AVPlayerViewController class, 698, 736

B

background fetches, 817
  Background Fetch background mode, adding, 843
  completion, handling, 842-843
  minimum fetch intervals, defining, 841-842
background-aware methods, 819-821
BackgroundColor app
  actions, 554
  background color preferences
    creating, 555
    reading, 557
    storing, 556-557
  color slider, 554
  key constants, adding, 552
  on/off switch, 554
  project setup, 552
  running, 557
  UI design, 552-553
backgroundColor property, 555
BackgroundDownload app, 839
  Background Fetch background mode, adding, 843
  completion, handling, 842-843
  minimum background fetch intervals, defining, 841-842
  outlet, 841
  project setup, 840
  running, 843
  UI, 840
backgrounding
  3D Touch Quick Actions, 817-819
    defining, 818-819
    events, handling, 846-847
    icons, 819, 844
    .plist file configuration, 844-845
    responding, 819
  Apple documentation, 848
  application life cycle methods, 820-821
  application suspension
    implementing, 823
    overview, 814
  background fetches, 817
    Background Fetch background mode, adding, 843
    completion, handling, 842-843
    minimum fetch intervals, defining, 841-842
  constraints, 9
  disabling, 821-822
  iPads, 859
  local notifications, 814-815
    authorization, requesting, 824-825
    creating, 826-827
    properties, 825
    scheduling, 827
  split-screen, 814
  suspension, 823
  task completion, 816
    counting slowly app. *See* SlowCount app
    enabling, 837-839
  task-specific. *See* task-specific backgrounding
backgrounds
  images
    adding, 271-272
    blur visual effect, adding, 272-273
  modes, 832
  touches, 249-250
badges
  creating, 355
  tab bar items, updating, 489
  tabs, 466
bar button items
  navigation bars, 460
  toolbars, 415-416
    adding, 415
    appearance, configuring, 416
    positioning, 416
bars
  navigation, 460
    bar button items, 460
    item attributes, 462-463
  tab
    defined, 465
    images, 468
    items, 465-466
  toolbars. *See* toolbars
battery life, 787
beginBackgroundTaskWithExpirationHandler method, 816
beginGeneratingDeviceOrientationNotifications method, 667
behaviors
  scrolling, 322-323
  text fields, editing, 233-235

BestFriend app
- actions, 765
- Contacts
  - CNContactPickerView Controller protocol, 765
  - contact selections, handling, 766-768
  - picker view, displaying, 766
- email, 771
  - composition completion, handling, 772-773
  - Mail compose delegate protocol, conforming, 771
  - messages, composing, 772
- mapping, 768
  - annotations, 770-771
  - display, controlling, 769-770
  - user locations, displaying, 768-769
- outlets, 764
- project setup, 762
  - frameworks, adding, 762
  - variables/connections, planning, 762
- running, 775
- Safari web views, opening, 775
- social networking, 773-774
- status bar appearance, 775
- UI design, 762-764

binding optional values, 95
Bluetooth, 131
blur effect, 272-273
- Cupertino app, adding, 798-799
- FlowerWeb app, 314-315
- labels, adding, 561

bookmarking Xcode documentation, 145-146
Bool data type, 85, 87, 136
boolForKey method, 557
Bottom Layout Guide icon (storyboards), 159
bottom layout guides, 594
breakpoints, 885-886
- navigator, 893
- watchpoints, configuring, 890-891

bridged data type classes, 136
browsing Xcode documentation, 144-145
build schemes, choosing, 50-51
building
- applications, 51
- code, 50-51

buttons, 223-224
- actions
  - creating, 246-247
  - triggering, 206
- adding, 241
- alerts, 332-333
- animations
  - starting/stopping, 271
  - triggering, 275
- appearance, 242-243
- attributes, editing, 242-243
- bar button items, 415-416
  - adding, 415
  - appearance, configuring, 416
  - positioning, 416
- centering, 604-606
- Clear, 234
- custom images, 243-245
- flower app, adding, 305
- keyboards, hiding, 248-249
- labels, editing, 342
- presses, tracking, 488
- state, changing, 243
- templates, creating, 227-232
  - images, adding, 227-228
  - slices, creating, 228-231
- touches, handling, 621
- types, 243
- UIButton class, 139
- variable/size matching constraints, 612-614

C

calculateDateDifference method, 433
Calendar and Reminders Programming Guide, 777
call stacks, 893
canBecomeFirstResponder method, 651
Canvas, Show Selection Highlights command (Xcode Editor menu), 166
Capitalize attribute, 236
case sensitivity (Swift), 77
cells
- colors, 502
- display, configuring, 516-517
- editing, disabling, 530
- further exploration, 533
- identifiers, 502
- images, adding, 502
- labels, 502
- master view controllers, creating, 528-529
- prototype, 512
- styles, 502
- tables, 498
  - prototype, configuring, 501
  - sizing, 501

centering layout constraints, 604-606
CFNetwork framework, 129
chaining
- methods, 98-99
- optional values, 99

characters (strings), 87
child preference panes, 544, 566-568
child view controllers, adding, 700
chooseImage method, 725
classes
- AnimalChooserViewController, 439
- audio playback, 698
- AVPlayerViewController, 736
- CLGeocoder, 757
- CLLocationManager class, 781

CMMotionActivity, 687
CMMotionActivityManager, 687
core application, 134-135
  UIApplication, 134
  UIControl, 135
  UIResponder, 135
  UIView, 135
  UIViewController, 135
  UIWindow, 134
CountingNavigationController, 459
data type, 136-137
  bridged, 136
  nonbridged, 136
  NSDate, 137
  NSURL, 137
DateChooserViewController, 425
declaration, 80
declaring, 80
defined, 75
EditorViewController, adding, 389
extensions, 76
gesture-recognizer, 628
interface, 138-141
  UIButton, 139
  UIDatePicker/UIPicker, 140
  UILabel, 138
  UIPopoverPresentation Controller, 141
  UISegmentedControl, 139
  UISlider, 139
  UIStepper, 140
  UISwitch, 139
  UITextField/UITextView, 140
MasterViewController, 527
media player, 694
NSURL, 295
NSURLRequest, 295
NSUserDefaults, 543
singletons, 76
size, 856, 857-859
  Any, Any, 859
  Any(w), Compact(h), 870-873
  Any(w), Regular(h), 868-870
  definitions and devices/ orientations represented, 858
  documentation, 857
  fonts, 865
  images, 866
  installed, configuring, 862-863
  manually configuring, 863-865
  responsive interfaces, 590
  selecting, 860-862
  storyboards, creating, 867-873
Swift file structure, 78-79
UIActionSheet, implementing, 335
UIBarButtonItem, 460
UIColor, 555
UIImagePickerController, 705
UILabel, 90
UINavigationBar, 460
UINavigationController, 459
UIScrollView, 324
UIStackView, 324
UITabBar, 465
UITabBarController, 464
UITouch, 659
UIViewControllerTransition Coordinator, 622
UIVisualEffectView, 285
UIWebView, 324
video playback, 698
view controllers, adding to navigation controllers, 472
ViewController, 43

cleanup, handling
  image pickers, 726-727
  video, 718
Clear buttons, 234
CLGeocoder class, 757
CLLocationManager class, 781
closures, 101, 330
CMMotionActivity class, 687
CMMotionActivityManager class, 687
CMMotionManager
  configuring, 669-670
  initializing, 680-681
CN contact picker delegate, 745-746
Cocoa, 125
Cocoa Touch, 23, 123
  advantages, 125
  alert controllers, 330
    action sheets, 334-337
    alerts, 331-334
  bridged data types, 136
  classes
    core application, 134-135
    data type, 136-137
    interface, 138-141
    Playground, 137
  Cocoa, compared, 125
  frameworks, 126-127
    Contacts UI, 126
    EventKit UI, 126
    GameKit, 126
    iAd, 127
    MapKit, 126
    Message UI, 126
    Notification Center, 127
    TextKit, 127
    UIKit, 126
  functionality, 124
  overview, 124
code
  application icons, 57-60
  auto completion, 44-45
  build schemes, choosing, 50-51
  comments, adding, 46-47
  curly braces/parenthesis, autocompleting, 45
  device orientation, setting, 56
  editing, 42
    assistant editor mode, 48
    tabbed editing, 49-50
  files, 35-36
  IB, writing, 186
  launch images/screens, setting, 60
  methods/properties, finding, 43
  mixing development methodology, 194

navigating, 43
problems, fixing, 52-55
searching, 45-46
UIs, connecting, 178
actions, 183
outlets, 181-182
project, opening, 178
colors (cells), 502
ColorTilt app
action, 679
application logic, 679-680
acceleration data, handling, 683-684
attitude data, handling, 682-683
CMMotionManager, initializing, 680-681
interface-orientation changes, disabling, 685-686
motion tracking switch updates, managing, 681-682
rotations, handling, 684-685
outlets, 679
project setup, 676-677
running, 686
UI, 677
commands
Editor menu (Xcode)
Add Horizontal Guide/Add Vertical Guide, 166
Align, 167
Arrange, 167
Canvas, Show Selection Highlights, 166
Show Document Outline, 232
File menu (Xcode)
New, Playground, 111
New, Tab, 49
New Project, 30
Hardware menu (Xcode)
Device, 65
Home, 65
Lock, 65
Reboot, 65
Shake Gesture, 65
Simulate Hardware Keyboard, 66
Simulate Memory Warning, 66
Toggle In-Call Status Bar, 66
Touch ID Enrolled/ Simulate Finger Touch, 65
TV Out, 66
Help, Quick Help (Xcode), 147
Preferences, Text Editing, Editing (Xcode), 45
View, Navigators, Show Navigator (Playground), 117
View menu (Xcode)
Utilities, Show Attributes Inspector, 173
Utilities, Show Connections Inspector, 183
Utilities, Show Identity Inspector, 174
Utilities, Show Object Library, 162
Utilities, Show Size Inspector, 168
Window, Devices (Xcode), 67
comments
code, adding, 46-47
Swift, 83
comparison expressions, 103
component constants (custom pickers), 440
composeViewControllerFor ServiceType method, 751
concatenation, 87
condition-based loops, 106-107
configuring
animations
loop default images, 265-266
animation speed, 280-281
application icons, 58-60
bar button item appearance, 416
CMMotionManager, 669-670
constraints, 609-612
custom button images, 243-245
date pickers
appearance, 418
scenes, 426
detail view controllers, 531-533
devices
development, 16-18
orientations supported, 56
image pickers, 705-706
initial scenes, 205
media pickers, 694-695
navigation controllers, 472-473
object identities, 187-188
popover segues, 402-404
projects as universal, 854
prototype cells, 501, 512
segmented control segments, 300-301
segue styles, 382
simulated screen sizes, 232
size classes
Any(w), Regular(h), 868-870
installed, 862-863
manually, 863-865
status bar appearance, 775, 796
table layout, 503-504
watchpoints, 890-891
Xcode
debugger breakpoints, 885-886
documentation downloads, 142-143
connections
actions. *See* actions
delegate/data source outlets to view controllers, 513-514
devices, 9
exits, 379-380
Mad Libs-style story creator, planning, 227
outlets. *See* outlets
tab bar scenes to tab bar controllers, 484-485
UIs, 178
actions, 183
deleting, 186
editing, 185-186

outlets, 181-182
project, opening, 178
view controllers, 205-206
views to view controllers, 212-215
actions, 215
outlets, 213-215
Connections Inspector, 183
constants
components (custom pickers), 440
declaring/initializing, 81, 92
defined, 76
key
implicit preferences, 552
settings bundles, 559
lastName, 92
location accuracy, 786
radian conversion, 677
screen orientation, 586-587
table sections, adding, 511
variables, compared, 93
constraints
Any(w), Compact(h) size class, 872
Any(w), Regular(h) size class, 869
Auto Layout, 170-171
adding, 591-593
alignment, 593
centering, 604-606
content compression resistance, 598
content hugging, 597
controls, expanding, 607-612
editing, 595-597
errors, fixing, 598-601
horizontal/vertical, 593
manually defining, 601-604
removing, 634
top/bottom layout guides, 594
variable/size matching, 612-614
viewing, 594
automatic, 172
date pickers
date chooser scenes, 428-429
initial scenes, 427-428
installed size classes, 862-863
Size Inspector, 169
object, 170
views, 394
contactPickerDidCancel method, 745
Contacts, 744
CNContactPickerViewController protocol, 765
contact selections, handling, 766-768
frameworks, 776
Contacts, 746-748
Contacts UI, 744-745
people picker
cancel action, 745
displaying, 745, 766
drilling down, 746
person selection, handling, 745
user selections, handling, 745-746
Contacts framework, 746-748
Contacts UI framework, 126, 744-745
container views, 712-713
Content Compression (Auto Layouts), 171
Content Compression Resistance (Auto Layout), 598
Content Hugging (Auto Layouts), 171, 597
contentSize attribute, 322
Continue icon, 888
controlHardware method, 681
controls
expanding, 607-612
constraints, configuring, 609-612
interface, creating, 607-608
repositioning based on orientation, 867
segmented, 293
adding, 300
connecting to actions, 307-309
iOS 6 *versus* iOS 7 appearance, 301
outlets, 306
segments, configuring, 300-301
sizing, 302
UISegmentedControl class, 139
convenience methods, 91, 278, 295
converting data types, 92
copies (images), creating, 266-267
copying/pasting text, 237
core application classes, 134-135
UIApplication, 134
UIControl, 135
UIResponder, 135
UIView, 135
UIViewController, 135
UIWindow, 134
Core Audio framework, 127
Core Bluetooth framework, 131
Core Data framework, 32, 129, 199, 579
Core Foundation framework, 129
Core Graphics framework, 128
Core Image framework, 127, 707
filters, applying, 708-709, 728-729
Programming Guide, 736
Core Location, 130, 781
accuracy, 783, 786
distance calculations, 795-796
errors, handling, 794
headings, 787-789
LocateMe Apple tutorial, 808
location manager
instance, creating, 793-794
preparing, 793

locations
authorization, requesting, 782
battery life, 787
location managers, 782
plist keys, 782
simulated, configuring, 798
updates, receiving, 782-784, 786
Core Motion, 130, 668-670
acceleration, handling, 683-684
attitude data, handling, 682-683
CMMotionManager, configuring, 669-670
interface-orientation changes, disabling, 685-686
motion manager, initializing, 680-681
motion updates, receiving, 669
radians to degrees conversion, 677
rotations, handling, 684-685
Core OS layer frameworks, 131-132
Accelerate, 131
Core Bluetooth, 131
External Accessory, 131
Local Authentication, 132
Security, 132
System, 132
Core Services frameworks, 129-131
Accounts, 129
Address Book, 129
CFNetwork, 129
Core Data, 129
Core Foundation, 129
Core Location, 130
Core Motion, 130
EventKit, 130
Foundation, 130
Health Kit, 130
Home kit, 130
Newsstand, 130
PassKit, 130
Quick Look, 131
Social, 131
StoreKit, 131
System Configuration, 131
Core Text framework, 128
Correction attribute, 236
Count Down Timer mode (date pickers), 418
count property, 89
counters
button presses, tracking, 488
current count, displaying, 480-481
dictionaries/arrays, 89
incrementing, 480
updates, 837
displaying with counter values, 488-489
triggering, 490
counting slowly app. *See* SlowCount app
CountingNavigationController class, 472
countUp method, 837
cover vertical transitions, 372
CPU usage, monitoring, 895
crash recovery (Simulator), 67
Create Story button, 246-247
createFlowerData method, 525
createStory method, 246-247, 252
cross dissolve transition, 372
Cupertino app
application logic
distance to Cupertino, calculating, 795-796
errors, handling, 794
location manager instance, creating, 793-794
location manager preparations, 793
.plist file update, 797
blur, adding, 798-799
outlets, 792
project setup, 789-790
background image resources, 789
location constants, adding, 790
running, 797
status bar appearance, 796
UI design, 790-792
Cupertino Audio Compass app, 828
audio directions, implementing, 829-832
heading updates audio feedback, adding, 830-831
sound file references, initializing, 829
audio files, adding, 828
AudioToolbox framework, adding, 828
background modes, 832
system sound ID variables, 829
Cupertino Compass app
application logic
heading to Cupertino, calculating, 804-805
heading updates, handling, 802, 805-807
recent locations, storing, 803-804
audio directions. *See* Cupertino Audio Compass app
outlet, 801
project setup, 799-800
running, 807
UI, 800-801
curly braces ({}), 45, 80
current context modal segues, 372
current date retrieval, 431
currentItem property, 716
customizing
annotations, 756-757
buttons, 243
attributes, 242-243
labels, 342-343
code, 42
assistant editor mode (Xcode), 48
tabbed editing, 49-50
constraints, 595-597
disabling, 529-530
keyboard display, 236-237

pickers
labels, 440
toolbars, 440
segues, 370
status bar appearance, 283-284
text field attributes, 233-235
text view attributes, 238-239
UIs
accessibility attributes, 174-175
connections, 185-186
object attributes, 173-174
variable values, 887
viewDidLoad method, 883
CustomPicker app
actions, 444
animal chooser scene, 439, 441-442
AnimalChooserViewController class, adding, 439
components
constants, 440
sizes, 448
custom picker view, 444-445
data source protocol, implementing, 445-446
delegate protocol, 446-448
image resources, adding, 439
initial scene, 441
outlet, 444
picker view data source/ delegate, 442-443
project setup, 438-440
row sizes, 448
running, 451
segues, 443
modal, dismissing, 450-451
popovers, 451
selections
default, 450
displaying, 448-450
reactions, 449-450
UI design, 440-443
view controller association, 439

D

d: date format, 432
darkening objects, 166
data
detectors, 239
models. *See* Core Data
sharing
navigation scenes, 464
tab bar scenes, 470
source protocols
custom picker views, implementing, 445-446
picker views, 420-421
tables. *See* tables
sources
picker view, 442-443
table outlets, connecting, 513-514
storage, 539, 542
direct file system access, 546-550
Playground, 550-551
settings bundles, 544-546
user defaults, 543-544
structures
creating, 524
populating, 525-526
type classes, 136-137
bridged, 136
nonbridged, 136
NSDate, 137
NSURL, 137
types
converting, 92
downcasting, 92
Swift, 85
datatips (variables), 887
dataUsingEncoding method, 550
Date and Time mode (date pickers), 418
Date mode (date pickers), 418
DateCalc app
actions, 430
application logic, 431
current date, retrieving, 431
date differences, calculating, 432-434
date output, updating, 434-435
date/time, displaying, 431-432
DateChooserViewController class, adding, 425
initial/date chooser scenes, configuring, 426
outputLabel outlet, 430
Playground, 435-436
project setup, 425-427
running, 437
segues, 436-437
UI design, 427-429
date chooser scene, 428-429
initial scene, 427-428
view controller association, 426
DateChooserViewController class, 425
dates/times
Apple tutorial, 453
date chooser scenes
creating, 428-429
date pickers, 426
date differences, calculating, 432-434
date pickers, 417-419, 424
actions, 430
appearance, configuring, 418
Apple tutorial, 452
constraints, 427-429
current date, retrieving, 431
date chooser scenes, 428-429
date differences, calculating, 432-434
date output, updating, 434-435
DateChooserViewController class, 425
date/time, displaying, 431-432
default calculation upon exiting, 434
initial/date chooser scenes, configuring, 426

locales, 418
modes, 418
outputLabel outlet, 430
Playground test, 435-436
segues, 429, 436-437
toolbars, creating, 427
UI design, 427-429
view controller association, 426
displaying, 431-432
NSDate class, 137
output, updating, 434-435
string formats, 432

debugging
debug navigator, 893-894
further exploration, 898
NSLog function, 878-879
calling, 879
output, displaying, 879-880
Xcode debugger. *See* Xcode debugger

declaring
arrays, 88
classes, 80
constants, 81, 92
defined, 80
dictionaries, 89
enumerations, 90
floating-point numbers, 86
IBOutlet, 81
import, 79-80
integers, 86
method, 82-83
objects, 90
optional values, 94
variable properties, 80-81
variables, 85

decrementing integers, 105

defer statements, 109-110

delegates
CN contact picker, 745-746
Mail compose, 750, 771
map view, 756-757
media picker, 696
navigation controllers, 707
picker views, 442-443
protocols
custom picker views, implementing, 446-448
location manager, 782-784
people picker, 765
picker views, 421-424
tables, 505-506, 517-518
table outlets, connecting, 513-514
text fields/text views, 254

deleting
annotations, 756
Auto Layout constraints, 634
breakpoints, 885
objects from memory, 110-111
Simulator devices, 68
UI
connections, 186
objects from views, 164
Xcode project files/resources, 37-38

describeInteger method, 883

desiredAccuracy property, 786

detail disclosure buttons, 243

detail scenes
sharing information with master scenes, 509
updating, 523

detail view controllers, configuring, 531-533

detailItem method, 532

developers
Apple Developer Program, 10-13
becoming, 2
requirements, 10
Windows options, 10
Xcode, installing, 13-15

development provisioning profiles
multiple devices, 7
overview, 15-16
viewing, 21

Device command (Xcode Hardware menu), 65

devices, 5-6
apps, running, 17-18
configuring for development, 16-18
connectivity, 9
graphics, 8
input/feedback, 9-10
locations. *See* locations
memory constraints, 9
model queries, 857
multitasking constraints, 9
orientations. *See* orientations, 56
processors, 8
resolution
iPads, 7
iPhones, 6
scaling, 7
rotation simulation, 64
screen sizes, 8
Simulator, adding, 67-69
size class representations, 858
universal apps
popovers, 404-407
vibrating for alerts, 354-355
Xcode, adding, 17

Devices command (Xcode Windows menu), 67

dictionaries, 85
classes, 136
contents, counting, 89
declaring, 89
iterating over, 106
key/value pairs, 89
soundSetting, 704
userMessages, 89
values
accessing, 89
assigning/modifying, 89

Did End on Exit event, 248-249

didSelectContact method, 745

didSelectContactProperty method, 746

direct file system access, 546-550
actions, 574-575
fields/text view outlets, 574
file paths, 548-549
keyboards, hiding, 575
reading/writing data, 549-550
results
displaying, 577-578
storing, 576-577
sandbox restrictions, 546-547
storage locations, 547-548

direction image resources, 799
directives (view controllers)
IBAction, 198
IBOutlet, 197
directories
Documents, 548
Library/Caches, 548
Library/Preferences, 547
tmp, 548
disabling
backgrounding, 821-822
editing, 529-530
interface-orientation changes, 685-686
Xcode Welcome screen, 30
Disconnected project
connections
actions, creating, 183
editing, 185-186
outlets, creating, 181-182
opening, 178
dismissDateChooser method, 436
dismissing
modal segues, 450-451
popovers, 385-386
segues, 378
dismissViewControllerAnimated method, 378
displaying
asset catalogs, 39
call stacks, 893
constraints, 594
counter values, 488-489
current count, 480-481
date/time, 431-432
document outline area, 232
image pickers, 705, 725-726
keyboards, 236-237
media pickers, 695, 732-733
Navigator (Playground), 117
NSLog output, 879-880
Object Library, 162
output (Playground), 114
people picker, 745, 766
picker view selections, 448-449
Playgrounds, 112
popovers on iPhones, 405
provisioning profiles, 21
Size Inspector, 168
storyboards, 157
survey results, 577-578
table cells, 516-517
UI
connections, 185
preview, 176-178
user locations, 768-769
variables
lists, 892
contents in Playground, 113-114
view controllers, 382
view hierarchy, 896-897
web views, 309-311, 313-314
Xcode
Quick Help Inspector, 148
templates, 30-31
distanceFilter property, 786
doAcceleration method, 684
doActionSheet method, 351
doAlert method, 345
doAlertInput method, 349
doAttitude method, 683
do-catch statements, 107-108
dock (storyboard scenes), 162
document outline area (IB), 232
document outline objects, 161
documentation (Xcode), 142
bookmarks, 145-146
browsing, 144
downloads, setting up, 142-143
navigating, 144-146
searching, 143-144
Documents directory, 548
doMultipleButtonAlert method, 347
Done button, 248-249
doRotation method, 685
doSound method, 354
dot notation, 81
Double data type, 85
doVibration method, 355
downcasting, 88-96, 92
downloading Xcode documentation, 142-143
dragging objects, 215

E

earlierDate method, 137
editing. *See* customizing
editing tools (IB), 165
alignment, 167-168
Arrangement, 167
guides, 165-166
selection handles, 166-167
Size Inspector, 168
editor (Xcode), 33
autocompleting code, 44-45
editing code, 42
Editor menu commands (Xcode)
Add Horizontal Guide/Add Vertical Guide, 166
Align, 167
Arrange, 167
Canvas, Show Selection Highlights, 166
Show Document Outline, 232
editor scenes, creating, 395-396
EditorViewController class, adding, 389
email, 748-749
composition
completion, handling, 750, 772-773
messages, composing, 772
view, 748
Mail compose delegate protocol, conforming, 771
Mail compose window, 749
Message UI framework, 749
emailField text field, 400
enabling
first responder view controllers, 651-653
interface orientation changes, 586-588
screen orientation changes, 616
task completion backgrounding, 837-839
endBackgroundTask method, 816
enumerations, 89-90
creating, 90
declaring, 90
values, 90

equal to (==) expression, 103
errors, 52
Auto Layout constraints, fixing, 598-601
code, 52-55
Core Location, 783
guard statements, 109
location, handling, 794
Playground, 116
Swift, 107
defer, 109-110
do-catch statements, 107-108
throw statements, 108-109
Euler angles, 683
event handling
alerts
actions, 334
buttons, 344-345
Did End on Exit, 248-249
motion, 130
orientation notifications, requesting, 667-668
Quick Actions, 846-847
segue navigation, 530-531
table row selections, 517-518
touches
background, 249-250
buttons, 621
multitouch, 64
EventKit framework, 130
EventKit UI framework, 126
exclamation points (!), 748
Exit icon (storyboards), 159
exits
connecting, 379-380
defined, 363
view controller preparations, 378-379, 393
expressions, 103
extensions, 76
External Accessory framework, 131
external device movie playback support, 700

F

Facebook posts, preparing, 751
feedback mechanisms, 9-10
FieldbuttonFun app, 225
action, creating, 246-247
application logic, implementing, 251-252
button templates, creating, 227-232
images, adding, 227-228
slices, creating, 228-231
connections, planning, 227
createStory method, implementing, 252
keyboard, hiding, 247-251
background touch, 249-250
code, adding, 250-251
Done button, 248-249
outlets, connecting, 245-246
Playground, 251
running, 252
setting up, 226
UI design
button attributes, editing, 242-243
buttons, adding, 241
custom button images, 243-245
plain *versus* attributed text, 235
scrolling options, 241
simulated screen size, 232
text field attributes, editing, 233-235
text fields, adding, 232-233
text input traits, 236-237
text view attributes, editing, 238-239
text views, adding, 237
variables, planning, 227
fields
alerts, adding, 349-350
direct file system access outlets, 574
survey, storing, 576-577
text. *See* text fields
user input, 206
File menu commands (Xcode)
New, Playground, 111
New, Tab, 49
New Project, 30
file paths, 548-549
fileExistsAtPath method, 549
fileHandleForReadingAtPath method, 550
fileHandleForUpdatingAtPath method, 550
fileHandleForWritingAtPath method, 550
files
asset catalogs, adding, 39
class, 202-203
importing (Swift), 79-80
storyboard, 203-205
Xcode projects
adding, 35-36
deleting, 37-38
filters
Core Image, 708-709, 728-729
location updates, 786
media picker, 695
Xcode project navigator, 35
finding methods/properties in code, 43
fireDate property, 825
First Responder icon (storyboards), 159
first responder view controllers, 651-653
fixing Auto Layout constraint errors, 598-601
FIXME comments, 47
flip horizontal transitions, 372
Float (floating point numbers) data type, 85-86, 136
floatForKey method, 557
FlowerColorTable app
application logic, 514-518
cells to display, configuring, 516-517
flower arrays, populating, 514
number of sections, returning, 515

row selections, handling, 517-518
section headings, 515
section row counts, returning, 515
delegate/data source outlets, connecting, 513-514
project setup, 510-511
image resources, adding, 510
section constants, adding, 511
running, 518
UI design, 511-512

FlowerDetail app
3D Touch peek and pop gestures, adding, 657-659
detail view controller, configuring, 531-533
flower data structures
creating, 524
populating, 525-526
master view controller, 527
editing, disabling, 529-530
number of section rows, retrieving, 527
number of sections, retrieving, 527
section labels, 527
segue navigation events, handling, 530-531
table cells, creating, 528-529
table view data source methods, 527-528
project setup, 520-521
hierarchy, 520-521
image resources, adding, 520
running, 533
UI design, 521-523
detail scene, updating, 523
master scene, updating, 522
web view outlet, connecting, 523

flowerDetailView web view, 303

flowerView web view, 303

FlowerWeb app
actions, 307-309
application logic, 309-314
Detail web view, hiding/ displaying, 309-311
flower images, loading, 311-313
initial display, updating, 313-314
blur visual effect, 314-315
outlets, 306
running, 315
setting up, 299
UI, 299
button, 305
color choice segmented controls, 300-302
flower details switch, 302-303
web view attributes, 304-305
web views, adding, 303-304

fonts, 865

for loops, 105-106

form sheet modal segue, 372

forward geocoding, 757-758

Foundation framework, 130

foundPinch method, 647

foundRotation method, 649

foundSwipe method, 646

foundTap method, 629, 645

frameworks
AudioToolbox
adding, 828
importing, 353
AV Foundation, 698-699
AV Kit, 698-699
external device playback/ AirPlay support, 700
loading/displaying movies, 699-700
playback completion, handling, 701-702
playback controls, 701
screen placement, 700
Cocoa Touch, 126-127
Contacts UI, 126
EventKit UI, 126
GameKit, 126
iAd, 127
MapKit, 126
Message UI, 126
Notification Center, 127
TextKit, 127
UIKit, 126
Contacts, 746-748, 776
Contacts UI, 744-745
Core Image, 707
Core Location. *See* Core Location
Core OS. *See* Core OS layer frameworks
Core Services. *See* Core Services frameworks
defined, 123
Map Kit, 123, 753
Media layer, 127-129
AV Foundation, 127
AVKit, 127
Core Audio, 127
Core Graphics, 128
Core Image, 127
Core Text, 128
Game Controller, 128
Image I/O, 128
Metal, 128
OpenGL ES, 128
Photos, 128
Photos UI, 128
Quartz Core, 129
SpriteKit, 129
Media Player, 694
classes, 694
media picker configuration, 694-695
Media Player. *See* Media Player framework
Message UI, 749
projects, adding, 710
Social, 751, 777

freeform sizing (scrolling views), 321

frosted glass appearance, 561

full screen modal segues, 372

functions
See also methods
methods, compared, 198

NSLog, 878-879
calling, 879
output, displaying, 879-880
print function, compared, 878
print *versus* NSLog, 878
setObject:forKey, 543
user defaults, reading/writing, 543

G

Game Controller framework, 128
GameKit framework, 126
geocoding, 757-759
forward, 757-758
mapping, 769-770
Playground, 759-760
reverse, 758-759
geographic north, 787
gestures
3D Touch peek and pop
adding, 631-632
editor screen, 655-657
implementing, 654-655
overview, 630-631
table views, 657-659
testing, 633
built-in, 641
image views
replacing, 644-645
resetting to default, 635
outlets, 642
recognizers
adding, 628-629
classes, 628
pinch, 640-641
rotation, 641
shake, 651-653
swipe, 639-640
tap, 638-639
responding
actions, 642-644
pinching, 646-649
rotations, 649-650
shake, 653
swipe, 646
tap, 645-646
SimpleGestureRecognizers tutorial project, 659
swipes, 627
views, creating, 636-637
Gestures app
actions, 642-644
application logic, 644
image view, replacing, 644-645
pinch recognizer response, 646-649
rotation recognizer response, 649-650
shake gesture response, 653
shake recognizer implementation, 651-653
swipe recognizer response, 646
tap gesture recognizer response, 645-646
gesture recognizers, adding
pinch, 640-641
rotation, 641
swipe, 639-640
tap, 638-639
interface, designing, 636-637
outlets, 642
overview, 633-634
project setup, 634-635
image resources, 635
image views, resetting to default, 635
running, 654
getFlower method, 312
getters, 81
GettingAttention app
action sheets, implementing, 350-352
actions, 344-345
alert sounds with vibrations, 354-355
alerts, implementing, 345-346
fields in alerts, adding, 349-350
local notifications, registering, 826-827
multibutton alerts, creating, 346-348
outlet, 343
project setup, 340
sound resources, adding, 340
UI design, 342-343
g-forces, 664
graphics
See also images
displaying, 8
frameworks
Core Graphics, 128
Metal, 128
SpriteKit, 129
groups
preferences, 544
tables, 498
Xcode projects, 35
guard statements, 109
guides, 165-166
gutter (Xcode), 883
gyroscopes, 663, 666-667
attitude data, handling, 682-683
radians per second, 666

H

handleButton method, 621
handling
acceleration data, 683-684
alerts
actions, 334
buttons, 344-345
attitude data, 682-683
background
fetch completion, 842-843
suspension, 823
button touches, 621
contact picker selections, 766-768
Core Location errors, 783, 794
device rotations, 684-685
email composition completion, 750
errors. *See* errors

heading updates, 805-807
image pickers, 706-707, 726-727
mail composition completion, 772-773
media picker selections, 696
music player
empty selections, 733-734
user selections, 733
orientation notifications, requesting, 667-668
people pickers
cancel action, 745
drilling down, 746
person selection, 745
selections, 745-746
playback completion
AV Audio Player, 703
video player, 701-702
Quick Actions events, 846-847
segue navigation, 530-531
table row selections, 517-518
video player cleanup, 718
Hardware menu commands (Xcode)
Device, 65
Home, 65
Lock, 65
Reboot, 65
Shake Gesture, 65
Simulate Hardware Keyboard, 66
Simulate Memory Warning, 66
Toggle In-Call Status Bar, 66
Touch ID Enrolled/Simulate Finger Touch, 65
TV Out, 66
headingfilter property, 787
headings, 787-789
destinations, calculating, 804-805
table sections, 515
updates, 830-831
handling, 805-807
requesting, 802
HealthKit framework, 130
height
cells, 501
picker components/rows, 422
popovers, 375-376
scrolling views, 323
view picker components, 448
HelloNoun app
application logic, 216-217
class files, 202-203
interface
objects, adding, 209-212
simulated attributes, 208
setting up, 201-202
storyboard file, 203-205
variable properties, 205-206
view/view controller connection, 212-215
actions, 215
outlets, 213-215
help
IB, 188-189
images, 285
MVC, 218
Swift, 118
toolbars, 452
view controllers, 407
Xcode, 24, 147-149
Help, Quick Help menu commands (Xcode), 147
hh: date format, 432
hideKeyboard action, 248
hiding
keyboards, 247-251, 401
background touch, 249-250
code, adding, 250-251
Done button, 248-249
file system preferences, 575
web views, 309-311
hierarchy
split view controllers, 520-521
storyboard scenes, 158
views, displaying, 896-897
high-pass filters, 687
Hint attributes, customizing, 174
Home command (Xcode Hardware menu), 65
HomeKit framework, 130
Hop button, 271
hopping bunnies animation. *See* ImageHop app
horizontal constraints (Auto Layout), 593
horizontal size class, 858
horizontal stack views, 319

I

iAd framework, 127
IB (Interface Builder), 155-156
accessibility attributes, 173-174
Attributes Inspector, 173-174
buttons, 242-243
custom button images, 244-245
text fields, 233-235
text views, 238-239
Auto Layout. *See* Auto Layouts
Connections Inspector, 183
default screens, 162
document outline area, 161, 232
editing tools
alignment, 166-167
Arrange menu, 167
guides, 165-166
selection handles, 166-167
further exploration, 188-189
gesture recognizers, adding, 629
Identify Inspector, 187
layout constraints, manually defining, 601-604
Object Library, 162-163
Quick Inspector, 185-186
Size Inspector. *See* Size Inspector
storyboards. *See* storyboards
UI objects. *See* UI, objects
IBAction directive, 198
IBOutlet declaration, 81

IBOutlet directive, 197
icons
application, 57-60
Quick Actions, 819, 844
scrolling view resources, adding, 316
stack views, adding, 318
storyboards, 158-159
templates, 819
universal applications, 855
Xcode debugger, 888
Identifier attributes, customizing, 174
identifiers
bar button items, 416
cells, 502
objects, configuring, 187-188
storyboards, 381
Identity Inspector, 187
Accessibility settings, 174-175
object identities configuring, 187-188
if-then-else statements, 104-105
Image I/O framework, 128
ImageHop app
actions, 264, 275-276
animation resources, adding, 263
application logic
animated image view, implementing, 277-278
incrementing animation speed, 282-283
setting animation speed, 280-281
starting/stopping animations, 278-280
background
image, 271-272
suspension, implementing, 823
blur visual effect, 272-273
outlets, 264, 274-275
Playground, 281-282
project setup, 263
Quick Actions
events, handling, 846-847
icons, adding, 844
.plist file configuration, 844-845
running, 847
running, 284
status bar appearance, 283-284
UI, 264
default images, setting, 265-266
Hop button, 271
image copies, creating, 266-267
image view, adding, 264
slider, adding, 267-268
slider range attributes, 268-269
speed output labels, 271
steppers, adding, 269-270
imagePickerControllerDidCancel method, 707, 726
imagePickerController:didFinishPickingMediaWithInfo method, 706, 726
images
animations. *See* animations
background
adding, 271-272
blur visual effect, adding, 272-273
buttons
custom, 243-245
templates, adding, 227-228
cells, adding, 502
copies, creating, 266-267
Core Image, 707
filters, 708-709, 728-729
Programming Guide, 736
frameworks
Core Image, 127
Image I/O, 128
Photos, 128
Photos UI, 128
further exploration, 285
launch, 60
pickers, 705
cleanup, handling, 726-727
configuring/displaying, 705-706
custom, 439
displaying, 725-726
further capabilities, 736
navigation controller delegate, 707
preparing, 724
selections, handling, 706-707, 726
resources
direction, 799
Gestures app, 635
settings bundles, 560
split view controllers, adding, 520
tables, adding, 510
selector preferences, creating, 563-565
size classes, 866
stack view icons, adding, 318
tab bars, 468
views, 261
animated, implementing, 277-278
animation loops, adding, 264
animationDuration property, 269
default, resetting, 635
replacing, 644-645
transformations in Playground, 650-652
imperative programming, 74
implicit preferences, 540
actions, 554
creating, 555
key constants, 552
project setup, 552
reading, 557
sliders, 554
storing, 556-557
toggle switches, 554
import declaration, 79-80
importing files (Swift), 79
incrementCount method, 480
incrementCountFirst method, 490
incrementing
animation speed, 282-283
integers, 105
incrementSpeed method, 283
index values (arrays), 88

info light buttons, 243
Information Property List Key Reference tutorial, 579
inheritance (OOP), 75
initial scenes
  creating, 393-395
  date pickers, 426
  labels, setting, 401
  unwinding back to, 398
initInterface method, 617
initNamed method, 278
initWithMediaTypes method, 729
initWithString method, 91
input/output
  alerts. *See* alerts
  buttons. *See* buttons
  data detectors, 239
  image views, 261
  input mechanisms, 9-10
  keyboards, hiding, 247-251
    background touch, 249-250
    code, adding, 250-251
    Done button, 248-249
  labels. *See* labels
  Mad Libs-style story creator. *See* Mad Libs-style story creator
  pickers. *See* pickers
  Playground
    displaying, 114
    generating, 114
    history, 114-115
  scrolling, 297
    enabling, 317-318, 322-323
    freeform sizing, 321
    icon resources, adding, 316
    outlet, 322
    pagination, 318
    stack views, adding, 318-321
    text view options, 241
    width/height values, 323
  segmented controls, 293
    adding, 300
    connecting to actions, 307-309
    iOS 6 *versus* iOS 7 appearance, 301
    outlets, 306
    segments, configuring, 300-301
    sizing, 302
  sliders. *See* sliders
  stack views, 297
    adding, 318-321
    horizontal, 319
    icon images, adding, 318
    vertical, 320
  steppers, 260-261
    animation loop speed, 269-270
    outlets, 275
    trigger action, 276
  switches. *See* switches
  System Sound Services, 337-339
  text fields. *See* text, fields
  touch. *See* touches
  visual effect views, 261
  web views. *See* web views
Inspector (Xcode Quick Help), 148
installing Xcode, 13-15
instances
  defined, 76
  methods, 76
  UILabel class, creating, 90
instantiateViewControllerWithIdentifier method, 381
instantiation, 76
integers (Int), 85, 86
  classes, 136
  incrementing/decrementing, 105
integration, 743
  Contacts, 744
    CNContactPickerViewController protocol, 765
    contact selections, handling, 766-768
    Contacts framework, 746-748
    Contacts UI framework, 744-745
    people picker, displaying, 745
    picker view, displaying, 766
    user selections, handling, 745-746
  email, 748-749
    composition completion, handling, 750, 772-773
    composition view, 748
    Mail compose delegate protocol, conforming, 771
    Mail compose window, 749
    Message UI framework, 749
    messages, composing, 772
  mapping, 753
    annotations, 755-757, 770-771
    display, controlling, 769-770
    geocoding, 757-759
    locations. *See* locations
    Map Kit, 753
    map views, adding, 753
    regions, 754
  Safari web browser, 752-753, 775
  social networking, 750-752, 773-774
interface (Xcode), 33-34
Interface Builder. *See* IB
interfaces. *See* UIs
interpolation, 87
Intrinsic Size setting (Auto Layouts), 172
iOS Design Cheat Sheet by Ivo Mynttinen website, 60
iOS devices. *See* devices
iPads
  multitasking, 859
  resolution, 7
iPhones
  popovers, displaying, 405
  resolution, 6
isAnimating property, 279
isAvailableForServiceType method, 751
issue navigator, 52-55
iterating over collections, 106

K

Keyboard Type attribute, 236
keyboards
- display, editing, 236-237
- hiding, 247-251, 401
  - background touch, 249-250
  - code, adding, 250-251
  - Done button, 248-249
  - file system preferences, 575
- simulation, 66

keys
- constants
  - implicit preferences, 552
  - settings bundles, 559
- dictionaries, 89
  - accessing, 89
  - assigning/modifying, 89
- .plist file, 782
- iterating over, 106

L

labels, 224
- animation speed output, 271
- attributes, customizing, 174
- blur effect, 561
- buttons, editing, 342-343
- cells, 502
- custom pickers, 440
- initial scenes setting, 401
- master view controller sections, 527
- navigation items, 477
- sliders, 267
- tab bar items, 485-486
- text
  - fields, adding, 233
  - orientation changes, 672
  - views, adding, 237
- UILabel class, 138
- user output, 206
- view controllers, 391
- visual effects, editing, 272

Landscape Left screen orientation constant, 586
Landscape Right screen orientation constant, 586
landscape *versus* portrait orientations, 620
lastName constant, 92
launch images/screens, 60, 856
launching. *See* starting
layouts
- Auto Layout. *See* Auto Layouts
- margins, 169
- tables, 503-504
- UIs, 165
  - Auto Layouts, 169-172
  - aligning, 167-168
  - arranging, 167, 169
  - constraints, 169
  - guides, 165-166
  - layout margins, 169
  - selection handles, 166-167

LetsNavigate app
- action, 479
- application logic
  - current count, displaying, 480-481
  - pushing scenes, 479-480
- navigation controller configuration, 472-473
- outlet, 478
- project setup, 471-472
- running, 481
- scenes, adding, 473-474
- show segues, creating, 476
- UI design, 477
- view controller
  - association, 473-474
  - classes, adding, 472

LetsTab app
- actions, 487
- application logic, 488
  - count button presses, tracking, 488
  - counter updates, triggering, 490
  - display updates with counter values, 488-489
  - tab bar item badges, updating, 489
- outlets, 487
- project setup, 482-484
- running, 490
- scene connections, 484-485
- scenes, adding, 483
- tab bar controllers, adding, 483
- UI design, 485-486
- view controller association, 483

libraries
- Object, 162-163
  - adding objects to views, 164
  - deleting objects from views, 164
  - displaying, 162
- Xcode, 163

Library/Caches directory, 548
Library/Preferences directory, 547
life cycles (applications)
- background-aware methods, 820-821
- diagram, 132
- tracing, 133

listings
- 3D Touch peek and pop, adding, 631
- adaptive segues, disabling, 406
- alert action handlers, 334
- animations
  - loading, 277-278
  - starting/stopping, 279
- annotations
  - adding, 755
  - customizing, 756
- applicationWillEnterForeground method, 823
- audio player
  - implementing, 722
  - playback, controlling, 723
  - recorded sounds, loading, 723
- audio recorder
  - controlling, 721
  - implementing, 720
- AV Audio Player playback completion, 703

background fetches
  completion, handling, 842
  minimum fetch intervals, defining, 841
background processing, starting, 838
canBecomeFirstResponder method, 651
chooseImage method, 725
CMMotionManager, 669, 680
contact picker selections, handling, 767
controlHardware method, 681
Core Image filters, 708, 728
Core Location errors, 785
counter updates, 837
createStory method, implementing, 252
data structures, populating, 525
date differences, calculating, 433-434
date pickers
  first displayed default calculation, 435
  modal scenes, dismissing, 436
  popover size, 437
describeInteger method, 883
detail view controller, configuring, 532
doAcceleration method, 684
doActionSheet method, 351
doAlert method, 345
doAlertInput method, 349
doAttitude method, 683
doMultipleButtonAlert method, 347
doRotation method, 685
doSound method, 354
doVibration method, 355
email composition completion, handling, 750
ending background processing, 838
exit points, 379
Facebook post, preparing, 751
foundPinch method, 647
foundRotation method, 649
foundSwipe method, 646
foundTap method, 645
geocoding
  forward, 757
  reverse, 758
Gestures app, image views, replacing, 644
getFlower method, 312
heading updates, 788
  audio feedback, adding, 830
  handling, 806
  requesting, 802
headings to destinations, calculating, 804
image pickers
  cleanup, handling, 727
  configuring/displaying, 706
  selections, handling, 706, 726
incrementCount method, 480
incrementCountFirst method, 490
incrementSpeed method, 283
initial scene labels to editor scene field, setting, 401
interface orientations changes, enabling, 617
interface rotations
  activating, 587
  disabling, 686
keyboards, hiding, 250, 401, 575
loading preferences upon startup, 570
loading/playing sounds, 338
local notifications
  authorization requests, 825
  registering, 826
location managers
  distance calculations, 795
  errors, handling, 794
  instance, creating, 793
location updates, receiving, 783
mail compose view, 772-773
Mail compose window, 749
maps
  annotations, customizing, 771
  centering, 769
media pickers
  displaying, 732
  selections, handling, 696
  setup/display, 695
motionEnded:withEvent method, 653
music player
  playing music, 734
  preparations, 731
  selections, handling, 733-734
newBFF method, 766
NSLog function, calling, 879
orientation changes
  responding, 674
  updates, registering, 673
people picker
  cancel action, 745
  drilling down, 746
  person selection, handling, 746
picker data source protocol, 420
picker delegate protocol, 421
picker views
  custom views, 423, 447
  data, loading, 445
  modal segues, dismissing, 451
  number of components, 445
  number of elements per component, 446
  popovers, dismissing, 451
popovers
  dismissal, 386
  presentation controller delegate, 406
  size, 403
preferences, updating, 557
preferredStatusBarStyle method, 284
prepareForSegue method, 387
Quick Action events, handling, 846
recent locations, storing, 803

responsive interfaces
    button touches, 621
    drawing upon launching, 620
    objects, initializing, 618
    rotations, handling, 621
    updating, 619
Safari web views, opening, 775
screen space changes, reacting, 589
scrolling, enabling, 323
segue navigation event handling, 531
setBackgroundHueValue method, 555-556
setOutput method, 217
setSpeed method, 280
setValuesFromPreferences method, 569
sound file references, initializing, 829
status bar appearance, 776, 797
storeSurvey method, 576
survey results, displaying, 577
Swift class file, 78-79
    class declaration, 80
    constant declaration, 81
    ending, 83
    IBOutlet declaration, 81
    import declaration, 79-80
    method declarations, 82-83
    variable property declarations, 80-81
tab bar item badges, updating, 489
tables. *See* tables
tap gesture recognizer, adding, 629
text fields, populating, 400
timers, initializing, 836
toggleFlowerDetail method, 311
tweet compose view, 774
UI editing, disabling, 530
UIActionSheet class, implementing, 335
UIAlertController object, implementing, 331-333
unwind segue placeholder, 393
updateCounts method, 488
user location authorization, 768
video player
    cleanup, 718
    loading/preparing, 717
    playback completion, handling, 701
view controller first responders, 653
view picker selections
    components/row sizes, 448
    default, 450
    displaying, 448-450
    reacting, 449-450
ViewController.swift
    connections, 206
    outlets/actions, 276
viewDidLoad method, 44, 314, 883
viewWillAppear:animated method, 480
watchpoints, configuring, 890
lldb, 881-882
    activity to monitor, adding, 883
    breakpoints
        configuring, 885-886
        navigator, 893
    Continue to Here, 890
    CPU/memory usage, monitoring, 895
    navigators, 893-894
        breakpoints, 893
        debug, 893-894
    source code line numbers, 883
    stepping through code, 888-890
    variables
        examining, 887
        lists, accessing, 892
        values, editing, 887
    view hierarchy, displaying, 896-897
    watchpoints, 890-891
    Xcode gutter, 883
loadMovie method, 717
loadRequest method, 313
Local Authentication framework, 132
local notifications
    authorization, requesting, 824-825
    creating, 826-827
    properties, 825
    scheduling, 814-815, 827
LocateMe Apple tutorial, 808
location managers, 782
    delegate protocol, 782-784
    distance calculations, 795-796
    headings, 787-789
    instances, creating, 793-794
    locations. *See* Core Location, locations
    preparing, 793
locationManager:didFailWithError method, 785, 794
locationManager:didUpdateHeading method, 788
locationManager:didUpdateLocations method, 795
locationManager:didUpdateToLocations method, 783
locationManager:ShouldDisplayHeadingCalibration method, 789
locations
    Core Location. *See* Core Location
    data storage, 547-548
    displaying, 768-769
    distance calculations, 795-796
    errors, handling, 794
    recent, storing, 803-804
    retrieving
        accuracy, 783, 786
        authorization, requesting, 782
        battery life, 787
        errors, 783
        location managers, 782
        .plist keys, 782
        updates, receiving, 782-784

simulated, configuring, 798
updates, filtering, 786
locations (maps), 760-761
Lock command (Xcode Hardware menu), 65
logic problems, 52
long pressing gesture class, 628
long-running background tasks. *See* task completion backgrounding
loops, 105
for, 105-106
animation. *See* animations, looping
condition-based, 106-107
low-pass filters, 687

M

Mad Libs-style story creator, 225
action, creating, 246-247
application logic, implementing, 251-252
button templates, creating, 227-232
images, adding, 227-228
slices, creating, 228-231
connections, 227
createStory method, implementing, 252
keyboard, hiding, 247-251
background touch, 249-250
code, adding, 250-251
Done button, 248-249
outlets, connecting, 245-246
Playground, testing, 251
project setup, 226
running, 252
UI design
buttons, 241-243
custom button images, configuring, 243-245
plain *versus* attributed text, 235
scrolling options, 241
simulated screen size, 232
text field attributes, editing, 233-235
text fields, adding, 232-233
text input traits, 236-237
text view attributes, editing, 238-239
text views, adding, 237
variables, planning, 227
magnetic compass
direction image resources, 799
headings
to destinations, calculating, 804-805
updates, handling, 802, 805-807
recent locations, storing, 803-804
magnetic north, 787
Mail compose view controller, 749
delegate, 750
delegate protocol, conforming, 771
mailComposeController:didFinish WithResult method, 771
mailComposeController:didFinish WithResult:error method, 750
Map Kit framework, 123, 126, 753
map views, adding, 753
mapping, 753
annotations, 755-757
adding, 755
customizing, 756-757, 770-771
deleting, 756
audio directions, 828
audio files, adding, 828
AudioToolbox framework, adding, 828
background modes, 832
heading update audio feedback, adding, 830-831
sound file references, initializing, 829
system sound ID variables, 829
Core Location, 781
display, controlling, 769-770
distance calculations, 795-796
errors, handling, 794
geocoding, 757-759
forward, 757-758
Playground, 759-760
reverse, 758-759
geographic north *versus* magnetic north, 787
headings, 787-789
LocateMe Apple tutorial, 808
locations, 760-761
accuracy, 783, 786
authorization, requesting, 782
battery life, 787
errors, 784-786
location managers, 782
.plist keys, 782
recent, storing, 803-804
simulated, configuring, 798
updates, 782-784, 786
magnetic compass
heading updates, handling, 802, 805-807
headings to destinations, calculating, 804-805
recent locations, storing, 803-804
Map Kit, 753
map views, adding, 753
regions, 754
user locations, displaying, 768-769
MARK comments, 47
Markdown documentation, 101-102
master scenes
sharing information with detail scenes, 509
updating, 522
master view controllers, 527
editing, disabling, 529-530
number of section rows, retrieving, 527
number of sections, retrieving, 527
section labels, 527

segue navigation events, handling, 530-531
table cells, creating, 528-529
table view data source methods, 527-528
Master-Detail Application template, 509, 519
MasterViewController class, 527
media
AddMusic Apple tutorial, 736
audio playback
controlling, 723
implementing, 722-723
recorded sounds, loading, 723-724
audio recorder
controlling, 721-722
implementing, 720-721
microphone prompt, 722
AV Audio Player
audio files, playing, 702
customizing, 736
playback completion, handling, 703
playback controls, 702
AV Audio Recorder, 703-705
AV Foundation/AV Kit frameworks, 698-699
Core Image, 707
filters, 708-709
Programming Guide, 736
files, adding, 710
formats supported, 699
frameworks. *See* frameworks
images. *See* images
items
accessing, 697-698
media picker selections, handling, 696
music player
empty selections, handling, 733-734
media picker, displaying, 732-733
media picker preparations, 730
playing music, 734-735
playlist, creating, 733
preparations, 730-731
pickers. *See* pickers, media
video player. *See* video player
Media Player framework, 694
classes, 694
media items, accessing, 697-698
media pickers
configuring, 694-695
displaying, 695
filters, 695
iPad popover enforcement, 707
selections, handling, 696
media player, 697
mediaPickerDidCancel method, 696
mediaPicker:didPickMediaItems method, 696
MediaPlayground app
actions, 715
audio playback
controlling, 723
implementing, 722-723
recorded sounds, loading, 723-724
audio recording
audio recorder, implementing, 720-721
controlling, 721-722
Core Image filters, applying, 728-729
image pickers
cleanup, handling, 726-727
displaying, 725-726
preparing, 724
selections, handling, 726
music library access
empty selections, handling, 733-734
media picker, displaying, 732-733
media picker preparations, 730
playlist, creating, 733
preparations, 730-731
outlets, 715
playing music, 734-735
project setup, 710
frameworks, adding, 710
media files, adding, 710
variables/connections, planning, 710-711
UI design, 711-713
video player
cleanup, handling, 718
implementing, 716
movie playback, 717-718
memory
constraints, 9
management
ARC (Automatic Reference Counting), 110-111
objects, removing, 110-111
usage, monitoring, 895
warning simulation, 66
Message UI framework, 126, 749
Metal framework, 8, 128
methods
See also functions
addTextFieldWithConfigurationHandler, 333
append, 88
applicationDidBecomeActive, 821
applicationDidEnterBackground, 133, 821
application:didFinishLaunchingWithOptions, 821, 824
application:performActionForShortcutItem, 846
application:performFetchWithCompletionHandler, 842
applicationWillEnterForeground, 821, 823
applicationWillResignActive, 821
applicationWillTerminate, 821
availableData, 550
background-aware, 820-821
beginBackgroundTaskWithExpirationHandler, 816
beginGeneratingDeviceOrientationNotifications, 667
boolForKey, 557
calculateDateDifference, 433

canBecomeFirstResponder, 651
chaining, 98-99
chooseImage, 725
closures, 101
composeViewControllerForServiceType, 751
contactPickerDidCancel, 745
controlHardware, 681
convenience, 91
convenience initialization, 278, 295
countUp, 837
createFlowerData, 525
createStory, 246-247, 252
dataUsingEncoding, 550
declaring, 82-83
describeInteger, 883
detailItem, 532
didSelectContact, 745
didSelectContactProperty, 746
dismissDateChooser, 436
dismissViewControllerAnimated, 378
doAcceleration, 684
doActionSheet, 351
doAlert, 345
doAlertInput, 349
doAttitude, 683
doMultipleButtonAlert, 347
doRotation, 685
doSound, 354
doVibration, 355
earlierDate, 137
endBackgroundTask, 816
fileExistsAtPath, 549
fileHandleForReadingAtPath, 550
fileHandleForUpdatingAtPath, 550
fileHandleForWritingAtPath, 550
finding in code, 43
floatForKey, 557
foundPinch, 647
foundRotation, 649
foundSwipe, 646
foundTap, 629, 645
functions, compared, 198
getFlower, 312
handleButton, 621
imagePickerControllerDidCancel, 707, 726
imagePickerController:didFinishPickingMediaWithInfo, 706, 726
incrementCount, 480
incrementCountFirst, 490
incrementSpeed, 283
initInterface, 617
initNamed, 278
initWithMediaTypes, 729
initWithString, 91
instance, 76
instantiateViewControllerWithIdentifier, 381
isAvailableForServiceType, 751
loadMovie, 717
loadRequest, 313
locationManager:didFailWithError, 785, 794
locationManager:didUpdateHeading, 788
locationManager:didUpdateLocations, 795
locationManager:didUpdateToLocations, 783
locationManager:ShouldDisplayHeadingCalibration, 789
mailComposeController:didFinishWithResult, 771
mailComposeController:didFinishWithResult:error, 750
Markdown documentation, 101-102
mediaPickerDidCancel, 696
mediaPicker:didPickMediaItems, 696
motionEnded:withEvent, 653
multiple return values, 100
myInstanceMethod, 95
named parameters, 98
newBFF, 766
numberOfComponentsInPicker, 445
numberofComponentsInPickerView, 420
numberOfSectionsInTableView, 503, 515
openWeb, 775
orientationChanged, 674
parameters, 76
picker view delegate protocol, 422-424
pickerView:didSelectRow:inComponent, 421, 448
pickerView:numberOfRowsInComponent, 420, 446
pickerView:rowHeightForComponent, 422, 448
pickerView:titleForRow:forComponent, 421
pickerView:viewForRow:forComponent:reusingView, 446
pickerView:viewForRow:viewForComponent:ReusingView, 422
pickerView:widthForComponent, 422, 448
playAudio, 723
playMovieFinished, 718
playMusic, 734
preferredStatusBarStyle, 284
prepareForSegue, 387, 406
presentViewController, 382
previewingContext:commitViewController, 631
previewingContext:viewControllerForLocation, 631
recordAudio, 720
registerForPreviewingWithDelegate:sourceView, 631
registerUserNotificationSettings, 815
removeConstraints, 634
scheduleLocalNotification, 815, 827
sendEmail, 772
sendTweet, 774
setBackgroundHueValue, 556
  implementing, 555
  switches/sliders, connecting, 554
setDisplayName, 101
setMinimumBackgroundFetchInterval, 842

setOutput, 217
setQueueWithItemCollection, 697, 730
setSpeed, 280
setToRecipients, 749
setValuesFromPreferences, 569
showAddress, 769
showResults, 577
startUpdatingHeading, 787
stopDeviceMotionUpdates, 670
stopUpdatingLocation, 782
storeSurvey, 576
storyboardWithName, 381
stringFromDate, 431
stubs, 187
supportedInterface Orientations, 586, 616
syntax, 96-97
systemMusicPlayer, 730
table view data source, 503, 527-528
tableView:canEditRowAtIndex Path, 530
tableView:cellForRowAtIndex Path, 503, 516, 528
tableView:didSelectRowAt IndexPath, 505
tableView:numberOfRowsIn Section, 503, 515
tableView:titleForHeaderIn Section, 503, 515
timeIntervalSinceDate, 432
titleForSegmentAtIndex, 293
toggleAnimation, 279
toggleFlowerDetail, 311
type, 76
updateBadge, 489
updateCounts, 488
updateEditor, 401
updateLayoutWithScreen Width:screenHeight, 619-620
valueForProperty, 697
viewDidLoad, 44, 314
viewWillAppear, 400
viewWillAppear:animated, 480
viewWillTransitionToSize:with TransitionCoordinator, 589, 621
whiteColor, 555
writeData, 550

MFMailComposeViewController Delegate, 771
microphone prompt (audio recording), 722
mixing code development methodology, 194
mm date format, 432
MMMM: date format, 432
modal segues, 371-373
- creating, 396-398
- date pickers, 436
- dismissing, 450-451
- manually triggering, 377-378

modal UI elements, 331
modal views, 363
ModalEditor app
- 3D Touch peek and pop gestures, adding, 655-657
- actions, 399
- application logic, 400-401
  - emailField text field, populating, 400
  - initial scene label to editor scene field, setting, 401
  - keyboard, hiding, 401
- editor scenes, creating, 395-396
- EditorViewController class, adding, 389
- initial scenes, creating, 393-395
- modal segues, creating, 396-398
- outlets, 399
- popover segues, configuring, 402-404
- project setup, 389
- running, 401
- UI design, 393-396
- universal app, creating, 404-407
- unwinding back to initial scene, 398
- view controllers
  - adding, 390
  - associations, 391
  - exit preparations, 393
  - labels, 391

Model-View-Controller. *See* MVC
momentum, 687
monitoring CPU/memory usage, 895
motion
- acceleration data, handling, 683-684
- accelerometers, 663-665
  - 1g of force on devices at rest, 665
  - acceleration data, 665
  - attitude, 665
  - measurable axes, 664
- attitude data, handling, 682-683
- Core Motion, 130, 668-670
  - CMMotionManager, configuring, 669-670
  - motion manager, initializing, 680-681
  - motion updates, receiving, 669
- gyroscopes, 663, 666-667
  - attitude data, handling, 682-683
  - radians per second, 666
- high-pass filters, 687
- interface-orientation changes, disabling, 685-686
- low-pass filters, 687
- momentum, 687
- Newton's laws of motion website, 687
- operations queues, 670
- orientation. *See* orientations
- radians to degrees conversion, 677
- rotations, handling, 684-685
- updates, monitoring, 681-682

motionEnded:withEvent method, 653
Movable Type Scripts latitude/longitude functions documentation website, 808

movies
playback, implementing, 717-718
playback cleanup, 718
video player
external device playback/AirPlay support, 700
loading/displaying movies, 699-700
playback completion, handling, 701-702
playback controls, 701
screen placement, 700
MPMediaItem class, 694
MPMediaItemCollection class, 694
MPMusicPlayerController, 694
MPPediaPickerController class, 694
multi value preferences, 545
multibutton alerts, creating, 346-348
multiple storyboards, 857
multiscene storyboards, 362
actions, 399
editor scenes, creating, 395-396
initial scenes, creating, 393-395
ModalEditor app. *See* ModalEditor app
moving backwards, 378-380
connecting to exits, 379-380
exit preparations, 378-379
naming scenes, 366
outlets, 399
passing data between scenes, 386-388
scenes, adding, 364-365
segues
creating, 368-371
manually creating popovers, 383-386
manually triggering, 377-378
modal, 371-373, 396-398
popover, 373-377, 402-404
programming from scratch, 380-383
transitions, 372-373
terminology, 363
UI design, 393-396
unwinding back to initial scenes, 398
view controllers, 367-368, 458
multitasking. *See* backgrounding
multitouch events (iOS Simulator), 64
multitouch gesture recognition, 627
3D Touch peek and pop
adding, 631-632
editor screen, 655-657
implementing, 654-655
overview, 630-631
table views, 657-659
testing, 633
built-in gestures, 641
image views
replacing, 644-645
resetting to default, 635
outlets, 642
recognizers, adding
adding, 628-629
classes, 628
pinch, 640-641
rotation, 641
shake, 651-653
swipe, 639-640
tap, 638-639
responding to gestures
actions, 642-644
pinching, 646-649
rotations, 649-650
shake, 653
swipe, 646
tap, 645-646
SimpleGestureRecognizers tutorial, 659
swipes, 627
views, creating, 636-637
music library
accessing
empty selections, handling, 733-734
media picker, displaying, 732-733
media picker preparations, 730
playlist, creating, 733
preparations, 730-731
filter, 695
playing music, 734-735
mutators, 81
MVC (Model-View-Controller), 23, 193
application parts, defining, 194
data models, 198
further exploration, 218
HelloNoun app, 203-205
Single View Application template, 200
application logic, implementing, 216-217
class files, 202-203
interface objects, adding, 209-212
project, setting up, 201-202
simulated interface attributes, 208
variable properties, 205-206
view/view controller connection, 212-215
view controllers
IBAction, 198
IBOutlet directive, 197
implementing, 196
views, implementing, 195-196
myInstanceMethod method, 95
myName string, 87
myOptionalString variable, 95

N

named parameters, 98
names
objects with connected outlets, 215
Playgrounds, 112
scenes, 366

Simulator, 69
storyboard scenes, 160
navigation, 459
action, 479
adding, 461-462
additional features, 491
Apple tutorial, 491
application UI design, 477
bars, 460
bar button items, 460
customizing, 462-463
defined, 460
item attributes, 462-463
navigation items, 460
code, 43
configuring, 472-473
counters, 480-481
data sharing, 464
delegate, 707
items, 460
attributes, 462-463
UIs, adding, 477
LetsNavigate project setup, 471-472
outlet, 478
pushing/popping, 459, 479-480
scenes, adding, 463, 473-474
show segues, creating, 475
view controller
association, 473-474
classes, adding, 472
Xcode
documentation, 144-146
projects, 34-35
Navigator (Playground), displaying, 117
navigators (Xcode), 33
code searches, 45-46
debugger, 893-894
breakpoints, 893
debug, 893-894
methods/properties, finding, 43
project navigation, 34-35
Navigators, Show Navigator command (Playground View menu), 117
negation (!) expression, 103
negative g-force, 664
New, Playground command (Xcode File menu), 111
New, Tab command (Xcode File menu), 49
New Project command (Xcode File menu), 30
newBFF method, 766
Newsstand framework, 130
Newton's laws of motion website, 687
NeXTSTEP platform, 125
nonbridged data types, 136
north (maps), 787
not equal (!=) expression, 103
Notification Center framework, 127
notifications
alerts
actions, 333-334
buttons, 332-333
example, 331
fields, adding, 349-350
implementing, 331-333, 345-346
multibutton, creating, 346-348
Playground, 336-337
sounds/vibrations, 339-340
text fields, adding, 333
GettingAttention app
action sheets, implementing, 350-352
actions, 344-345
alert sounds with vibrations, 354-355
alerts, implementing, 345-346
fields in alerts, adding, 349-350
multibutton alerts, creating, 346-348
outlet, 343
project setup, 340
sound resources, adding, 340
UI design, 342-343
local
authorization, requesting, 824-825
creating, 826-827
properties, 825
scheduling, 814-815, 827
orientation changes, 667-668, 673
remote, 815
sounds/vibrations, 337-338
video player playback completion, 701-702
nowPlayingItem property, 697
NSDate class, 137
NSLog function, 878-879
calling, 879
output, displaying, 879-880
print function, compared, 878
NSOperationQueue, 670
NSURL class, 137, 295
NSURLRequest class, 295
NSUserDefaults class, 543
numberOfComponentsInPicker View method, 420, 445
numberOfSectionsInTableView method, 503, 515

O

Object Library, 162-163
displaying, 162
objects
adding to views, 163-164
deleting from views, 164
object-oriented programming. *See* OOP
objects
application, 134
built-in actions, 185
declaring/initializing, 90
defined, 76
document outline, 161
dragging, 215
identities, configuring, 187-188
installed size classes, 862-863

names, 215
personInformation, 101
pinning, 171
removing from memory, 110-111
responsive interfaces
defining, 617
initializing, 617-618
variable properties, retrieving, 81
onscreen controls, 135
OOP (object-oriented programming)
classes, 75
constants, 76
extensions, 76
inheritance, 75
instances, 76
instantiation, 76
methods
instance, 76
type, 76
objects, 76
overview, 74-75
parameters, 76
protocols, 76
self, 77
singletons, 76
subclasses, 75
superclasses, 75
variable properties, 76
variables, 76
OpenGL, 8
OpenGL ES framework, 128
openWeb method, 775
operations queues, 670
opinionated software, 539
optional values
accessing, 94-95
binding, 95
chaining, 99
declaring, 94
downcasting, 88-96
overview, 93-94
OR (||) expression, 103
Orientation app, 671
application logic, 672-675
orientation changes, responding, 673-675
orientation updates, registering, 673
interface design, 671
label text outlet, 672
project setup, 671
orientationChanged method, 674
orientations
devices
changes, responding, 673-675
controls, repositioning, 867
label text changes, 672
notifications, requesting, 667-668
portrait *versus* landscape, 620
sensing, 670
size class representations, 858
updates, registering, 673
interface-related
Auto Layout, 588
disabling changes, 685-686
enabling changes, 586-588, 616
handling changes, 621
programming in code, 589
rotation, activating, 587
size classes, 590
views, swapping, 590
outlets
animation loops, 274-275
BackgroundDownload app, 841
BestFriend app, 764
ColorTilt app, 679
connections, creating, 181-182
Cupertino app, 792
Cupertino Compass app, 801
CustomPicker app, 444
date picker initial scene, 430
defined, 181
delegate/data source, connecting, 513-514
direct file system access, 574
gestures, 642
ImageHop app, 264
label text orientation, 672
MediaPlayground app, 715
ModalEditor app, 399
navigation controllers, 478
scrolling views, 322
segmented controls, 306
sliders, 275, 554
SlowCount app, 836
steppers, 275
tab bar controllers, 487
text fields/text views, connecting, 244-246
toggle switches, 554
user output label, 343
view controller variable properties, 205-206
view to view controller connections, 213-215
web views, 306, 523
Xcode object naming, 215
outputLabel outlet, 430

P

page sheet modal segue, 372
pagination, 318
panning gesture class, 628
parameters
defined, 76
named, 98
parent classes, 75
parenthesis ( ), autocompleting, 45
partial vurl transitions, 372
passing data between scenes, 386-388
PassKit framework, 130
passthrough views (popovers), 376-377
pause method, 730
peek and pop gestures
adding, 631-632
editor screen, 655-657
implementing, 654-655
overview, 630-631
table views, 657-659
testing, 633
people pickers
cancel action, 745
contact selections, 766-768

delegate protocol, 765
displaying, 745, 766
drilling down, 746
user selections, handling, 745-746
personInformation object, 101
Photos framework, 128
Photos UI framework, 128
pickers, 417
custom, 437, 444
actions, 444
chooser scene, 441-442
component constants, 440
component/row sizes, 448
data, loading, 444-445
data source protocol, implementing, 445-446
image resources, adding, 439
initial scene, 441
labels, 440
outlet, 444
picker view data source/delegate, 442-443
project setup, 438-440
scenes, adding, 439
segues, 443
toolbar, adding, 440
UI design, 440-443
view controller association, 439
view controller class, adding, 439
data source, 442-443
date. *See* date pickers
delegates
custom picker, 442-443
protocol, implementing, 446-448
height/width components/rows, 422
images. *See* images, pickers
media
configuring, 694-695
displaying, 695, 732-733
filters, 695
iPad popover enforcement, 707
music players, preparing, 730
selections, handling, 696
number of components, returning, 445
number of elements per component, returning, 446
people picker
cancel action, 745
displaying, 745, 766
drilling down, 746
person selection, handling, 745
user selections, handling, 745-746
segues
modal, dismissing, 450-451
popovers, 451
selections
default, 450
displaying, 448-449
reacting, 449-450
spinning, 452
UIDatePicker/UIPicker classes, 140
views, 419
customizing, 423-424
data source protocol, 420-421
delegate protocol, 421-422
pickerView:didSelectRow: inComponent method, 421, 448
pickerView:numberOfRowsIn Component method, 420, 446
pickerView:rowHeightFor Component method, 422, 448
pickerView:titleForRow: forComponent method, 421
pickerView:viewForRow: forComponent:reusingView method, 446
pickerView:viewForRow:viewFor Component:ReusingView method, 422
pickerView:widthForComponent method, 422, 448
pinch gesture
class, 628
recognizer
adding, 640-641
responding, 646-649
pinning, 171, 603
placeholder text, 234
plain tables, 498
plain text, 235
play method, 730
playAudio method, 723
playback (media player), 697
playbackState property, 697
player property, 716
Playground, 111
alerts, 336-337
animation loops, 281-282
Cocoa Touch classes, 137
code, entering, 113
creating, 111-112
data storage, 550-551
date pickers, 435-436
displaying, 112
FieldbuttonFun app, testing, 251
geocoding, 759-760
names, 112
Navigator, displaying, 117
output
displaying, 114
generating, 114
history, 114-115
timelines, 116
resources, 117
run modes, switching, 117
transformations, 650-652
troubleshooting, 116
variable contents, displaying, 113-114
web views, testing, 296-297
playing
audio. *See* audio
movies/video. *See* movies; video
music, 734-735
empty selections, handling, 733-734
media picker, displaying, 732-733

media picker preparations, 730
playlist, creating, 733
sounds, 338
system sounds, 353-354
playMovieFinished method, 718
playMusic method, 734
.plist files
keys, 782
Quick Actions
adding, 818
configuring, 844-845
podcasts filter, 695
popovers, 373-377
action sheets, 352
anchors, 376
appearance, troubleshooting, 405
configuring, 402-404
creating, 374-375
date pickers, 437
dismissing, 385-386
iPhones, displaying, 405
outside touch dismissal, 452
presentation controller
accessing, 383
delegates, 406
presentation directions/ passthrough views, 376-377, 384-385
Presentation segues, 370
sizing, 375-376, 385, 403, 451
source location, 384
UIPopoverPresentation Controller class, 141
universal app, creating, 404-407
populating
data structures, 525-526
tables, 504-505, 514
text fields, 400
Portrait screen orientation constant, 586
Portrait Upside-Down screen orientation constant, 586
portrait *versus* landscape orientations, 620
positioning
bar button items, 416
objects on guides, 165-166
posting to social networking sites, 750-752
preferences
Apple Settings application, 540-541
benefits, 540
direct file system access. *See* direct file system access
further exploration, 579
implicit. *See* implicit preferences
loading upon startup, 570
opinionated software, 539
Playground, 550-551
settings bundles. *See* settings bundles
types, 544
user defaults, 543-544
Preferences, Text Editing, Editing command (Xcode), 45
Preferences menu commands (Xcode), 45
preferredContentSize property, 437
preferredStatusBarStyle method, 284
prepareForSegue method, 387, 406
Present Modally segues, 370
presentation directions (popovers), 376-377
presentViewController method, 382
previewing UIs, 176-178
previewingContext:commitView Controller method, 631
previewingContext:viewController ForLocation method, 631
print functions *versus* NSLog, 878
procedural programming, 74
processors, 8
projects
animation resources, adding, 263
configuring as universal, 854
creating, 30-32
product names/ organization names/ identifiers, 31-32
templates, 30-31
files, deleting, 37-38
filtering, 35
groups, 35
navigating, 34-35
new code files, adding, 35-36
properties, managing, 55
resources
adding, 36-37
deleting, 37-38
saving, 32
SimpleGestureRecognizers tutorial, 659
UICatalog Apple sample, 254
.xcodeproj file extension, 32
properties, 76
allowsExternalPlayback, 700
animationDuration, 269, 278
animationImages, 278
audioPlayer, 722
audioRecorder, 720
backgroundColor, 555
count, 89
declaring, 80-81
desiredAccuracy, 786
distanceFilter, 786
finding in code, 43
headingFilter, 787
isAnimating, 279
local notifications, 825
nowPlayingItem, 697
playbackState, 697
preferredContentSize, 437
pushCount, 479-480
retrieving, 81
setdateFormat, 431
splitViewController, 509
UIDeviceOrientation, 667-668
variables, compared, 197-198
video playback, 698
Xcode projects, managing, 55
protocols, 80
data source
custom picker views, implementing, 445-446
tables, 503-505

defined, 76
delegate
custom picker views, implementing, 446-448
tables, 505-506
image picker, 724
location manager delegate, 782-784
Mail compose delegate, conforming, 771
picker views
data source, 420-421
delegate, 421-422
table view data source
numberofSectionsInTable View method, 515
tableView:cellForRowAt IndexPath method, 516
tableView:numberOfRowsIn Section method, 515
tableView:titleForHeaderIn Section method, 515
table view delegate, 517-518
UIPopoverPresentation ControllerDelegate, 385
prototype cells, configuring, 501, 512
provisioning profiles
multiple devices, 7
overview, 15-16
viewing, 21
pushCount property, 479-480
pushing scenes, 479-480

Q

Quartz Core framework, 129
question marks (?), 94, 748
Quick Actions, 817-819
defining, 818-819
events, handling, 846-847
icons, 819, 844
.plist file configuration, 844-845
responding, 819
Quick Help (Xcode), 147
Inspector, 148
results, 148-149
Quick Help command (Xcode Help menu), 147
Quick Inspector, 185-186
Quick Look framework, 131
quitting applications, 218

R

radians per second, 666
radians to degrees conversion, 677
range attributes
sliders, 268-269
steppers, 270
reading
data, 549-550
preferences, 557
user defaults, 543-544
Reboot command (Xcode Hardware menu), 65
recent locations, storing, 803-804
recordAudio method, 720
recording audio, 703-705
controlling, 721-722
implementing, 720-721
microphone prompt, 722
recovering Simulator crashes, 67
regions (maps), 754
registerForPreviewingWithDelegat e:sourceView method, 631
registerUserNotificationSettings method, 815
remote notifications, 815
removeConstraints method, 634
repeatInterval property, 825
repeat-while loops, 106-107
resetting Simulator, 63
resolution
iPads, 7
iPhones, 6
scaling, 7
resources
animation, adding, 263
icon, 316
images
custom pickers, adding, 439
direction, 799
Gestures app, 635
settings bundles, 560
split view controllers, adding, 520
tables, adding, 510
Playground, 117
sound, adding, 340
Xcode projects
adding, 36-37
deleting, 37-38
responding
acceleration data, 683-684
attitude data, 682-683
device rotations, 684-685
gestures
pinching, 646-649
rotations, 649-650
shake, 653
swipe, 646
tap, 645-646
screen orientation changes, 673-675
table touches, 505-506
UIResponder class, 135
responsive interfaces
AllInCode app, 616-617
Auto Layout, 588, 590
buttons
centering, 604-606
touches, handling, 621
variable/size matching, 612-614
controls, expanding, 607-612
constraints, configuring, 609-612
interface, creating, 607-608
drawing upon launching, 620
interface orientation changes, enabling, 586-588
objects
defining, 617
initializing, 617-618
orientation changes, enabling, 616

portrait *versus* landscape orientations, 620
programming in code, 589
rotations
activating, 587
handling, 621
size classes, 590
update method, implementing, 619-620
views, swapping, 590
results, storing, 576-577
Retina assets, 40-42
Return Key attribute, 236
return values (methods), 100
ReturnMe app
application logic, 569-570
project setup, 559-560
image resources, 560
key constants, adding, 559
running, 570
settings bundle, creating, 562-568
About, 566-568
Contact Information group, 565
Sympathy Image, 563-565
UI, 560-561
reverse geocoding, 758-759
rich media. *See* media
rotations
devices
handling, 684-685
simulation, 64
gestures
adding, 641
class, 628
responding, 649-650
interfaces, 685-686
rows (tables)
counts, 503-504, 515
selections, handling, 517-518
run modes (Playground), 117
running
animation loops, 284
applications
build process, 51
development provisioning profiles, 15-16
device configuration, 16-18
iOS Simulator, 51
Welcome app, 19-21
what happens on devices, 17-18

S

Safari web browser
access, 752-753
URLs, opening, 775
Sample Code (Xcode documentation), 144
sandbox, 546-547
saving Xcode projects, 32
scaling
resolution, 7
web views, 305
scenes
custom pickers, adding, 439
date pickers
creating, 428-429
initial/date chooser, configuring, 426
defined, 363
detail, 523
dock, 162
editor, creating, 395-396
hierarchy, 158
initial
configuring, 205
creating, 393-395
labels, setting, 401
unwinding back to, 398
master, updating, 522
multiscene storyboards, 362
names, 160, 366
navigation. *See* navigation, 459
pushing, 479-480
segues. *See* segues
split view controller, 465, 509
tab bars. *See* tab bars
transitions, 372-373
views, compared, 196
scheduleLocalNotification method, 815, 827
scheduling local notifications, 814-815, 827
screen-edge panning gesture class, 628
screens
defaults, 162
graphics, 8
orientations
Auto Layout, 588
enabling changes, 586-588, 616
handling changes, 621
label text changes, 672
notifications, requesting, 667-668
portrait *versus* landscape, 620
programming in code, 589
responding to changes, 673-675
repositioning controls, 867
size classes, 590, 858
updates, registering, 673
views, swapping, 590
resolution
iPads, 7
iPhones, 6
rotations
activating, 587
handling, 684-685
scaling, 7
sizes, 8, 232
video player placement, 700
Scroller app
icon resources, adding, 316
outlet, 322
running, 323-324
scrolling, enabling, 322-323
setting up, 316
UI
scroll views, adding, 317-318
stack views, adding, 318-321
scrolling, 297
Apple tutorial, 324
enabling, 317-318, 322-323
freeform sizing, 321
icon resources, adding, 316
outlet, 322

pagination, 318
stack views, adding, 318-321
text view options, 241
width/height values, 323
SDK Guides, 144
searching
code, 45-46
Xcode documentation, 143-144
sections
master view controllers
labels, 527
number of, retrieving, 527
row counts, returning, 527
tables
configuring, 503-504
constants, adding, 511
headings, 515
number, returning, 515
row counts, returning, 515
Secure attribute, 236
security
ATS, 311
framework, 132
local notifications, 824-825
user locations, 768-769, 782
segmented controls, 293
adding, 300
Apple tutorial, 324
connecting to actions, 307-309
iOS 6 *versus* iOS 7 appearance, 301
outlets, 306
segments, configuring, 300-301
sizing, 302
UISegmentedControl class, 139
segments, configuring, 300-301
segues, 368
adaptive, disabling, 406
creating, 368-371
CustomPicker app, 443
date pickers, 429, 436-437
defined, 363
manually triggering, 377-378
dismissing, 378
starting, 377-378
modal, 371-373
creating, 396-398
date pickers, 436
dismissing, 450-451
ModalEditor app. *See* ModalEditor app
navigation events, handling, 530-531
peek and pop gestures, adding, 654-655
picker views
modal, dismissing, 450-451
popovers, 451
popovers. *See* popovers
programming from scratch, 380-383
displaying view controllers, 382
instantiating view controllers, 381
segue styles, 382
storyboard identifiers, 381
show
creating, 476
navigation scenes, 463
styles, configuring, 382
transitions, 372-373
types, 370
unwind, 379-380, 397-398
selection handles, 166-167
selections (picker views)
default, 450
displaying, 448-450
reacting, 449-450
self, 77
sendEmail method, 772
sending tweets, 773-774
sendTweet method, 774
services integration. *See* integration
setBackgroundHueValue method, 556
implementing, 555
switches/sliders, connecting, 554
setdateFormat property, 431
setDisplayName method, 101
setMinimumBackground FetchInterval method, 842
setObject:forKey function, 543
setOutput method, 217
setQueueWithItemCollection method, 697, 730
sets (Swift), 88
setSpeed method, 280
setters, 81
Settings application, 540-541
settings bundles, 544-546
adding, 544
creating, 562-568
child preference pane, 566-568
image selector preferences, 563-565
custom, 545
image resources, 560
key constants, adding, 559
loading preferences upon startup, 570
preference types, 544
text fields, 565
setToRecipients method, 749
setValuesFromPreferences method, 569
SFSafariViewController, 752
shake gesture, 628
adding, 651-653
responding, 653
Shake Gesture command (Xcode Hardware menu), 65
sharing data
between split view controller scenes, 509
navigation scenes, 464
tab bar scenes, 470
Show Detail segues, 370
Show Document Outline command (Xcode Editor menu), 232
show segues, 370
creating, 476
navigation scenes, 463
showAddress method, 769
showResults method, 577
SimpleGestureRecognizers tutorial project, 659
Simulate Hardware Keyboard command (Xcode Hardware menu), 66

Simulate Memory Warning command (Xcode Hardware menu), 66
simulated screen size, setting, 232
Simulator, 61
- Accessibility Inspector, enabling, 175-176
- applications
  - launching, 62-63
  - running, 51
- conditions, testing, 65-66
- crash recovery, 67
- devices
  - adding, 67-69
  - rotation, 64
- limitations, 62
- multitouch events, 64
- names, 69
- resetting, 63

Single View Application template, 200
- application logic, 216-217
- interface design
  - objects, adding, 209-212
  - simulated attributes, 208
- project, setting up, 201-202
  - class files, 202-203
  - storyboard files, 203-205
- variable properties, 205-206
- view/view controller connection, 212-215
  - actions, 215
  - outlets, 213-215

singletons, 76, 543
size classes. *See* classes, size
Size Inspector, 168
- cells, 501
- constraints
  - content compression resistance, 598
  - content hugging, 597
  - editing, 595-597
  - viewing/editing, 594-597
- segmented controls, 302

sizes
- buttons, 612-614
- cells, 501
- popovers, 375-376, 385, 403, 451
- screens, 8
- scrolling views freeform, 321
- segmented controls, 302
- view picker components/rows, 448

SLComposeViewController, 751
slicing
- button templates, creating, 227-232
  - images, adding, 227-228
  - slices, creating, 228-231
- images, 40

sliders, 260
- animation loop speed
  - adding, 267-268
  - range attributes, 268-269
- implicit preferences, 554
- labels, 267
- outlets, 275
- preferences, 544
- range attributes, 268-269
- trigger actions, 276
- UISlider class, 139

SlowCount app
- application logic, 836
  - counters, updating, 837
  - timer, initializing, 836
- background task processing, enabling, 837-839
- outlet, 836
- project setup, 834-835
- running, 839
- UI design, 835

Snapshots, 49
Social framework, 131, 751, 777
social networking, 750-752
- Social framework, 131, 751, 777
- tweets, sending, 773-774

sound file references, initializing, 829
soundName property, 825
sounds. *See* audio
soundSetting dictionary, 704
source control (Xcode) tutorial, 19
speed (animations)
- incrementing, 282-283
- loops
  - output labels, 271
  - slider, 267-269
  - steppers, 269-270
- setting, 280-281

Spell Checking attribute, 236
spinning pickers, 452
split view controllers, 506-507
- application data structures
  - creating, 524
  - populating, 525-526
- detail scenes, updating, 523
- detail view controller, configuring, 531-533
- hierarchy, 520-521
- image resources, adding, 520
- implementing, 508-509
- master scenes, updating, 522
- master view controller, 527
  - editing, disabling, 529-530
  - number of section rows, retrieving, 527
  - number of sections, retrieving, 527
  - section labels, 527
  - segue navigation events, handling, 530-531
  - table cells, creating, 528-529
  - table view data source methods, 527-528
- Master-Detail Application template, 509
- sharing information between master/detail scenes, 509
- web view outlets, connecting, 523

split-screen multitasking, 814
splitViewController property, 509
SpriteKit framework, 129
ss: date format, 432
stack views, 297
- adding, 318-321
- icons
  - horizontal, 319
  - images, adding, 318
- vertical, 320

starting
- animations, 271, 278-280
- applications, 18-22
- applications in Simulator, 62-63
- background processing, 838
- segues, 377-378
- Xcode, 29

startUpdatingHeading method, 787
state
- buttons, changing, 243
- media player, 697
- preservation, 541-542
- switch default, 303

statements
- defer, 109-110
- do-catch, 107-108
- guard, 109
- if-then-else, 104-105
- Swift, 77
- throw, 108-109

status bar appearance, customizing, 283-284, 775, 796
Step Into icon, 888
Step Out icon, 888
Step Over icon, 888
steppers, 260-261
- animation loops, adding, 269-270
- outlets, 275
- trigger action, 276
- UIStepper class, 140

stopDeviceMotionUpdates method, 670
stopping animations, 271, 278-280
stopUpdatingLocation method, 782
StoreKit framework, 131
storeSurvey method, 576
storing data
- direct file system access, 546-550
  - file paths, 548-549
  - reading/writing data, 549-550
  - sandbox restrictions, 546-547
  - storage locations, 547-548
- Playground, 550-551
- recent locations, 803-804
- settings bundles, 544-546
  - adding, 544
  - custom, 545
  - preference types, 544
- user defaults, 543-544

Storyboard Entry Point icon (storyboards), 159
storyboards, 157
- Apple tutorial, 407
- defined, 363
- detail scenes, updating, 523
- displaying, 157
- document outline objects, 161
- HelloNoun project file, 203-205
- icons, 158-159
- identifiers, 381
- initial scenes, configuring, 205
- master scenes, updating, 522
- multiple, 857
- multiscene. *See* multiscene storyboards
- navigation. *See* navigation
- old autosizing features, turning on, 172
- scenes. *See* scenes
- segues. *See* segues
- size classes, creating, 867-873
  - Any, Any, 873
  - Any(w), Compact(h), 870-873
  - Any(w), Regular(h), 868-870
  - previewing, 873
  - repositioning controls based on orientation, 867
- split view controllers, adding, 508-509
- tab bars. *See* tab bars
- view controllers
  - EditorViewController class association, 390-391
  - exit preparations, 393
  - instantiating, 381
  - labels, 391
- views, 160, 196

storyboardWithName method, 381
stretching UI objects, 166-167
stringFromDate method, 431
strings, 85
- characters, 87
- classes, 136
- concatenation, 87
- date formats, 432
- initializing, 87
- interpolation, 87
- myName, 87

structs, 84
structure. *See* MVC
stub methods, 187
subclasses, 75
superclasses, 75
supportedInterfaceOrientations method, 586, 616
Survey app
- actions, 574-575
- application logic, 575-578
  - displaying survey results, 577-578
  - keyboard, hiding, 575
  - storing survey results, 576-577
- field/text view outlets, 574
- project setup, 571-572
- UI design, 572

survey fields, storing, 576-577
suspending applications, 133, 814
Swift, 23
- access levels, 84
- arrays
  - contents, counting, 89
  - declaring/initializing, 88
  - index values, 88
  - iterating over, 106
  - new items, adding to end, 88
- Boolean values, 87
- bridged data type classes, 136
- case sensitivity, 77
- class files
  - ending, 83
  - structure, 78-79
- classes, declaring, 80

comments, 83
constants
declaring/initializing, 92
variables, compared, 93
data types
converting, 92
downcasting, 92
listing of, 85
declarations
class, 80
constant, 81
IBOutlet, 81
import, 79-80
dictionaries
contents, counting, 89
declaring, 89
iterating over, 106
key/value pairs, 89
values, 89
dot notation, 81
enumerations, 89-90
creating, 90
declaring, 90
values, 90
error handling, 107
defer statements, 109-110
do-catch statements, 107-108
guard statements, 109
throw statements, 108-109
expressions, 103
floating-point numbers, declaring, 86
getters/setters, 81
help learning, 118
if-then-else statements, 104-105
integers, declaring, 86
key/value pairs, 106
loops, 105
for, 105-106
condition-based, 106-107
iterating over collections, 106
methods
chaining, 98-99
closures, 101
convenience, 91
convenience initialization, 278
declaring, 82-83
Markdown documentation, 101-102
multiple return values, 100
named parameters, 98
syntax, 96-97
objects, declaring, 90
optional values
accessing, 94-95
binding, 95
chaining, 99
declaring, 94
downcasting, 88-96
overview, 93-94
overview, 77
print function *versus* NSLog, 878
properties
declaring, 80-81
retrieving, 81
variables, compared, 197-198
protocols, 80
sets, 88
statements, 77
defer, 109-110
do-catch, 107-108
guard, 109
throw, 108-109
strings
characters, 87
concatenation, 87
initializing, 87
interpolation, 87
structs, 84
tuples, 100
variables, 86
constants, compared, 93
declaring, 85
swipe gesture, 627
adding, 639-640
class, 628
responding, 646
switches
actions, 307-309
adding, 302-303
Apple tutorial, 324
default state, 303
overview, 292
toggle, 544, 554, 679
UISwitch class, 139
symbol navigator, 43
system buttons, 243
System Configuration framework, 131
System framework, 132
System Sound Services, 337-338
accessing, 338-339
alert sounds with vibrations, playing, 354-355
AudioToolbox framework, importing, 353
system sounds, playing, 353-354
systemMusicPlayer method, 730

T

tab bars, 464
actions, 487
adding, 466-467, 483
additional features, 491
Apple tutorial, 491
application UI design, 485-486
badges, 466
count button presses, tracking, 488
counter updates, triggering, 490
customizing, 467-468
data sharing, 470
defined, 465
display updates with counter values, 488-489
example, 465
images, 468
items, 465
attributes, 466
badges, updating, 489
labels, 485-486
LetsTab project setup, 482-484
outlets, 487
scenes
adding, 469, 483
connections, 484-485
Tabbed Application template, 466
tabbed editing, 49-50

tables, 497
3D Touch peek and pop gestures, adding, 657-659
adding, 499
appearance, 498
Apple tutorials, 534
attributes, 500-501, 512
cells, 498. *See also* cells
data source
methods, 527-528
outlets, 513-514
protocol, 503-505, 515-516
delegate
outlets, 513-514
protocol, 505-506, 517-518
functionality, increasing, 534
image resources, adding, 510
layout, configuring, 503-504
populating, 504-505, 514
row selections, handling, 517-518
sections
constants, adding, 511
headings, 515
number, returning, 515
row counts, returning, 515
user touches, responding, 505-506
tableView:canEditRowAtIndexPath method, 530
tableView:cellForRowAtIndexPath method, 503, 516, 528
tableView:didSelectRowAtIndex Path method, 505
tableView:numberOfRowsIn Section method, 503, 515
tableView:titleForHeaderInSection method, 503, 515
tap gestures
adding, 628-629, 638-639
class, 628
responding, 645-646
taps attribute (tap gesture recognizer), 638
tapView view, 629
task completion backgrounding, 816
counting slowly app
counters, updating, 837
outlet, 836
project setup, 834-835
timer, initializing, 836
UI design, 835
enabling, 837-839
task-specific backgrounding
audio map directions, 828
audio files, adding, 828
AudioToolbox framework, adding, 828
background modes, 832
heading update audio feedback, adding, 830-831
sound file references, initializing, 829
system sound ID variables, 829
overview, 816
technology layers. *See* layers
templates, 30-31
buttons, creating, 227-232
images, adding, 227-228
slices, creating, 228-231
icons, 819
Master-Detail Application, 509
Single View Application. *See* Single View Application template
Tabbed Application, 466
text, 224
adding, 237
attributes, editing, 238-239
copying/pasting, 237
Core Text framework, 128
data detectors, enabling, 239
delegates, 254
direct file system access outlets, 574
fields, 224
alerts, adding, 333, 349-350
attributes, editing, 233-235
copy/paste functionality, 237
delegates, 254
input traits, 236-237
labels, adding, 233
outlets, connecting, 244-246
placeholders, 234
plain *versus* attributed, 235
populating, 400
preferences, 544
settings bundles, 565
UIs, adding, 232-233
UITextField class, 140
user input, 206
input traits, 236-237
labels
adding, 237
orientation changes, 672
outlets, connecting, 244-246
placeholder, 234
plain *versus* attributed, 235
scrolling options, 241
Text Editing preferences (Xcode), 45
TextKit framework, 127
throw statements, 108-109
Time mode (date pickers), 418
timeInternalSinceDate |method, 432
timelines (Playground), 116
timers, initializing, 836
timeZone property, 825
title preferences, 544
titleForSegmentAtIndex method, 293
tmp directory, 548
TODO comments, 47
Toggle Breakpoints icon, 888
Toggle In-Call Status Bar command (Xcode Hardware menu), 66
toggle switches, 544, 554
action, 679
motion updates, monitoring, 681-682
outlets, connecting, 679
toggleAnimation method, 279
toggleFlowerDetail method, 311

toolbars, 33, 413
adding, 414
bar button items, 415-416
adding, 415
appearance, configuring, 416
positioning, 416
custom pickers, adding, 440
date picker, creating, 427
further exploration, 452
Top Layout Guide icon (storyboards), 159
top layout guides, 594
Touch ID Enrolled/Simulate Finger Touch command (Xcode Hardware menu), 65
touches
3D Touch peek and pop
adding, 631-632
editor screen, 655-657
implementing, 654-655
overview, 630-631
table views, 657-659
testing, 633
3D Touch Quick Actions, 817-819
defining, 818-819
events, handling, 846-847
icons, 819, 844
.plist file configuration, 844-845
responding, 819
background, 249-250
buttons, handling, 621
lower-level handling, 659
multitouch gesture recognition. *See* multitouch gesture recognition
SimpleGestureRecognizers tutorial project, 659
tables, responding, 505-506
touches attribute, 638
Traits attribute, customizing, 175
transformations (Playground), 650-652
transitions. *See* segues
transparent blurs, 273
triggering
buttons, 206, 275
counter updates, 490
segues manually, 377-378
dismissing, 378
starting, 377-378
sliders, 276
steppers, 276
troubleshooting
Auto Layout constraint errors, 598-601
Core Location errors, 783
Playground, 116
popover appearance, 405
tuples, 100
TV Out command (Xcode Hardware menu), 66
tweets, sending, 773-774
type methods, 76

U

UIActionSheet class, implementing, 335
UIAlertController object, implementing, 331-333
UIApplication class, 134
UIBarButtonItem class, 460
UIButton class, 139
UICatalog Apple sample project, 254
UIColor class, 555
UIControl class, 135
UIDatePicker class, 140
UIDeviceOrientation property, 667-668
UIImagePickerController class, 705
UIImagePickerControllerDelegate protocol, 724
UIInterfaceOrientationMask.All constant, 587
UIInterfaceOrientationMask. AllButUpsideDown constant, 587
UIInterfaceOrientationMask. Landscape constant, 586
UIInterfaceOrientationMask. LandscapeLeft constant, 586
UIInterfaceOrientationMask. LandscapeRight constant, 586
UIInterfaceOrientationMask. Portrait constant, 586
UIInterfaceOrientationMask. PortraitUpsideDown constant, 586
UIKit framework, 126
UILabel class, 90, 138
UILongPressGestureRecognizer class, 628
UINavigationBar class, 460
UINavigationController class, 459
UINavigationControllerDelegate protocol, 724
UIPanGestureRecognizer class, 628
UIPicker class, 140
UIPinchGestureRecognizer class, 628
UIPopoverPresentationController class, 141, 383
UIPopoverPresentationController Delegate protocol, 385
UIResponder class, 135
UIRotationGestureRecognizer class, 628
UIs
appearance
accessibility attributes, 174-175
Attributes Inspector, 173-174
Auto Layouts, 169-172
constraints, 170-172
Content Compression, 171
Content Hugging, 171
Intrinsic Size setting, 172
turning off, 172
BackgroundColor app, 552-553
BackgroundDownload app, 840
BestFriend app, 762-764
classes, 138-141
UIButton, 139
UIDatePicker/UIPicker, 140
UILabel, 138
UIPopoverPresentation Controller, 141
UISegmentedControl, 139

UISlider, 139
UIStepper, 140
UISwitch, 139
UITextField/UITextView, 140
ColorTilt app, 677
connections, 178
actions, 183
deleting, 186
editing, 185-186
outlets, 181-182
project, opening, 178
controls, expanding, 607-608
creating
default screens, 162
Object Library, 162-163
Cupertino app, 790-792
Cupertino Compass, 800-801
CustomPicker app, 440-443
DateCalc app, 427-429
date chooser scene, 428-429
initial scene, 427-428
editing, disabling, 530
flower app, 299
button, 305
color choice segmented controls, 300-302
flower details switch, 302-303
web views, 303-305
FlowerColorTable app, 511-512
FlowerDetail app, 521-523
detail scene, updating, 523
master scene, updating, 522
web view outlet, connecting, 523
Gesture app, 636-637
GettingAttention app, 342-343
ImageHop app. *See* ImageHop app
LetsNavigate app, 477
LetsTab app, 485-486
MediaPlayground app, 711-713
modal elements, 331
multiscene storyboards, 393-396
objects. *See* objects
Orientation app, 671
orientation changes, enabling, 586-588
previewing, 176-178
responsive. *See* responsive interfaces
ReturnMe app, 560-561
rotations, 685-686
scrolling views, adding, 317-318
simulated attributes, 208
SlowCount app, 835
stack views, adding, 318-321
Survey app, 572
text fields, adding, 232-233
UIScreenEdgePanGesture Recognizer class, 628
UIScrollView class, 324
UISegmentedControl class, 139
UISlider class, 139
UIStackView class, 324
UIStepper class, 140
UISwipeGestureRecognizer class, 628
UISwitch class, 139
UITabBar class, 465
UITabBarController class, 464
UiTapGestureRecognizer class, 628
UITextField class, 140
UITextView class, 140
UITouch class, 659
UIView class, 135
UIViewController class, 135
UIViewControllerTransition Coordinator class, 622
UIVisualEffectView class, 285
UIWebView class, 324
UIWindow class, 134
Unicode date formats website, 432
universal applications, 586, 853
creating, 854
icons, 855
launch screen, 856
documentation, 873
popovers, creating, 404-407
size classes. *See* classes, size
tools/techniques, 856-857
unwind segues, 379-380, 398
unwinding, 363
unwrapping optional values, 94-95
updateBadge method, 489
updateCounts method, 488
updateEditor method, 401
updateLayoutWithScreen Width:screenHeight method, 619-620
updates
counters, 490, 837
date output, 434-435
detail scenes, 523
display with counter values, 488-489
headings, 788
audio feedback, adding, 830-831
handling, 805-807
implicit preferences, 557
locations, receiving, 782-784
master scenes, 522
motion
monitoring, 681-682
receiving, 669
orientation, registering, 673
.plist files, 797
responsive interfaces, 619-620
tab bar item badges, 489
user defaults, 543-544
userAge variable, 86
userAgeInDays variable, 86
userMessages array, 88
userMessages dictionary, 89
Utilities commands (Xcode View menu)
Show Attributes Inspector command, 173
Show Connections Inspector command, 183
Show Identity Inspector command, 174

Show Object Library command, 162
Show Size Inspector command, 168
utility area (Xcode), 33

V

valueForProperty method, 697
values
dictionaries
accessing, 89
assigning/modifying, 89
enumerations, 90
method multiple return, 100
optional
accessing, 94-95
binding, 95
chaining, 99
declaring, 94
downcasting, 88-96
overview, 93-94
UIDeviceOrientation property, 667-668
var keyword, 86
variables
constants, compared, 93
contents, displaying in Playground, 113-114
declaring, 85
defined, 76
Mad Libs-style story creator, planning, 227
myOptionalString, 95
properties, compared, 197-198
userAge/userAgeInDays, 86
Xcode debugger
examining, 887
listing of, displaying, 892
values, editing, 887
vertical constraints (Auto Layout), 593
vertical size class, 858
vertical stack views, 320
vibrancy, 272
vibrations
alerts, 339-340, 354-355
System Sound Services, 337-339
video
formats supported, 699
frameworks
AV Foundation, 127
AVKit, 127
Core Image, 127
playback classes, 698
video player
cleanup, handling, 718
external device playback/AirPlay support, 700
implementing, 716
loading/displaying movies, 699-700
movie playback, 717-718
playback completion, handling, 701-702
playback controls, 701
screen placement, 700
View, Navigators, Show Navigator command (Playground), 117
View Controller icon (storyboards), 158
view controllers
adding, 390
animal chooser class, 439
associations, 483
EditorView Controller class, 391
navigation controllers, 473-474
tab bar controllers, 483
child view controllers, adding, 700
classes, 472
container views, adding, 712-713
custom picker scenes, associating, 439
DateChooserViewController class association, 426
defined, 363
delegate/data source outlets, connecting, 513-514
detail, configuring, 531-533
directives
IBAction, 198
IBOutlet, 197
displaying, 382
exits
connecting, 379-380
preparations, 378-379, 393
first responders, 651-653
further exploration, 407
implementing, 195-196
interfaces
objects, adding, 209-212
simulated interface attributes, 208
labels, 391
Mail compose, 749-750
master, 527
editing, disabling, 529-530
number of section rows, retrieving, 527
number of sections, retrieving, 527
section labels, 527
segue navigation events, handling, 530-531
table cells, creating, 528-529
table view data source methods, 527-528
multiscene development, 458
navigation. *See* navigation
passing data between scenes, 387
properties, 205-206
relationships, 469
Safari, 775
SFSafariViewController, 752
social networking, 751
split. *See* split view controllers
storyboards
identifiers, 381
instantiating, 381
subclass support, adding, 367-368
tab bars, 464
views, connecting, 212-215
actions, 215
outlets, 213-215
View icon (storyboards), 159

View, Utilities commands (Xcode)
Show Attributes Inspector, 173
Show Connections Inspector, 183
Show Identity Inspector, 174
Show Object Library, 162
Show Size Inspector, 168
view pickers
data source protocol, 420-421
delegate protocol, 421-422
implementing, 421-422
methods, 422-424
ViewController class, 43
ViewController.swift
connections code listing, 206
outlets/actions, 276
setOutput method, 217
viewDidLoad method, 44, 314, 883
viewing. *See* displaying
views
3D Touch ready, registering, 631
animated image, implementing, 277-278
background images, adding, 271-272
blur visual effect, 272-273
constraints, 394
Contacts picker, displaying, 766
container, 712-713
custom annotation, 770-771
defined, 363
email composition, 748
gesture recognizers, adding
pinch, 640-641
rotation, 641
swipe, 639-640
tap, 638-639
hierarchy, displaying, 896-897
image, 261
animation loops, adding, 264
default, resetting, 635
replacing, 644-645
transformations (Playground), 650-652
implementing, 195-196
mail compose
completion, handling, 772-773
displaying, 772
map
adding, 753
annotations, 755-757
delegate protocol, 756-757
regions, 754
modal, 363
pickers. *See* pickers
scenes, compared, 196
scrolling, 297
adding, 317-318
Apple tutorial, 324
freeform sizing, 321
icon resources, adding, 316
outlet, 322
pagination, 318
scrolling, enabling, 322-323
stack views, adding, 318-321
width/height values, 323
Single View Application template, 200
stack, 297
adding, 318-321
horizontal, 319
icon images, adding, 318
image views, adding, 318
vertical, 320
storyboards, 160
swapping, 590
tables. *See* tables
tapView, 629
text. *See* text
transparent blur effect, 273
tweet compose, 774
UI objects
adding, 164
aligning, 167-168
arranging, 167
deleting, 164
positioning, 165-166
stretching, 166-167
UIView class, 135
UIViewController class, 135
view controllers, connecting, 212-215
actions, 215
outlets, 213-215
visual effects, 180, 261
web. *See* web views
zooming in/out, 166
viewWillAppear method, 400
viewWillAppear:animated method, 480
viewWillTransitionToSize:withTransitionCoordinator method, 589, 621
visual effects
blur, 272-273, 314-315
frosted glass, 561
views, 180, 261
visual notifications
action sheets, 334-337
alerts
actions, 333-334
buttons, 332-333
example, 331
implementing, 331-333
Playground, 336-337
text fields, adding, 333
vmOSX, 10

W

warnings, 52, 116
watchpoints, configuring, 890-891
web views, 294-297
adding, 303-304
Apple tutorial, 324
attributes, 304-305
content, loading, 311-313
content types supported, 294
hiding/displaying, 309-311
initial display, 313-314
outlets, 306, 523
Playground test, 296-297
remote content, loading, 295-297
scaling, 305
segments, configuring, 300-301

websites
- Adobe Flash Professional, 10
- Apple Developer Program Enrollment page, 11
- Apple IDs, 11
- Euler angles, 683
- high-pass filters, 687
- icon templates, 819
- iOS Design Cheat Sheet by Ivo Mynttinen, 60
- low-pass filters, 687
- Markdown, 101
- momentum, 687
- Movable Type Scripts latitude/longitude functions documentation, 808
- Newton's laws of motion, 687
- *Sam's Teach Yourself iOS 9 Application Development in 24 Hours*, 3
- Swift access control, 84
- Unicode date formats, 432
- vmOSX, 10
- Xcode, 14, 19

Welcome app, running, 19-21

Welcome screen (Xcode), 30

while loops, 106-107

whiteColor method, 555

width
- picker components/rows, 422
- popovers, 375-376
- scrolling views, 323
- view picker components, 448

Window, Devices command (Xcode), 67

Windows development options, 10

Wolfram Mathworld Euler angles website, 683

writeData method, 550

writing
- data, 549-550
- user defaults, 543-544

X

Xcode
- Apple ID, adding, 16
- applications
  - building process, 50
  - device orientations, setting, 56
  - icons, 57-60
  - running in iOS Simulator, 51
- asset catalogs. *See* asset catalogs
- assistant editor mode, 47
- build schemes, choosing, 50-51
- code
  - autocompletion, 44-45
  - editing, 42
  - methods/properties, finding, 43
  - navigating, 43
  - problems, fixing, 52-55
  - searching, 45-46
- comments, adding, 46-47
- Core Data, 32
- curly brace/parenthesis, autocompleting, 45
- debugger, 881-882
  - activity to monitor, adding, 883
  - breakpoints, setting, 885-886
  - Continue to Here, 890
  - CPU/memory usage, monitoring, 895
  - editing variable values, 887
  - examining variables, 887
  - gutter, 883
  - navigators, 893-894
  - source code line numbers, 883
  - stepping through code, 888-890
  - variable lists, accessing, 892
  - view hierarchy, displaying, 896-897
  - watchpoints, 890-891
- devices, adding, 17
- documentation. *See* documentation
- gutter, 883
- help, 24
- IB. *See* IB
- installing, 13-15
- interface, 33-34
- iOS Simulator. *See* Simulator
- launch images/screens, 60
- launching, 29
- libraries, 163
- Playground. *See* Playground
- projects. *See* projects
- Quick Help, 147
  - Inspector, 148
  - results, 148-149
- Snapshots, 49
- source control tutorial, 19
- Stop button, 571
- tabbed editing, 49-50
- Text Editing preferences, 45
- view controllers
  - IBAction, 198
  - IBOutlet directive, 197
  - implementing, 195-196
- views, implementing, 195-196
- Welcome screen, 30

.xcodeproj file extension, 32

Y

YYYY: date format, 432

Z

zooming views, 165