Fritz Anderson

Xcode 6 Start to Finish

iOS and OS X Development

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

Xcode 6
Start to Finish

Developer's Library Series

Stephen G. Kochan Second Edition

Programming in Objective-C 2.0

A complete introduction to the Objective-C language for Mac OS X and iPhone development

Developer's Library

Shane Conder Lauren Darcey

Android Wireless Application Development

Developer's Library

Jason Ouellette

Development with the Force.com Platform

Building Business Applications in the Cloud

Developer's Library

Erica Sadun Second Edition

The iPhone™ Developer's Cookbook

Building Applications with the iPhone 3.0 SDK

Developer's Library

Addison-Wesley

Visit **developers-library.com** for a complete list of available products

The **Developer's Library Series** from Addison-Wesley provides practicing programmers with unique, high-quality references and tutorials on the latest programming languages and technologies they use in their daily work. All books in the Developer's Library are written by expert technology practitioners who are exceptionally skilled at organizing and presenting information in a way that's useful for other programmers.

Developer's Library books cover a wide range of topics, from open-source programming languages and databases, Linux programming, Microsoft, and Java, to Web development, social networking platforms, Mac/iPhone programming, and Android programming.

PEARSON

Addison-Wesley Cisco Press EXAM/CRAM IBM Press QUE PRENTICE HALL SAMS | Safari Books Online

Xcode 6 Start to Finish

---

iOS and OS X Development

Fritz Anderson

Addison-Wesley

New York • Boston • Indianapolis • San Francisco
Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

Xcode is a trademark of Apple, Inc., registered in the U.S. and other countries.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the United States, please contact international@pearsoned.com.

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data
Anderson, Fritz.
  Xcode 6 start to finish : iOS and OS X development / Fritz Anderson.
    pages cm
  Includes index.
  ISBN 978-0-13-405277-9 (pbk. : alk. paper)
  1. Mac OS. 2. iOS (Electronic resource) 3. Macintosh (Computer)—Programming. 4. iPhone (Smartphone)—Programming. 5. Application software—Development. I. Title.
   QA76.774.M33A534 2015
   005.4'46—dc23
2015004190

Copyright © 2015 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, 200 Old Tappan Road, Old Tappan, New Jersey 07675, or you may fax your request to (201) 236-3290.

ISBN-13: 978-0-13-405277-9
ISBN-10: 0-13-405277-3

Text printed in the United States on recycled paper at Edwards Brothers Malloy in Ann Arbor, Michigan.
First printing, June 2015

Editor-in-Chief
Mark L. Taub

Senior Acquisitions Editor
Trina MacDonald

Senior Development Editor
Chris Zahn

Managing Editor
John Fuller

Full-Service Production Manager
Julie B. Nahil

Copy Editor
Stephanie Geels

Indexer
Ted Laux

Proofreader
Kathleen Allain

Technical Reviewers
Duncan Champney
Chuck Ross
Dan Wood

Editorial Assistant
Olivia Basegio

Cover Designer
Chuti Prasertsith

Compositor
Lori Hughes

❖

For the Honorable Betty Shelton Cole,
a tough old broad

❖

This page intentionally left blank

Contents at a Glance

Contents ix

Acknowledgments xxiii

About the Author xxv

Introduction 1

I First Steps 7

1 Getting Xcode 9

2 Kicking the Tires 17

3 Simple Workflow and Passive Debugging 25

4 Active Debugging 35

5 Compilation 45

6 Adding a Library Target 69

7 Version Control 79

II The Life Cycle of an iOS Application 105

8 Starting an iOS Application 107

9 An iOS Application: Model 117

10 An iOS Application: Controller 141

11 Building a New View 157

12 Auto Layout in a New View 185

13 Adding Table Cells 207

14 Adding an Editor 227

15 Unit Testing 243

16 Measurement and Analysis 265

17 An iOS Extension **279**

18 Provisioning **297**

III Xcode for Mac OS X 319

19 Starting an OS X Application **321**

20 Bindings: Wiring an OS X Application **343**

21 Localization **373**

22 Bundles and Packages **401**

23 Property Lists **417**

IV Xcode Tasks 433

24 Documentation in Xcode **435**

25 The Xcode Build System **459**

26 Instruments **489**

27 Debugging **515**

28 Snippets **531**

V Appendixes 547

A Some Build Variables **549**

B Resources **565**

Index **579**

Contents

Acknowledgments xxiii

About the Author xxv

Introduction 1

How This Book Is Organized 1

About Versions 4

About the Code 4

Conventions 5

I First Steps 7

1 Getting Xcode 9

Before You Begin 9

Installing Xcode 10

- Command-Line Tools 11

Removing Xcode 11

Apple Developer Programs 12

Downloading Xcode 13

Additional Downloads 14

Summary 15

2 Kicking the Tires 17

Starting Xcode 17

Hello World 19

- A New Project 19
- Quieting Xcode Down 22
- Building and Running 22
- The Real Thing 24

Getting Rid of It 24

Summary 24

3 Simple Workflow and Passive Debugging 25

Creating the Project 25

Building 29

Running 30

Simple Debugging 32
Summary 34

4 Active Debugging 35
A Simple Test Case 35
Going Active 35
Setting a Breakpoint 36
The Variables Pane 37
Stepping Through 37
Fixing the Problem 39
Behaviors 40
The Fix 42
Summary 43

5 Compilation 45
Compiling 45
Dynamic Loading 52
Xcode and Clang 52
Local Analysis 53
Cross-Function Analysis 55
Indexing 56
Swift 57
Compiler Products 62
Intermediate Products 62
Precompilation 64
Summary 66

6 Adding a Library Target 69
Adding a Target 69
Targets 70
Target Membership 71
Adding Files to a Target 71
Headers in Targets 74
A Dependent Target 74
Adding a Library 75
Debugging a Dependent Target 76
Summary 77

7 Version Control 79
Taking Control 80
Creating a Git Repository by Hand 81
The State of Your Files 82
How Xcode Works with Git 83
Your First Commit 84
Working with Remote Repositories 84
Setting Up a “Remote”—Locally 87
Pushing to the Remote 88
Merges and Conflicts 89
User A 90
User B 93
Back to User A 96
The Version Editor 99
Comparison 99
Blame 101
Log 101
Branching 102
Summary 104

II The Life Cycle of an iOS Application 105

8 Starting an iOS Application 107
Planning the App 107
Model-View-Controller 107
The Model 108
The Views 108
The Controllers 110
Starting a New iOS Project 110
Target Editor 111
What’s in the Project 112
Summary 114

9 An iOS Application: Model 117
Implementing the Model 117
Entities 118
Attributes 118
Relationships 120

Managed-Object Classes 123
- Creating the Classes—the Wrong Way 124
- Why Doing It Xcode's Way Is a Mistake 125
- The Right Way—`mogenerator` 126

Preparation 128
- Utilities 129
- Extensions 129
- `passer_rating` 131

Specializing the Core Data Classes 132
- Putting `Game` to Work 132
- Putting `Passer` to Work 133
- Some Test Data 134
- Source Control and Product Files 136

Making the Model Easier to Debug 139

Summary 139

10 An iOS Application: Controller 141

Renaming Symbols in Objective-C 141
- Refactoring the Name of an Objective-C Method 142
- Refactoring a Class Name 142

Renaming a Class in Swift 144

Editing the View Controller 144
- The Table View 145
- Setting Up the Passer List 146
- Creating a New Passer 147

Live Issues and Fix-it 148

The Real Passer Rating 149
- Another Bug 149
- Running Passer Rating 154

Summary 155

11 Building a New View 157

The Next View Controller 157
- If You Want to Add a View Controller 157
- Storyboards, Scenes, and Segues 158

Building a View 161
- Outlets and Assistants, in Passing 162
- The Billboard View 164

Linking Views to a View Controller 166
Auto Layout for the Nonce 167
Lots of Labels 169
Cleaning Up 171
The Table View 174
Outlets 175
Hooking Up the Outlets 177
Checking Connections 177
Connecting `GameListController` 178
Code Completion and Snippets 180
Code Snippets 181
Testing the Billboard View 183
Summary 184

12 Auto Layout in a New View 185
Why Auto Layout? 185
Limitations of Autoresizing 185
Auto Layout 186
The Thing to Remember 186
The Player Billboard, Revisited 186
Why You Should Do More 187
Factoring Layout into Subviews 188
The Playground 189
`StatView` 191
Installing `StatView` 196
Planning Constraints 197
Two Line Counts, Two Labels 200
Constraints for Real 202
Default (Any/Any) 202
Any Height (not Compact) 203
Landscape (wAny/hCompact) 203
Chasing Issues 203
A Tweak 204
Summary 205

13 Adding Table Cells 207

The Game Table 207

Outlets in the Table View 207

Adding Required Protocol Methods 208

Adding Model-to-View Support 210

A Prototype Cell 211

The Game Table: First Run 211

A Custom Table Cell 214

Adding Some Graphics 217

A Cell with an Image in It 217

Hooking the Image View to the Images 218

The Assets Catalog 219

Adding Images to the Assets Catalog 220

Icons and Launch Displays 221

Summary 225

14 Adding an Editor 227

The Plan 227

Adding a Modal Scene 227

An Embedded View Controller 229

Linking the Editor to the Passer List 231

Static Table Cells 232

The Editor View Controllers 233

The Editor Table 233

Passing the Data to the Editor 235

Getting the Data Back 237

Segues 239

Summary 240

15 Unit Testing 243

The Test Navigator 244

Testing the CSV Reader 246

The CSV Test Code 247

Test Data 252

Running the Tests 252

Testing and the Debugger 254

Adding a Test Class 256

Asynchronous Tests 260
 Testing Asynchronous Code 260
 Documentation of Last Resort 261
XCTest Assertions 261
 Simple Tests 262
 Equality 262
 Exceptions 263
Summary 264

16 Measurement and Analysis 265
Speed 265
 The Debug Navigator 266
 Instruments 268
XCTest and Performance 276
Memory 277
Summary 278

17 An iOS Extension 279
Adding the Today Target 280
Designing the Widget 281
 Data Access 282
A Shared Library in a Framework 285
The Today Extension 290
Build Dependencies 294
The Result 295
Summary 296

18 Provisioning 297
Apple Developer Programs 297
 General (App Store) Program 298
 Enterprise Program 298
Provisioning for iOS 299
 What You'll See 300
 Registering Your App 300
 Protecting Your Assets 303
 Prerelease Distributions 304

The Capabilities Editor 306
- OS X-only Capability 306
- Capabilities for Both iOS and OS X 306
- iOS Capabilities 307

OS X Sandboxing 308
- Why Sandbox? 310
- Why Not Sandbox? 310

Gatekeeper and Developer ID 311
- Getting a Developer ID 311
- Using Developer ID 312
- Limitations 313

Distribution Builds 314
- Basic Build Settings 314
- Adjusting the Build Settings 315
- The Build 317

Summary 318

III Xcode for Mac OS X 319

19 Starting an OS X Application 321

The Goal 321

Getting Started 322

Model 325
- Porting from iOS 326
- Adding an Entity 326

Wiring a Menu 330
- Target/Action 331
- First Responder 332
- Loading Data into `LeagueDocument` 333
- Adapting to a Managed Document 334
- Testing the Command 335
- Identifying a Type for League Data 336
- Specifying How the App Handles League Files 338
- Application and Document Icons 339

Summary 341

20 Bindings: Wiring an OS X Application 343

Storyboard Segues in OS X 343

Building the Document Window 345

- Loading the Window 345
- A Table View 347

Filling the Table—Bindings 350

- Object Controllers 352
- Binding the Table to the Teams 354
- Binding the Columns to Team Properties 355

The Arc of League Document Data 357

- From League Table to Source List 357
- Capturing the Team Selection 359
- From Team to Tables 361
- The Passer Section 363

Summary 371

21 Localization 373

How Localization Works 373

Adding a Localization 374

- Base Localization 374
- Why Base Localization? 375

Something Worth Localizing 376

- Game Detail View: Layout 376
- Game Detail View: Code 378
- Modules and Namespaces 382

Localizing for French 382

- Adding a Locale 383
- Starting Simple: `Credits.rtf` 385
- Localizing `Main.storyboard` 389
- Localizing Resources 392
- Localizing Program Strings 394
- `genstrings` 395
- `xliff` Files 396
- The Rest 397

Localizing System Strings 398

Summary 400

22 Bundles and Packages 401
A Simple Package: RTFD 401
Bundles 403
Application Bundles 403
The `Info.plist` File 405
  Localizing `Info.plist` 406
`Info.plist` Keys for Applications 406
  Keys for Both iOS and OS X 406
  Keys for OS X 409
  Keys for iOS 412
Summary 415

23 Property Lists 417
Property List Data Types 417
Editing Property Lists 419
  The Property List Editor 422
  Why Not the Property List Editor? 427
Other Formats 429
  Text Property Lists 429
  Binary Property Lists 430
  JSON 430
Specialized Property Lists 431
Summary 432

IV Xcode Tasks 433

24 Documentation in Xcode 435
Quick Help 435
  Inspector 435
  Popover 436
Open Quickly 437
Help 438
The Documentation Window 439
  The Navigator Sidebar 439
  The Table of Contents Sidebar 440
  Class Info 440
  Searching and Navigation 440

Keeping Current 444
Your Own Quick Help 446
C-Family Documentation 446
Doxygen 449
Running Doxygen 454
Installing a Docset 455
Swift and reStructuredText 456
Summary 458

25 The Xcode Build System 459
How Xcode Structures a Build 459
Build Variables 462
Settings Hierarchy 463
Levels 464
Editing Build Variables 465
Configurations 466
Adjusting Configurations 466
Configuration Files 468
Creating a Configuration File 468
SDK- and Architecture-Specific Settings 469
Preprocessing `xcconfig` Files 470
Command-Line Tools 471
`xcodebuild` 471
`xcode-select` 472
`xcrun` 473
Custom Build Rules 474
Builds in the Report Navigator 476
A Simple Build Transcript 476
Summary 487

26 Instruments 489
What Instruments Is 489
Running Instruments 490
The Trace Document Window 492
Toolbar 492
Track Area 494
Detail Area 495
Extended Detail Area 495
Library 499

- Tracing 500
  - Recording 500
  - Saving and Reopening 503
  - Tracing without Instruments 504
- The Instruments 504
  - Behavior 504
  - Core Data 505
  - Dispatch 505
  - Filesystem 505
  - Graphics 506
  - Input/Output 506
  - Master Tracks 506
  - Memory 506
  - System 507
  - System—iOS Energy Instruments 509
  - Threads/Locks 509
  - Trace 509
  - UI Automation 510
  - User Interface 510
- Custom Instruments 511
- The Templates 512
  - All Platforms 513
  - iOS Only 513
  - Mac Only 513
- Summary 514

27 Debugging 515

- Scheme Options 515
  - Info 515
  - Arguments 516
  - Options 516
  - Diagnostics 518
- Doing More with Breakpoints 518
- View Hierarchy 521
- The `lldb` Command Line 523
- Tips 525
- Summary 528

28 Snippets 531
Tricks 531
General 531
Code-Folding Ribbon 535
The Assistant Editor 536
Instruments and Debugging 538
Building 539
Traps 541

V Appendixes 547

A Some Build Variables 549
Useful Build Variables 550
Environment 551
Code Signing 552
Locations 553
Compiler Settings 556
Other Tools 559
`Info.plist` 559
Search Paths 561
The `DEVELOPER_` Variables 561
Source Trees 562

B Resources 565
Books 565
Books about Swift 566
On the Net 567
Forums 567
Mailing Lists 568
Developer Technical Support 568
Sites and Blogs 569
Face to Face 570
Meetings 570
Classes 570
Other Software 570
Text Editors 571
Helpers 572
Package Managers 574

Version Control 575
AppCode 576
Alternatives to Cocoa 576

Index 579

Acknowledgments

Only part of the effort that went into putting *Xcode 6 Start to Finish* into your hands was spent at a text editor. I am indebted to those without whom this book could not have been made. Many of them appear in the formal production credits; they deserve better-than-formal thanks.

Trina MacDonald went through enormous pains to turn a new edition around quickly in the face of the many changes in Xcode. This was not a sure thing; she made sure it happened. Thank you.

Lori Hughes stepped in with skill and good humor to give us a head start in getting the manuscript ready to go on a tight schedule. Julie Nahil, the production manager, stepped in early to make this possible, in addition to her job-title work of turning the manuscript I submitted into the book I'd hoped for.

Olivia Basegio made sure the contracts, correspondence, (and advance payments!) all got through. She guided the reviewers through their work while the book was still in unassembled pieces on the ground.

The reviewers, Chris Zahn, Gene Backlin, and Josh Day, saved me much embarrassment, and made this a much better work than it started. Errors remain. Some are intentional, some not; they are all my own.

Stephanie Geels, the copy editor, made the prosecution of typos, grammar, and which instance of a particular word gets which typeface, more fun than you'd think. In fact, fun.

A full-time day job is not an author's best friend (except for the part about paying the rent). Alan Takaoka, my boss in the Web Services department of IT Services, The University of Chicago, got me three- and four-day weekends while I wrote. I promised to keep all my meetings and deadlines while I worked on the book, but somehow none of them fell on a Monday or Friday. Cornelia Bailey, who manages most of my projects, kept rescheduling them.

Bess and Kate bore more than daughters should of my doubts and frustrations, and were simply confident that I would do fine—which was all they needed to do.

This page intentionally left blank

About the Author

Fritz Anderson has been writing software, books, and articles for Apple platforms since 1984. This is his fifth book.

He has worked for research and development firms, consulting practices, and freelance. He was admitted to the Indiana bar, but thought better of it. He is now a senior iOS developer for the Web Services department at The University of Chicago. He has two daughters.

This page intentionally left blank

Introduction

Welcome to *Xcode 6 Start to Finish*! This book will show you how to use Apple's integrated development environment to make great products with the least effort.

Xcode 6 is the descendant of a family of development tools dating back more than 20 years to NeXT's ProjectBuilder. It started as a text editor, a user-interface designer, and a front end for Unix development tools. It has become a sophisticated system for building applications and system software, with a multitude of features that leverage a comprehensive indexing system and subtle incremental parser to help you assemble the right code for your project, and get it right the first time.

That much power can be intimidating. My aim in *Xcode 6 Start to Finish* is to demystify Xcode, giving you a gradual tour through examples that show you how it is used day to day. Don't let the gentle approach fool you: This book will lead you through the full, best-practices workflow of development with Xcode 6. There are no "advanced topics" here—I'll show you version control and unit testing in their proper places in the development process.

How This Book Is Organized

First, a word on my overall plan. This is a book about developer tools. If it teaches you something about how to use the Cocoa frameworks, or something about programming, that's fine, but that's incidental to showing you the Xcode workflow. There are many excellent books and other resources for learning the frameworks; you'll find many of them listed in Appendix B, "Resources."

Every tour needs a pathway, and every lesson needs a story. The first three parts of this book demonstrate Xcode through three applications—a command-line tool, an iOS app, and an OS X application—that calculate and display some statistics in American football. None of the apps are very useful; the graphical apps run almost entirely on sample data. But they demand enough of the development tools to give you a solid insight into how to use them.

Xcode supports some technologies, like Core Data and OS X bindings, that are *not* for beginners. *Xcode 6 Start to Finish* dives straight into those techniques, ignoring conceptually simpler approaches, so I can demonstrate how Xcode works. Other "advanced" techniques, like unit testing and version control, appear at the points where best practices require them. This will be the workflow as Xcode supports it.

I'm using applications for iOS and OS X as examples, but read both Parts II and III, even if you're only interested in one platform. The applications are only *stories*; the techniques apply to both platforms.

First Steps

In Part I, I'll take you from installing Xcode and running your first project through basic debugging skills. You'll work through a small command-line application. The application may be simple, but you'll learn foundational skills you'll need before adding the complexity of graphical apps.

- **Chapter 1, Getting Xcode**—Some things to consider before you download Xcode 6; two ways to download and install it.
- **Chapter 2, Kicking the Tires**—Your first look at Xcode, setting up a trivial project and running it.
- **Chapter 3, Simple Workflow and Passive Debugging**—Write, build, and run a simple application, and respond to a crash.
- **Chapter 4, Active Debugging**—Take charge of debugging by setting breakpoints and tracing through the program. I'll show you how to organize your workspace.
- **Chapter 5, Compilation**—A pause to describe the process of building an application.
- **Chapter 6, Adding a Library Target**—Add a library target to a project, and learn how to build a product from multiple targets.
- **Chapter 7, Version Control**—Why source control is important, and how to take advantage of Xcode's built-in support for versioning through Git and Subversion.

The Life Cycle of an iOS Application

Part II tells the story of a small iPhone app, and how to use Apple's developer tools to build it. It introduces you to the graphical editor for user interfaces, and shows how to profile an app to optimize its speed and memory burden.

- **Chapter 8, Starting an iOS Application**—You'll start by creating an iOS project, and learn the Model-View-Controller design at the core of Cocoa on iOS and OS X alike.
- **Chapter 9, An iOS Application: Model**—Design a Core Data schema and supplement it with your own code.
- **Chapter 10, An iOS Application: Controller**—Create a controller to link your model to the on-screen views. On the way, I'll tell you about refactoring, and Xcode's continual error-checking.
- **Chapter 11, Building a New View**—Design the user-interface views for your app with the integrated Interface Builder, and take advantage of source-code completion.

- **Chapter 12, Auto Layout in a New View**—In Xcode 6, Auto Layout is more about getting things done than fighting the tools. Learn how to make Cocoa layout do what you want.
- **Chapter 13, Adding Table Cells**—While adding an in-screen component to your app, you'll debug memory management, and control how Xcode builds, runs, and tests your apps through the Scheme editor.
- **Chapter 14, Adding an Editor**—Add an editor view, and get deep into Storyboard.
- **Chapter 15, Unit Testing**—Unit testing speeds development and makes your apps more reliable. I'll show you how Xcode supports it as a first-class part of the development process.
- **Chapter 16, Measurement and Analysis**—Use Instruments to track down performance and memory bugs.
- **Chapter 17, An iOS Extension**—Create a system-wide extension and a shared library to bring your app's value beyond its own screen.
- **Chapter 18, Provisioning**—Behind the scenes, the process of getting Apple's permission to put apps on devices is complicated and temperamental. I'll show you how Xcode saves you from most of the pain, and give you a few tips on how to get out if it backs you into a corner.

Xcode for Mac OS X

Part III shifts focus to OS X development. Some concepts are more important to OS X than iOS, but you'll be learning techniques you can use regardless of your platform.

- **Chapter 19, Starting an OS X Application**—Carrying iOS components over to OS X; what the responder chain is, and how Interface Builder makes it easy to take advantage of it.
- **Chapter 20, Bindings: Wiring an OS X Application**—As you build a popover window, you'll use OS X bindings to simplify the link between your data and the screen. You'll also encounter autosizing, a legacy technique for laying out view hierarchies.
- **Chapter 21, Localization**—How you can translate your Mac and iOS apps into other languages.
- **Chapter 22, Bundles and Packages**—You'll master the fundamental structure of most Mac and iOS products, and how both platforms use the `Info.plist` file to fit apps into the operating system.
- **Chapter 23, Property Lists**—Learn the basic JSON-like file type for storing data in both OS X and iOS.

Xcode Tasks

By this point in the book, you'll have a foundation for digging into the details of the Xcode toolset. Part IV moves on to topics that deserve a more concentrated treatment than Parts II and III.

- **Chapter 24, Documentation in Xcode**—How Xcode gives you both immediate help on API, and browsable details on the concepts of Cocoa development. Find out how you can add your own documentation to the system.
- **Chapter 25, The Xcode Build System**—I'll show you the fundamental rules and tools behind how Xcode goes from source files to executable products.
- **Chapter 26, Instruments**—Using Apple's timeline profiler, you can go beyond basic performance and memory diagnostics to a comprehensive look at how your program uses its resources.
- **Chapter 27, Debugging**—How to use breakpoint actions and conditions to eliminate in-code diagnostics. You'll also find a tutorial on the `lldb` debugger command set, for even closer control over your code.
- **Chapter 28, Snippets**—A roundup of tips, traps, and features to help you get the most from the developer tools.

Appendixes

The appendixes in Part V contain references to help you master the build system, and find help and support.

- **Appendix A, Some Build Variables**—The most important configuration and environment variables from Xcode's build system.
- **Appendix B, Resources**—A compendium of books, tools, and Internet resources to support your development efforts.

About Versions

This book was finished in the fall of 2014. *Xcode 6 Start to Finish* is written to early versions of 10.10, iOS 8.2, and Xcode 6.2.

About the Code

Xcode 6 Start to Finish has many examples of executable code—it's about a system for creating code and running it. My goal is to teach *workflow.* What the code itself does is practically incidental. In particular, be aware: **Much of the code in this book will not run as initially presented.** *Xcode 6 Start to Finish* is about the development process, most of which (it seems) entails prosecuting and fixing bugs. You can't learn the workflow unless you learn how to respond to bugs.

So I'll be giving you buggy code. You may find it painful to read, and if you try to run it, it will be painful to run. Trust me: It's for a reason.

Also, sample code for this book is available at `informit.com/title/9780134052779` (register your book to gain access to the code). You'll find archives of the projects in this book as they stand at the end of each chapter. With very few exceptions—I'll make them very clear—if you want the project as it stands at the *start* of a chapter, you should use the archive for the *end* of the previous chapter.

The chapter archives do not include version-control metadata. If you are following along with the examples, and using Git (or Subversion) for your work, copy the changes into your own working directory. If you replace your directory with a sample directory, you'll lose your version history.

Conventions

This book observes a number of typographic and verbal conventions.

- Human-interface elements, such as menu items and button labels, are shown **like this**.
- File names and programming constructs are shown `like this`. This will sometimes get tricky as when I refer to the product of the "Hello World" *project* (plain text, because it's just a noun) as the *file* `Hello World`.
- Text that you type in will be shown **`like this`**.
- When I introduce a new term, I'll call it out *like this*.

I'll have you do some command-line work in the Terminal. Some of the content will be wider than this page, so I'll follow the convention of breaking input lines with backslashes (\) at the ends. I'll break long output lines simply by splitting them, and indenting the continuations. When that output includes long file paths, I'll replace components with ellipses (. . .), leaving the significant parts.

For its first 20 years, the Macintosh had a one-button mouse. (Don't laugh—most purchasers didn't know what a mouse *was*; try pushing the wrong button on an old Mac mouse.) Now it has four ways to effect an alternate mouse click: You can use the right button on an actual mouse (or the corner of the mouse where the right button would be); you can hold down the Control key and make an ordinary click; you can hold down two fingers while clicking on a multi-touch trackpad (increasingly common even on desktop Macs); or you can tap at a designated corner of a multi-touch trackpad. And there are more variations available through System Preferences. Unless the distinction really matters, I'm simply going to call it a "right-click" and let you sort it out for yourself.

This page intentionally left blank

3 Simple Workflow and Passive Debugging

This chapter begins your use of Xcode in earnest. Here's where I introduce the problem that is the basis for all of the example code in this book.

The problem is the calculation of *passer ratings*. In American/Canadian football, quarterbacks advance the ball by throwing (passing) it to other players. The ball may be caught (received, a good thing) by someone on the quarterback's own team, in which case the ball is advanced (yardage, more is better), possibly to beyond the goal line (a touchdown, the object of the game); or it may be caught by a member of the opposing team (intercepted, a very bad thing).

But those are four numbers, and everybody wants a figure-of-merit, a single scale that says (accurately or not) who is the better passer. The National Football League and the Canadian Football League have a formula for passer ratings, yielding a scale running from 0 to (oddly) 158.3. A quarterback who rates 100 has had a pretty good day.

Creating the Project

As in Chapter 2, "Kicking the Tires," you'll start with a command-line project. Start Xcode and click **Create a new Xcode project**, or select **File →New →Project. . .** (⇧⌘**N**). In the New Project assistant sheet, select an OS X Command Line Tool, and name the tool **`passer-rating`**; for **Language**, once again choose **C**.

Another difference: When you are shown the get-file sheet to select the location for the new project, check the box labeled **Create Git repository on**, and select **My Mac**. Git is a *version-control system*, an essential part of modern development. You'll learn all about it in Chapter 7, "Version Control," but now is the time to start.

Note

Are you ever going to change anything in a project? Get it under version control. Seriously. Your work will be safer, and you'll do it faster.

Again, you'll be shown target settings, which you can ignore for now. Instead, mouse over to the Project navigator at the left side of the Workspace window, and select `main.c`.

Delete everything in the `main()` function but its outer braces, and replace the body of the function so the file reads thus (keep the comments at the top of the file):

```
#include <stdio.h>
#include "rating.h"     //  Yet to create; initially an error

int main(int argc, const char * argv[])
{
    int         nArgs;
    do {
        int         comps, atts, yards, TDs;
        printf("comps, atts, yards, TDs: ");
        nArgs = scanf("%d %d %d %d %d",
                      &comps, &atts, &yards, &TDs);
        if (nArgs == 5) {
            float  rating = passer_rating(comps, atts, yards, TDs);
            printf("Rating = %.1f\n", rating);
        }
    } while (nArgs == 5);

    return 0;
}
```

You'll notice that as you type closing parentheses, brackets, and braces, the corresponding opening character is briefly highlighted in yellow.

The rating calculation itself is simple. Put it into a file of its own: Select **File →New →File. . .** (⌘**N**). You'll be presented with the New File assistant sheet; see Figure 3.1. Navigate the source list on the left, and the icon array on the right thus: OS X → Source → C File.

Click **Next**, and use the save-file sheet that appears to name the file **`rating`** (Xcode will append `.c` automatically).

The save-file sheet has two custom controls. The **Group** popup lets you place the new file in the Project navigator (the source list at the left of the project window). Roughly, groups are simply ways to organize the Project inspector list; they have no influence on how the new file will be placed on-disk. Make sure the passer-rating group is selected.

Second is **Targets**, a table that for now has only one row, **passer-rating**. A target is a group of files and settings that combine to build a product. A file that isn't part of a target isn't used to build anything. Make sure that **passer-rating** is checked.

Note

It's easy to get the target assignment wrong. Xcode 6 sets the same targets for new files as the ones for the last files that were added. If you forget to set the proper targets, you won't know about it until your app fails to build or mysteriously crashes because a needed resource wasn't included.

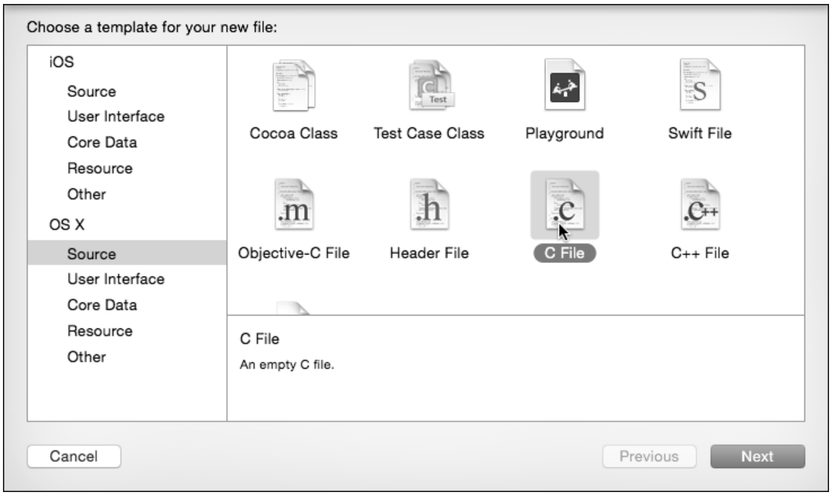


Figure 3.1 The New File assistant sheet offers many templates you can use to start a new file. Select the category from the source list on the left, and pick the template from the array of icons on the right.

Here's what goes into `rating.c`:

```
#include "rating.h"

static
double pinPassingComponent(double component)
{
    if (component < 0.0)
        return 0.0;
    else if (component > 2.375)
        return 2.375;
    else
        return component;
}

float
passer_rating(int comps, int atts, int yds, int tds, int ints)
{
    //  See http://en.wikipedia.org/wiki/Quarterback_Rating


    double      completionComponent =
                  (((double) comps / atts) * 100.0 - 30.0) / 20.0;
    completionComponent = pinPassingComponent(completionComponent);
    double      yardageComponent =
                   (((double) yds / atts) - 0.3) / 4.0;
                   //  intentional bug
    yardageComponent = pinPassingComponent(yardageComponent);

    double      touchdownComponent =
                   20.0 * (double) tds / atts;
    touchdownComponent = pinPassingComponent(touchdownComponent);

    double      pickComponent =
                   2.375 - (25.0 * (double) ints / atts);
    pickComponent = pinPassingComponent(pickComponent);

    double retval =  100.0 * (completionComponent +
                             yardageComponent +
                             touchdownComponent +
                             pickComponent) / 6.0;
    return retval;
}
```

Note

You see a few bugs in this code. Well done. Throughout this book, I'm going to give you some buggy code to illustrate debugging techniques. Just play along, okay?

By now, you've missed a couple of braces, and you are tired of tabbing to get the extra level of indenting. Xcode can do this for you—it's among the features I had you turn off in the last chapter.

Open the Preferences window (**Xcode →Preferences**, **⌘ comma**) and select the **Text Editing** panel. In the **Editing** tab, check **Code completion: Automatically insert closing "}"**. In the **Indentation** tab, check **Syntax-aware indenting: Automatically indent based on syntax**.

Now type an open brace in your code, at the end of a line. So what, it's a brace. Now press Return. Xcode adds two lines: Your cursor is now at the next line, indented one level, and a matching closing brace appears on the line after that.

Finally, you've noticed that both `main.c` and `rating.c` refer to a `rating.h`, which notifies `main()` of the existence of the `passer_rating` function. Press **⌘N** again, and choose Header File from the source files. Name it **`rating`**, and put this into it:

```
#ifndef passer_rating_rating_h
#define passer_rating_rating_h

float passer_rating(int comps, int atts, int yds,
                    int tds, int ints);
#endif
```

Note

Place header files wherever you like among the project groups, but don't add them to any targets. There are exceptions; if you need to do it, you'll know. Chapter 6, "Adding a Library Target," has more.

Click **Create**.

Building

That's enough to start. Let's try to run it. It's easy: Click the **Run** button at the left end of the toolbar, or select **Product →Run** (**⌘R**). It doesn't matter if you haven't saved your work; by default Xcode saves everything before it attempts a build. Xcode chugs away at your code for a bit. . . and stops.

- A heads-up placard flashes, saying "Build Failed."
- The Navigator area switches to the Issue navigator, which shows two items under `main.c`. (If the Issue navigator doesn't appear, click the fourth tab at the top of the Navigator area.) One is tagged with a yellow triangle (a warning), and the other with a red octagon (an error). These include descriptions of the errors (Figure 3.2, top).
- When you click one of the items, the editor highlights two lines in `main.c`. The line that triggered the warning is tagged in yellow, with a banner describing the warning; the error line is in red, with a banner of its own (Figure 3.2, bottom).

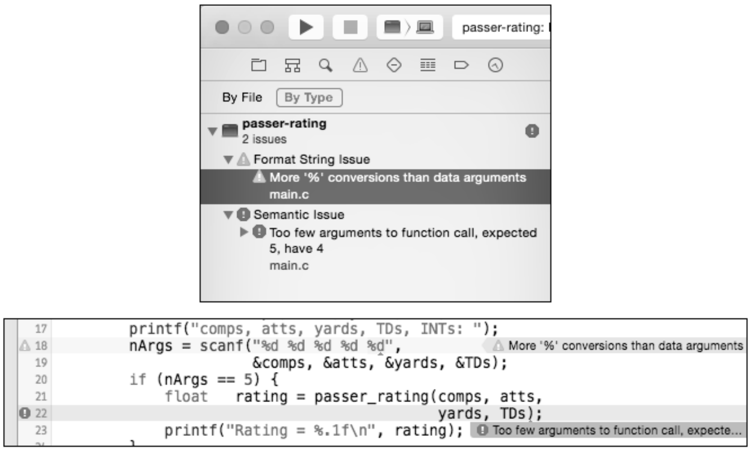
It seems the only place where I remembered about interceptions was the format string of the `scanf` call. The compiler was smart enough to match the number of format specifiers to the number of arguments of the `scanf` and complain. Similarly, I left off the last parameter to `passer_rating`, which is an outright error.

Note

For a compiler, an *error* is a flaw in your source that makes it impossible to translate your code into executable form. The presence of even one error prevents Xcode from producing a program. A *warning* indicates something in your source that *can* be translated but will probably result in a bug or a crash when you run it. Experienced developers do not tolerate warnings; there is even a compiler option to make a build fail upon a warning just as though it were an error. Don't ever ignore a warning.

Note

Need a reminder of what `passer_rating` does with its parameters? Try this: While `main` is visible in the Editor area, hold down the Command key and point the mouse at the symbol `passer_rating`. You'll see it turns blue and acquires an underline, as if it were a link on a standard web page. Click it: The editor jumps to the declaration of `passer_rating`. You can jump back by clicking the back-arrow button in the upper-left corner of the editor; by pressing ^⌘←; or by swiping right across the editor area with two fingers, if you've enabled the gesture in System Preferences. (Option-clicking the name



passer-rating
2 issues
Format String Issue
More '%' conversions than data arguments
main.c
Semantic Issue
Too few arguments to function call, expected 5, have 4
main.c

```
        printf("comps, atts, yards, TDs, INTs: ");
        nArgs = scanf("%d %d %d %d %d",            More '%' conversions than data arguments
                      &comps, &atts, &yards, &TDs);
        if (nArgs == 5) {
            float   rating = passer_rating(comps, atts,
                                           yards, TDs);
            printf("Rating = %.1f\n", rating);  Too few arguments to function call, expecte...
```

Figure 3.2 (top) When Xcode detects build errors, it opens the Issue navigator to display them. (bottom) Clicking an issue focuses the editor on the file and line at which the issue was detected.

gets you a popover that tells you `passer_rating` was declared in `rating.h`. More on this in Chapter 24, "Documentation in Xcode.")

You can dash off a fix very quickly:

```
do {
    int         comps, atts, yards, TDs, INTs;
    printf("comps, atts, yards, TDs, INTs: ");
    nArgs = scanf("%d %d %d %d %d",
                  &comps, &atts, &yards, &TDs, INTs);
    if (nArgs == 5) {
        float   rating = passer_rating(comps, atts, yards,
                                       TDs, INTs);
        printf("Rating = %.1f\n", rating);
    }
} while (nArgs == 5);
```

To be conservative (I don't want Xcode to run the program if a warning remains), **Product** →**Build** (⌘**B**) will compile and link `passer-rating` without running it. You needn't have worried: It compiles without error, displaying a "Build Succeeded" placard.

Note

The Issues navigator will show a warning or two. Let's play dumb and carry on.

Running

Now you have something runnable. Run it (**Run** button, first in the toolbar; or ⌘**R**).

There is a transformation: The Debug area appears at the bottom of the window; and the **View** control in the toolbar highlights its middle button to match the appearance of the Debug area (Figure 3.3).

The right half of the Debug area is a console that you'll be using to communicate with the `passer-rating` tool. If all has gone well, it should be showing something like this:

```
comps, atts, yards, TDs, INTs:
```

. . . which is just what the `printf()` was supposed to do. `passer-rating` is waiting for input, so click in the console pane and type:

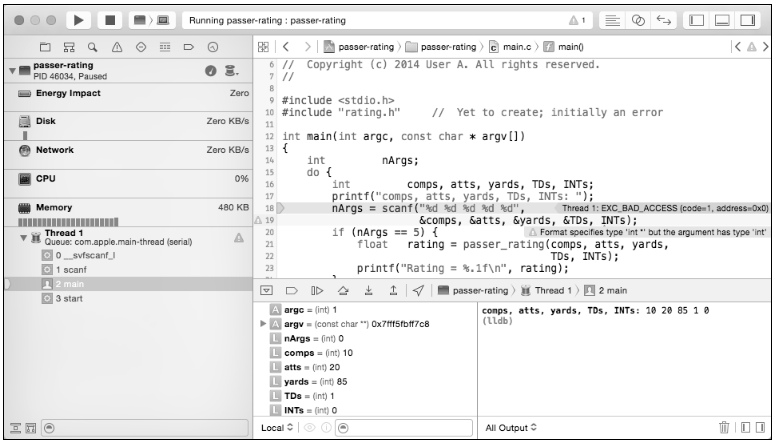
```
10 20 85 1 0 <return>
```

Something went wrong. `passer-rating` crashed. `lldb`, the debugging engine, takes control, and the debugging displays fill up.

- In the Navigator area, the Debug navigator appears, showing the status of the program when it crashed. The upper part of the navigator contains performance bar charts that will be useful when you get to more complex projects. Ignore them for the moment.

  What's left is a *stack trace*, showing the chain of function calls that led to the crash. The top line, labeled 0, is the name of the function, `__svfscanf_l`, where the crash occurred; if you click it, you can see the assembly code (the source isn't available) focused on the very instruction that went wrong. The next line is `scanf`, which you recognize as the function you called from `main`, the function on the next line.

  Xcode identifies `main` as your work by flagging it with a blue head-and-shoulders icon. Click that line to see what your code was doing when the crash occurred.



```
//  Copyright (c) 2014 User A. All rights reserved.
//

#include <stdio.h>
#include "rating.h"     //  Yet to create; initially an error

int main(int argc, const char * argv[])
{
    int         nArgs;
    do {
        int         comps, atts, yards, TDs, INTs;
        printf("comps, atts, yards, TDs, INTs: ");
        nArgs = scanf("%d %d %d %d %d",        Thread 1: EXC_BAD_ACCESS (code=1, address=0x0)
                      &comps, &atts, &yards, &TDs, INTs);
        if (nArgs == 5) {                      Format specifies type 'int *' but the argument has type 'int'
            float   rating = passer_rating(comps, atts, yards,
                                           TDs, INTs);
            printf("Rating = %.1f\n", rating);
```

```
argc = (int) 1
argv = (const char **) 0x7fff5fbff7c8
nArgs = (int) 0
comps = (int) 10
atts = (int) 20
yards = (int) 85
TDs = (int) 1
INTs = (int) 0
```

```
comps, atts, yards, TDs, INTs: 10 20 85 1 0
(lldb)
```

Figure 3.3 Running an app in Xcode opens the Debug area (at the bottom of the project window).

- In the Debug area at the bottom of the window, the left-hand pane fills with the names of variables and their values. You see, for instance, "**atts** = (int) 20," which is just what you entered. Chapter 4, "Active Debugging," discusses this more.
- The Editor area has the most interesting change: A green flag at the left margin and a green banner at the right of the call to `scanf`. The banner says, "Thread 1: `EXC_BAD_ACCESS` (code=1, address=0x0)." The message may be truncated; you can see the full text in a tooltip that appears if you hover the mouse cursor over the banner.

Note

Xcode has a lot of these banners; often they are the only way it will convey important messages. You will probably set your editor fonts to the smallest you can comfortably read so you can see more of your work at once. The banners are one line high, and their margins take up some space, so the text in them may be *smaller* than you can comfortably read. The only solution is to select larger fonts for everyday use; see the **Fonts & Colors** panel of the Preferences window.

Simple Debugging

`EXC_BAD_ACCESS` entails the use of a bad pointer, perhaps one pointing into memory that isn't legal for you to read or write to. (The 64-bit virtual-memory management on OS X and modern iOS is set so any address that might be derived from a 32-bit integer is illegal, making it harder to cross `int`s and pointers.) Reexamine the line in `main` that crashed the application and allow a scale to fall from your eyes:

```
        nArgs = scanf("%d %d %d %d %d",
                      &comps, &atts, &yards, &TDs, INTs);
```

`scanf` collects values through pointers to the variables that will hold them. This call does that for all values except `INTs`, which is passed by value, not by reference. One of the warnings I had you ignore said exactly that: "Format specifies type '(int *)' but the argument has type 'int'." Simply inserting an `&`

```
        nArgs = scanf("%d %d %d %d %d",
                      &comps, &atts, &yards, &TDs, &INTs);
```

should cure the problem. Sure enough, running `passer-rating` again shows the crasher is gone:

```
comps, atts, yards, TDs, INTs: 10 20 85 1 0
Rating = 89.4
comps, atts, yards, TDs, INTs: <^D>
```

With the **^D** keystroke, the input stream to `passer-rating` ends, the program exits, and the Debug area closes.

You ordinarily wouldn't want to run or debug a program under Xcode if another is running. Instead, you'd like the incumbent app to clear out. There are four ways to do this.

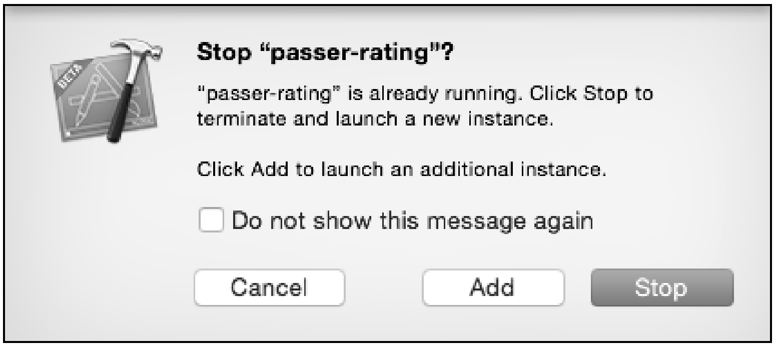
- Simply let the app exit on its own, as you did when you used **^D** to stop `passer-rating`, or would by selecting the **Quit** command in an OS X application. But this doesn't work for iOS apps, which in principle never quit. You'll have to use one of the other techniques.
- Click the **Stop** button in the toolbar.
- Select **Product** → **Stop** (**⌘period**).
- Tell Xcode to run an application, the same or a different one, and click **Stop** in the alert sheet that appears. See Figure 3.4.

That alert sheet also offers an **Add** button, which will run and debug the new process without quitting the old one. Xcode will start a new execution context: You can switch between them using the jump bar at the top of the Debug area, and the **Stop** button in the toolbar becomes a menu you can use to select which instance to stop.

Note

Don't check the **Do not show this message again** box. There will come a time when you want to continue the execution of a program you are debugging, and rather than clicking the tiny button the debugger provides, you'll go for the large, friendly **Run** button in the toolbar. That time comes to me frequently. The add-or-stop sheet is the only thing standing between you and the ruin of your debugging session.

For the moment, you're done: The `scanf` call will return fewer than five inputs if the standard input stream ends. You end it as you would in a terminal shell, by pressing **^D**.



Stop "passer-rating"?

"passer-rating" is already running. Click Stop to terminate and launch a new instance.

Click Add to launch an additional instance.

☐ Do not show this message again

Figure 3.4 When you tell Xcode to run an application while it already has an application running, it displays a sheet asking what you want to do with the existing app. Normally, you'd click **Stop**, but there is also the option to **Add** the new instance to run concurrently with the old one.

Note

M and **A** badges are accumulating in the Project navigator. These have to do with version control. Nothing is wrong. Patience! I'll get to it in Chapter 7, "Version Control."

Summary

This chapter stepped you up to writing and running a program of your own. It introduced the first level of debugging: what to do when your program crashes. It turns out that Xcode offers good facilities to help you analyze a crash without you having to do much. You accepted Xcode's guidance, quickly traced the problem, and verified that your fix worked.

But we're not done with `passer-rating`. There are still bugs in it, and this time you'll have to hunt for them.

Index

A

.a (static libraries), 69, 539
Accessibility package, 14
Accessory setting
  details, 232
  editors, 211
Accounts panel
  Apple ID, 298
  Developer ID, 311
  iOS provisioning, 300–301, 303
  remote repositories, 88–89
  version control, 80
ACTION variable, 551
Actions
  for wiring menus, 331
  xcodebuild, 471–472
Activate/Deactivate Breakpoints, 39
Active Allocation Distribution style, 496
Activity Monitor instrument, 507–508
AD_HOC_CODE_SIGNING_ALLOWED, 552
Ad-hoc distributions, 299, 304, 315–316
Add, 33
Add 4 Constraints, 175
Add an Account, 300
Add Apple ID, 298, 300
Add Breakpoint at Current Line, 36
Add Build Rule, 474–475
Add Entity, 118
Add Exception Breakpoint, 150

Add Files to
  availability, 326
  get-file sheets, 128, 252
  mogenerated directory, 328
  projects, 460
  target pickers, 73
Add Item, 422-423
Add Localization, 383
Add Missing Constraints, 171, 364–365
Add Missing Constraints in Game List Controller, 168
Add Other, 72
Add Relationship, 121
Add Remote, 87–88
Add/Remove Breakpoint at Current Line, 527
Add Run Script Build Phase, 135
Add shortcut, 501
Add Target, 244
Add to targets, 128
Add User-Defined Setting, 550
Added file state, 83
Added folders, 128
Additional exported UTI properties view, 338
addWindowController method, 347
Adobe PhoneGap, 576–577
Advanced attributes for models, 120
Agent applications, 411
Alert sheets
  in debugging, 33
  Git messages, 82
  version control, 96
Alignment of labels, 170–171
All 2 Constraints, 173
All Entities, 452
All Frames in Container, 173
All in builds, 462, 476
All Issues, 476
All Messages, 476
All Processes, 493
Allocation Density style, 496
Allocations & Leaks service, 501
Allocations instruments, 496, 506, 538
Allow Location Simulation, 517
Allows Editing Multiple Values Selection, 355
alltargets, 472
Also create XIB file, 158, 175
Always Presents Application Modal Alerts, 355
Always use deferred mode, 502
Analysis and measurement, 264
  memory, 277–278
  speed. *See* Speed
Analysis message display, 56
analyze action for xcodebuild, 471
Anchor View, 378
Antecedents in makefile goals, 459
Antialiasing, 414
.app directory, 403
App Extensions, 493
App groups, registering, 282–283
App Store
  Enterprise program, 298, 315–316
  OS X applications, 309
  program members, 302
  provisioning, 297–299
  sandboxing, 308-311
  TestFlight distributions, 304–305
  Xcode downloads, 10–13
  Xcode updates, 472
.app suffix, 473
Appcelerator Titanium, 577
AppCode, 576
AppDelegate class
  description, 112, 289–290
  OS X applications, 324

AppIcon image set, 221
Apple Developer Forums, 567
Apple developer programs, 12–13, 297–299
Apple Pay system, 307
AppleGlot tool, 391
Application Data popup, 517
Application IDs, 299
Application Language popup, 388, 517
Application Region popup, 517
Applications
  bundles, 403–405
  icons, 339–340
  Info plist keys for, 406–409
  iOS. *See* iOS
  registering, 300–303
  tests, 260
/Applications directory, 10, 17
applicationWillResignActive method, 290
applicationWillTerminate method, 290
Apps Groups, 307
apropos in lldb, 523
Architecture-specific build settings, 469–470
archive action in xcodebuild, 471
Archives organizer, 312, 317
ARCHS, 556
ARCHS_STANDARD, 466, 556
ARCHS_STANDARD_32_BIT, 556
Arguments panel, 516
Ask on Launch, 294
Assembly listings, 63
assert macro, 531
Assertions
  description, 243
  XCTest, 261–264
Asset catalogs, 484, 559
ASSETCATALOG_COMPILER_APPICON_NAME, 559
ASSETCATALOG_COMPILER_LAUNCHIMAGE_NAME, 559
Assets, protecting, 303
Assets catalog, 219
  adding images to, 220–221
  image sets, 219–220
Assignments (=) in Boolean contexts, 149
Assistant editor
  assembly display, 63
  caller display, 58
  connection checks, 176
  Editor area, 162
  Interface Builder, 159
  jump bar, 164–165
  linking views, 166
  localizations, 389, 393
  Option-key navigation, 437
  overview, 536–538
  Preview, 165–166
  views, 162, 176–177
Associated Domains, 307
Associative arrays, 418, 429
Asynchronous tests, 260–261
At sign (@) notation, 520
atIndexPath method, 146
ATSApplicationFontsPath key, 410
Attributes for models, 118–121
Attributes Inspector, 164–165
Audio package, 14
@author keyword, 448
Authorization in iOS provisioning, 299
Authorized devices in iOS provisioning, 299
Auto Layout, 185
  labels, 171, 200–202
  localizations, 375–378
  overview, 186
  permanent views, 202–205
  planning constraints, 197–200

Auto Layout (*continued*)
purpose, 185–186
size classes, 197
size constraints, 186–188
subviews, 188–197
views, 167–169
Automatic code completion, 22, 28, 180–183
Automatic for Assistant editor, 162
Automatic Reference Counting, 58
Automatic Snapshotting, 507
Automatically continue after evaluating actions, 520
Automation instrument, 510
Autoresize limitations, 185–186
Auxiliary tools, 14

B

@b bold comment format, 448
B2B program, 298
Background Modes, 307
backslashes (\) for breaking input lines, 5
Badges for test navigator, 244
Bar graphs in Debug navigator, 211–213, 266–267
Base localization, 373–376
Baseline performance, 276
Basic button, 462
BBEdit text editor, 571
Beta distributions, 299, 304–306, 315–316
Billboard view
overview, 164–166
size constraints, 186–188
testing, 183–184
Binaries, fat, 482
Binary property lists, 430
Binary stores, 336
Bindings, 343
columns to team properties, 355–357
document window, 345–350
filling, 350–357
game array controller, 369
game table, 369–370
game-table labels, 369
League table to source list, 357–359
object controllers, 352–354
Passer section, 363–364
passer table, 365–367
passer-table labels, 364–365
running, 367–369
storyboard segues, 343–345
tables to teams, 354–355
team selection, 359–361
teams to tables, 361–363
Blame view in Comparison editor, 99, 101–102
Block Graph style, 496
Blogs, 569–570
Bluetooth instrument, 509
Bookmark navigator, 440
Books, 565–567
Borders for buttons, 229
Branching in version control systems, 102–104
Breakpoint navigator, 150
breakpoint set, 527
Breakpoints, 35
listing, 150
lldb, 524
removing, 36
setting, 36–37
tips, 525
unit testing, 254
working with, 518–521

brokenByLines method, 289
@bug keyword, 448
build action in xcodebuild, 471
Build Configuration popup, 466
Build For Running, 476
Build New Instrument, 512
Build Phases tab
description, 70
libraries, 75
targets, 50, 71–72, 460–461
text data, 135
widget, 294–295
Build Rules tab, 70, 474–475
Build settings, 462–463, 549–550
code signing, 552–553
Compiler, 556–559
DEVELOPER, 561–562
environment, 551–552
Info.plist, 559–560
Java, 560
locations, 553–556
search paths, 561
source trees, 562–563
Build Settings tab
build settings, 462–466, 550
code size, 531
flags, 558
hierarchy, 463–464
packages, 405
product names, 322
Quick Help, 436, 447
release size, 532
SDK, 112
targets, 70, 317
Building views, 161–162
labels, 169–171
outlets and Assistant editors, 163–164
Builds and build system, 459
command-line tools, 471–473
configuration files, 468–471
configurations, 466–467
custom rules, 474–475
dependencies in widget, 294–295
distribution, 314–318
projects, 22–23, 29–30
Report Navigator, 476–477
settings, 462–463, 465–466
settings hierarchy, 463–465
structures, 459–462
transcript, 476–487
tricks, 539–541
BUILT_PRODUCTS_DIR, 554
builtin-copy tool, 483
Bulleted lists, 456
Bumgarner, Bill, 539
Bundle Identifier setting
new projects, 19
OS X applications, 323
Bundles, 279, 401, 403
application, 403–405
Info.plist keys and file, 405–409
location settings, 555–556
.strings files, 392
targets, 543
Button borders, 229

C

.c files, 462
@c comment format, 448
Call-tree detail, 271
Call-tree list, 498

Canvas
segues on, 239
view controllers, 159
Canvas menu, 188
Capabilities editor, 306–308
Capabilities tab, 282–283, 309
Capitalization, 233
Carbon Events instrument, 510
Cascade delete rule, 122
CC, 462
Cell-based views, 348
cellForRowAtIndexPath method
custom cells, 215–217
images, 218–219
outlets, 208–209
prototype cells, 211
table view, 145
Cells. *See* Tables and table cells
Certificates
code signing, 552
Developer ID, 12, 311–312
distribution builds, 314–315
Identifiers & Profiles site, 300
iOS provisioning, 299, 303
private keys, 303
team membership, 300–301
CFBundleIconFiles key, 412
CFBundleVersion, 316
Change color, 452
Check and Install Now, 445
Check for and install updates automatically, 445
Check Out, 89
Check out an existing project, 17, 80, 85, 89
checkNSErrorContent method, 248
Choose a profiling template for, 491
Choose Target, 493
clang compiler, 47
builds, 482
cross-function analysis, 55–56
drawbacks, 57–58
indexing, 56–57
local analysis, 53–55
modules, 65–66
overview, 52–53
precompilation, 64
CLANG_CXX_LANGUAGE_STANDARD, 557
CLANG_ENABLE_MODULES, 557
CLANG_ENABLE_OBJC_ARC, 557
Class Info settings, 440–441
Class Prefix, 111
Classes (educational), 570
Classes (objects)
managed-object. *See* Managed-object classes
name refactoring, 142–143
object allocations by, 489
renaming, 144
specializing, 132–138
clean action for xcodebuild, 471
Cloning repositories, 85
close_enough function, 189–190
Close Project, 24
cocoa-dev list, 568
Cocoa Events instrument, 510
Cocoa language application frameworks
alternatives, 576–577
Core Data, 111
libraries, 76
pointers, 54
Cocoa Layout instrument, 506
Cocoa Touch framework, 107, 180, 286
CocoaHeads meetings, 570
CocoaPods package manager, 574
Code completion, 22, 28, 180–183

Code completion: Automatically insert closing "}", 28
Code-folding ribbon, 535–536
CODE_SIGN_IDENTITY, 314, 552
CODE_SIGNING_ALLOWED, 552
CODE_SIGNING_ENTITLEMENTS, 553
Code Signing Identity, 266
CODE_SIGNING_RESOURCE_RULES_PATH, 553
Code signing settings, 552–553
Code snippets, 181–183
Color
  buttons, 221
  labels, 170, 189
  views, 164
Color controls
  palette, 164
  well, 164
Column Sizing setting, 350
Columns, 172–173
Combined for build settings, 462
Combo fields for property lists, 432
Command Line Developer Tools package, 12
Command Line Tool template, 19
Command-line tools, 11
  builds, 471–473
  package, 14
Comments
  documentation, 447–449
  reStructured Text, 456–457
Commit editor, 84–85
Commit sheet, 91
Commits
  selective, 91–93
  version control systems, 84–85, 98
Company Identifier setting, 323
Comparison editor, 99–100
  Blame view, 101–102
  Log view, 101
Compile Sources build phase, 50, 460–462
Compilers and compiling, 45
  build settings, 556–559
  clang, 52–53
  controllers, 148–149
  cross-function analysis, 55–56
  dynamic loading, 52–53
  indexing, 56–57
  intermediate products, 62–64
  linking, 50–52
  local analysis, 53–55
  precompilation, 64–65
  process, 45–52
  warnings, 29–30, 539
Completes action, 42
Completion, code, 22, 28, 180–183
componentsSeparatedByCharactersInSet method, 254–255
COMPRESS_PNG_FILES, 559
Condition field for breakpoints, 520
Conditionally Sets Editable, 355, 366
Configuration files, 468–471
Configure Repository sheet, 85
configureView method, 178–179
Conflicted file state, 83–84
Conflicts
  assignments, 533
  version control systems, 89–98
Connecting outlets, 163–164
Connection inspector for First Responders, 332–333
Connections for outlets, 177–180
Connections instrument, 508
Console applications, 23
Console windows, 526

Constraints
description, 188
labels, 200–202
planning, 197–200
size, 186–188
trace document window, 499
views, 168, 186–188
Contained extensions, 280
Container apps, 294
Content Compression Resistance, 203
Content Compression Resistance Priority, 204
Contents directory, 403
CONTENTS_FOLDER_PATH, 555
Continue, 39
Continuously Updates Value, 356
Controller Key setting, 354–355
Controllers layers, 141
MVC model, 108, 110
object, 352–354
view. *See* View controllers
Converting
data types, 429
property list formats, 430
Copy Bundle Resources build phase
build rules, 475
folder references, 533
sample-data.csv, 136–137
targets, 74
Xcode structures, 460
Copy for dictionaries, 424
Copy items into destination group's folder (if needed), 326
Copy Source Changes, 100
Copy Transcript for Shown Results, 477
Core Animation instrument, 506
Core Data
events, 490
model objects, 117
Core Data Cache Misses instrument, 505
Core Data Faults instrument, 505
Core Data Fetches instrument, 505
Core Data Saves instrument, 505
Counters instrument, 508
CPU Activity instrument, 509
CPU bar for speed analysis, 267
CPU perspective, 494
CPU Usage style, 496
Create a new Xcode Project, 17, 25
Create Document-Based Application, 323
Create folder references, 128
Create folder references for any added folders, 533
Create Git repository on, 20, 25, 80, 323
Create groups, 128
Create groups for any added folders, 128, 326
CREATE_INFOPLIST_SECTION_IN_LIBRARY, 560
Create New Remote, 85
Create NSManagedObject Subclass, 124
Create Symbolic Breakpoint, 527
Credits.rtf file, 385–388
Cross-function analysis, 55–56
CSResourcesFileMapped key, 410
.csv data files, 246
CSV Reader, 246–251
CSVError class, 253
CSVFileTests class, 247
Current Bytes style, 496
Current Views, 40
Custom build rules, 474–475
Custom instruments, 511–512

Custom script, 475

Custom segues, 239

Custom table cells, 214–217

D

DarwinPorts package manager, 574

Dash styles tool, 572

Dashcode package, 14

Data access in widget, 282–285

Data formatters for numbers, 367–368

Data Model editor, 118

Data Model inspector, 119–122

Data Protection, 308

Data types for property lists, 417–419, 429–431

dataSource property, 207

Date attribute, 119

Date data type
- property lists, 417–419, 429
- Swift, 60

Debug area
- breakpoints, 36
- components, 23, 31–33
- hiding, 23, 40–41
- variables, 37, 254, 525

DEBUG macro, 531

Debug navigator, 31
- actions, 42
- Game table, 211–212
- speed analysis, 266–268
- stack trace, 151–152

Debug Selected Views, 195, 292

Debug XPC services used by this application, 517

Debugging, 515
- breakpoints. *See* Breakpoints
- controllers, 149–154
- dependent targets, 76
- lldb command line, 523–525
- models, 139
- problem fixes, 39–42
- projects, 32–34
- scheme options, 515–518
- stepping through code, 37–39
- tips, 525–528
- tricks, 538–539
- UI Hierarchy, 521
- unit testing, 254–256
- Variables pane, 37–38
- views, 521–523

Debugging-symbol package, 486

Decrease Deck Size, 495

Deepest Stack Libraries style, 496

Default attribute, 120

Default - Property List XML, 427–428

Defaults, sharing, 283–284

Deferred Mode, 502

#define directive, 62

DEFINES_MODULE, 557

Definitions, 535

Delays Events, 380

Delegate design pattern, 145

delegate property, 207

Delete Rule for relationships, 121

Deleting
- menus, 330
- projects, 24

Deny delete rule, 122

Dependencies
- implicit, 76
- makefile goals, 459–460, 462
- widget, 294–295

Dependent targets, 74–76
Deployment Target field, 111-112
@deprecated keyword, 448
DERIVED_FILE_DIR, 475, 554–555
Derived files, 553–554
description method, 526
Descriptions for exceptions, 151
Designable views, 193
Designing widget, 281–282
destination for xcodebuild, 472
Destination locations
- Doxygen, 452
- settings, 553–555

Detail area in trace document window, 495
Detail Disclosure, 232
DetailViewController class, 113
DEVELOPER_APPLICATIONS_DIR, 562
DEVELOPER_BIN_DIR, 562
DEVELOPER_DIR, 562
/Developer directory, 10
DEVELOPER_FRAMEWORKS_DIR, 562
Developer ID, 311–314
Developer ID Application identity, 311
Developer ID Installer identity, 311
DEVELOPER_LIBRARY_DIR, 562
Developer programs, 12–13, 297–299
DEVELOPER_SDK_DIR, 562
Developer Technical Support (DTS), 12, 568–569
DEVELOPER_TOOLS_DIR, 562
DEVELOPER_USR_DIR, 562
Development process in iOS applications, 304
Devices settings, 111, 302–303
Diagnostics tab, 518
Diagrams panel, 452–453
Diamond badges, 244
Dictionaries
- object properties, 233
- property lists, 417–419, 422, 424, 431–432

didReceiveMemoryWarning method, 292–293
didSet method, 247–248, 378
Directives in Swift, 139
Directories for localization, 373–374
Directory I/O instrument, 505
Discard All Changes, 100, 125
Discard Changes, 99–100
Disclosure triangles in trace document window, 495
Disk image (.dmg) files, 13–14
Disk Monitor instrument, 508
Disk space requirements, 10
Dispatch instruments, 505
Display Brightness instrument, 509
Display Pattern field, 364, 366–367, 369
Display requirements, 10
Display Settings tab, 497–499
Distributed source-control systems, 84
Distributions
- builds, 314–318
- iOS applications, 304
- prerelease, 304–306

ditto command, 481
.dmg (disk image) files, 13–14
Do not show this message again, 33
Dock, 17
DOCSET_BUNDLE_ID setting, 453
DOCSET_FEEDNAME setting, 453

DOCSET_ PUBLISHER_ID setting, 454
DOCSET_PUBLISHER_NAME setting, 454
Docsets (documentation sets)
  installing, 455–456
  overview, 444–445
  searching, 442
Document class, 323–324
Document Extension setting, 323
Document outlet sidebar, 176
Document outline view, 159
Document Versions: Allow debugging when using document Versions Browser, 517
Document window, building, 345–350
Document.xcdatamodeld file, 324
Documentation, 435
  docsets, 444–445
  Documentation window, 439–444
  downloading, 11
  Doxygen, 449–456
  help menu, 438–439
  Open Quickly, 437–438
  Quick Help, 435–437, 446–449
  reStructured Text, 456–458
Documentation and API Reference settings, 438
Documentation sets (docsets)
  installing, 455–456
  overview, 444–445
  searching, 442
Documents
  application bundles, 409
  icons, 339–340
  OS X, 321–322
Dollar sign ($) setting, 533
Dot panel in Doxygen, 454
DOT_PATH setting, 454
Double Length Pseudolanguage, 388, 391
Downloading
  docsets, 444–445
  packages, 14–15
  Xcode, 13–14
Downloads panel, 11, 15, 445
Doxygen generator, 446
  basic settings, 450–454
  comments, 449
  docset installation, 455–456
  expert settings, 453
  preparation, 449–450
  running, 454–455
DSTROOT, 554
.dSYM packages, 485
DTPerformanceSession framework, 504
DTrace Data Import, 512
DTrace Script Export, 512
DTrace tool, 511–512
DTS (Developer Technical Support), 12, 568–569
Duck-typing, 58
Dynamic libraries (.dylib), 539
Dynamic loading, 52–53

E

@e keyword, 448
Edges for views, 228
Edit All in Scope, 57
Edit Breakpoint, 520–521
Edit Find Options, 90
Edit for targets, 493
Edit Instrument sheet, 511
Edit ‘Reads/Writes’ Instrument, 511
Edit Scheme, 76, 270, 387

Editing
- build settings, 465–466
- property lists, 419–429
- view controllers, 144–147

Editing tab, 22, 28, 180
Editor area, 32, 162
Editor control, 42
Editor menu, adjusting, 532
Editor Style control, 118
Editor table, 233–235
- passing data to, 235–237
- retrieving data from, 237–239

Editor view controllers, 233–235
Editors, 227
- Assistant. *See* Assistant editor
- Capabilities, 306–308
- Commit, 84–85
- Comparison, 99–102
- Data Model, 118
- linking, 231–232
- Merge, 97
- Project, 375
- Property List, 406, 422–429
- RTF, 385–386
- segues, 240
- static table cells, 232–233
- Target. *See* Targets and Target editor
- text, 571–572
- Version, 99

ellipses (...) for file paths, 5
@em keyword, 448
emacs text editor, 572
Email Link, 444
Embed in Application, 280, 286
Embedded view controllers, 229–231
Enable for Development button, 302
ENABLE_NS_ASSERTIONS macro, 532
Enable user interface debugging, 521
#end directive, 139
Energy Usage instrument, 509
enqueueGame method, 274
Enterprise developer program
- ad-hoc distributions, 304
- Apple developer programs, 298–299
- build settings, 315

Entities, 117
- models, 118
- OS X applications, 326–330

Entitlement chains, 306
Enumerated lists, 456–457
Environment settings, 551–552
Epsilon values, 262
Equality assertions, 262–263
Errors
- compiler, 29–30
- debugging. *See* Debugging
- displaying, 53, 55
- unit testing, 254

Errors Only, 476
Escape key shows code completions, 181
Event Profiler instrument, 508
Events, 490, 508
EXC_BAD_ACCESS message, 32
Exception breakpoints, 150
@exception keyword, 448
Exceptions, 150–154
- assertions, 263–264
- temporary, 309

EXECUTABLE_FOLDER_PATH, 555
EXECUTABLE_PATH, 556
existingPasserByName function, 329–330
Expand Variables Based, 516
expectationWithDescription method, 260
Expert tab in Doxygen, 451–453
Export Accounts, 303

Export as a Mac Application, 539
Export button in Documentation window, 443
Export Developer ID-signed Application, 312
Export for Localization, 396
Export Items, 303
Export Snapshot, 95
Exported UTIs category, 336
expression, 524
expression interpreter, 524
Extended Detail area, 495–496
  Display Settings, 497–499
  Record Settings, 496–497
Extensions.swift file, 129
extern keyword, 535

F

F-keys, 39
Face to face support, 570
Family popup for labels, 170
Fat (universal) binaries, 482
Features, turning off, 22
Fetched Properties table, 118
fetchedResultsController method, 146
Field lists in reStructuredText, 457
File Activity instrument, 505
File Attributes instrument, 505
File Format menu, 336
File inspector tab, 72
File Locks instrument, 505
File Types column, 385
Files
  adding to targets, 71–73
  configuration, 468–471
  renaming, 534
  searching, 535
  sharing, 284–285
  states, 82–83
File's Owner setting, 332, 354
Filesystem instruments, 505–506
Fill With Test Data, 334
Filled Line Graph style, 496
Filling bindings, 350–357
fillViewContents method, 284
fillWithData, 332–336
Filtering stack trace, 151–152
Find and Replace, 89, 425
Find for property lists, 425
Find Implicit Dependencies, 76
Find in Project, 93
Find in Workspace/Project, 425
Find navigator, 144
Finder
  bundles, 403
  docset versions, 445
  instruments, 491
  iOS apps, 405
  packages, 402
Fink package manager, 574
First Responders, 332–333
Fix-it popover, 148
Flatten Recursion, 498
FlushGameQueue method, 274
FlushGameQueues method, 274
Folders
  new projects, 20
  references, 533
Font field for labels, 169–170
Fonts & Colors panel, 32
Fonts for widget, 292
forAllPassersInContext method, 274
Format menu, deleting, 330
Formats tab for strings, 399

Formatters for numbers, 367–368
Forums, 567–568
FOSS (free and open-source) software, 574
Foundation command-line program, 53
Fraction Digits setting, 367
frame in lldb, 524
Frameworks
 benefits, 52
 in compiling, 51
 header files, 64, 74
 libraries in, 76, 285–290
 Objective-C, 114
 overview, 279
 playgrounds, 189
 testing, 243
Frameworks directory, 405
FRAMEWORKS_FOLDER PATH, 556
Frameworks groups in iOS projects, 114
Free and open-source software (FOSS), 574
free function, 489
French localization, 375
 Credits.rtf file, 385–388
 locales, 383–385
 Main.storyboard, 389–392
 process, 382
 resources, 392–394
 strings, 394–397
fulfill method, 261
FULL_PRODUCT_NAME, 551
fullViewContents method, 293–294
Function keys, 39

G

Game Center mediator, 306
Game table, 207
 first run, 211–213
 Model-to-View support, 210
 outlets, 207–208
 protocol methods, 208–209
 prototype cells, 211
 table cells, 214–217
GameDB class, 287–289
GameDetailController class, 376–382
GameListController class, 144
 billboard view, 164, 166, 201
 connections, 177–180, 215
 outlets, 175–177
 tables, 207
 widget, 281
gameTableClicked method, 381
Garbage collection, 558
Gatekeeper, 12, 311–314
gcc compiler, 53
GCC_ENABLE_CPP_RTTI, 462
GCC_ENABLE_OBJC_GC, 559
GCC_OPTIMIZATION_LEVEL, 557
GCC_PREPROCESSOR_DEFINITIONS, 558
GCC_PREPROCESSOR_DEFINITIONS_NOT_USED_IN_PRECOMPS, 559
GCC_TREAT_WARNINGS_AS_ERRORS, 558
GCC_VERSION, 558
GCC_VERSION_IDENTIFIER, 558
GCC_WARN_flags, 559
GCC_WARN_INHIBIT_ALL_WARNINGS, 558
General editor for iOS projects, 111
General settings
 automatic features, 22
 controllers, 148
 icons, 221
 images, 221
 Info.plist file, 405
 instruments, 501–502
 iOS projects, 113
 Issue Navigator detail, 56

libraries, 76
property lists, 419
registering apps, 300
Target editor, 224
GENERATE_DOCSET setting, 453
Generate Test Data build phase, 137
Generic team provisioning profiles, 302
genstrings utility, 395–396
Gestures for navigation, 537–538
Get Started with a playground, 18
Git version-control system, 25, 575
OS X applications, 323
repositories. *See* Repositories
servers, 88
Xcode with, 83–84
GitHub, 575
.gitignore files, 81
Global hot key combinations, 501
GNU General Public License, 53
Goals for makefiles, 459
GPS instrument, 509
GPU Driver instrument, 506
GPU Frame Capture, 518
Graphics, 217
assets catalog, 219–221
icons and launch images, 221–225
image views, 218–219
table cells, 217–218
Graphics instruments, 506
Graphics package, 14
Graphs in Debug navigator, 211–213
GraphViz package, 449
GROUP, 551
Group from Selection, 124, 161
Group popup, 26
Grouped style, 232
Groups, framework, 114

H

Handoff, 307
hAny bar, 200–202
Hardware capabilities, 309
Hardware IO package, 14
hCompact, 202
HEADER_SEARCH_PATHS, 561
HeaderDoc format, 446, 456
Headers, 29
library targets, 74
prefix, 64–65
Heads-up (HUD) window, 176
HealthKit framework, 308
Heights Equally, 203
Hello World project, 19
building and running, 22–23
creating, 19–22
deleting, 24
Help
application bundles, 410
help menu, 438–439
lldb, 523–525
Quick Help, 435–437, 446–449
help breakpoint, 523
help command, 523
Help menu, 438–439
HFS+ filesystem, 544–545
Hidden binding, 364
Hide Missing Symbols, 271, 273, 497–498
Hide/Show Debug Area button, 38
Hide system calls in the stack trace button, 499
Hide System Libraries, 271–272, 498

Hide Toolbar, 526
Hiding Debug area, 41
Highlight setting, 357
HOME, 551
Homebrew package manager, 449, 574
HomeKit framework, 308
Hooking up outlets, 177
Hopper Disassembler tool, 47, 572–573
Hosted extensions, 280
Hosting apps, 294
Hot key combinations, 501
HTML and Doxygen, 453–454
HUD (heads-up) window, 176

I

@i comment format, 448
@IBAction type
  linking controls to actions, 177
  unwind segues, 238
@IBDesignable type, 193
@IBInspectable type, 193
@IBOutlet type
  array controllers, 352
  constraints, 188
  MacStatView, 379
  outlets, 175–177
  removing, 542
  view controllers, 163, 166
iCloud capabilities, 306
Icons
  applications and documents, 339–340
  launch images, 221–225
Identifier popup, 229
#if directive, 139
Ignored file state, 82
Image sets, 219–220
Image Views, 218–219
Images. ***See*** **Graphics**
Images.xcassets catalog, 220
  OS X applications, 324
  overview, 113
Implicit dependencies, 76
#import directive, 62
Import Energy Diagnostics from Device, 509
Import Localizations, 397
In-app purchases, 306
In-house distributions, 304, 315
In Project, 535
In Workspace, 535
#include directive, 62, 470
Increase Deck Size, 495
Indentation tab, 22, 28
Indexed attribute, 120
Indexing, 56–57
Individuals in Apple developer programs, 298
Info.plist file, 324, 431–432
  advertising in, 306
  application keys, 406–409
  background modes, 307
  builds, 315–316, 485
  bundles, 403, 405–406, 409, 415
  gloss effect, 223
  localizations, 406
  OS X applications, 324
  packages, 401–402
  property lists, 420, 422
  settings, 559–560
  strings, 398
Info tab
  application keys, 409–410
  builds, 466–467, 469
  Info.plist file, 405
  localizations, 375
  property lists, 419–420

Quick Help, 436
schemes, 515–516
tests, 245
INFOPLIST_EXPAND_BUILD_SETTINGS, 560
INFOPLIST_FILE, 314, 405, 559
INFOPLIST_OUTPUT_FORMAT, 430, 560
INFOPLIST_PREPROCESS, 560
InfoPlist.strings file, 387, 398, 406
INFOSTRINGS_PATH, 560
Inherited setting, 533
INPUT_FILE_BASE, 475
INPUT_FILE_DIR, 475
INPUT_FILE_NAME, 475
INPUT_FILE_PATH, 475
Input lines, breaking, 5
Input/output instruments, 506
Insert Pattern, 90
insertNewObject method, 147
Inspectable properties, 189, 196
install action for xcodebuild, 471–472
Installing
docsets, 455–456
Xcode, 11
installsrc action, 472
instantiateController method, 345
Instruction, 527
Instruments, 504
behavior, 504
Core Data, 505
custom, 511–512
Dispatch, 505
filesystem, 505–506
graphics, 506
input/output, 506
iOS energy, 509
master tracks, 506
overview, 489–490
recording, 500–503
running, 490–491
saving and reopening, 503–504
speed analysis, 268–272
system, 507–509
templates, 512–513
threads/Locks, 509–510
trace, 509
trace document window, 492–500
tricks, 538
UI automation, 510
user interface, 510
Integer Digits setting, 368
Intentions for views, 168
Inter-App Audio service, 307
Interface Builder
Auto Layout, 186
class names, 143
constraints, 187–188
designable views, 193
game detail, 378–380
inspectable properties, 189, 196
linking views, 166
localizations, 385, 389
outlets, 175, 177, 208
permanent views, 202
property editing, 332
scene editing, 172
size classes, 197
table views, 349
view controllers, 158–161, 163, 238
widget, 291
Interface Builder tab, 42, 160
Intermediate compiler products, 62–64
Interpreted languages, 53
Invert Call Tree, 271–272, 497–498
I/O Activity instrument, 506

iOS
- application bundles, 406–409, 412–415
- Auto Layout. *See* Auto Layout
- capabilities, 306–308
- energy instruments, 509
- measurement and analysis. *See* Measurement and analysis model. *See* Models
- MVC design pattern, 107–110
- as packages, 405
- porting from, 326–330
- prerelease distributions, 304
- provisioning. *See* Provisioning
- scheme options, 517–518
- starting projects, 110–112
- table cells. *See* Tables and table cells
- templates, 112–114
- unit testing. *See* Unit testing
- view controllers. *See* View controllers
- widget extension. See Widget

"iOS Debugging Magic (TN2239)", 538

iOS Enterprise developer program
- Apple developer programs, 298
- build settings, 316

iOS icon is pre-rendered, 223

iOS Simulator
- limitations, 213, 266
- memory, 507
- speed analysis, 266
- templates, 513
- tests, 252

IPHONEOS_DEPLOYMENT_TARGET, 561

iprofiler, 504

ISO-standard languages, 373

Issue navigator, 29

Issue Navigator Detail, 56

Issues: Show live issues, 53

Items of New Constraints, 175

J

Java, Build settings for, 560

JetBrains AppCode, 576

Join a Program, 298

JSON format for property lists, 430–431

Jump bars
- Assistant editor, 164–165
- description, 160–161

K

Kaleidoscope tool, 573

Keep in Dock, 491

Key Bindings panel
- controllers, 148
- key equivalents, 227–228
- Preferences window, 533–534

Key Equivalent field, 331

Key-Value Coding (KVC), 180, 233–234

Key-Value Observation (KVO), 351, 353, 361

Key-value pairs
- localizations, 389
- property lists, 422

Keyboard panel, 39, 412, 501

Keyboard Shortcuts tab, 412

Keychain sharing, 306

keyPathsForValuesAffectingCurrentTeamName method, 362

Keys for applications, 406–415

KVC (Key-Value Coding), 180, 233–234

KVO (Key-Value Observation), 351, 353, 361

L

Labels
  building views, 169–171
  constraints, 200–202
  tags, 215
Language & Region panel, 373
Language & Text panel, 399
Language popup, 111
Language setting, 323
Language tab, 373
Launch behavior for bundles, 410–411
Launch due to a background fetch event, 518
Launch images, 221–225
Launch storyboards, 224
LaunchScreen.xib layout, 113
Layout
  Auto Layout. See Auto Layout
  localizations, 375–378
Layout guides for views, 168
layoutSubviews function, 194–195
Leading edges of views, 228
LeagueDocument class, 333–334
LeagueViewController class, 344, 347, 358–360, 363
LeagueWindowController class, 345–347
Leaks instrument, 507, 538
Left-side group for labels, 169–171
Levels for build settings, 464–465
libcrypto API, 9
Libraries
  adding, 75–76
  dynamic, 52–53
  instruments, 499–500, 504
  object, 51, 347–348
  static, 69, 539
  targets. *See* Library targets
/Library/Developer directory, 11–12
Library navigator, 444
Library palette, 504
LIBRARY_SEARCH_PATHS, 561
Library targets, 69
  adding, 69–71
  debugging, 76
  dependent, 74–76
  description, 70–71
  headers, 74
  membership, 71–73
Library window, 499–500
limitPinner function, 129
Line Graph style, 496
Line wrapping: Wrap lines to editor width, 22
Link Binary With Libraries build phase, 50–51, 66, 75, 460
Linking and linkers
  editing, 51
  editors, 231–232
  process, 50–52
  tricks, 540
  views, 166–167
Lion, 308
lipo tool, 482
Lists in reStructured Text, 456–457
Live Rendering, 193
lldb debugger
  command line, 523–525
  linking description, 53
***LLDB Quick Start Guide*, 525**
.lldbinit files, 525
llvm library, 53, 149
loadGames function, 272–274, 334–335
Loading
  document window, 345–347
  dynamic, 52–53
  LeagueDocument data, 333–334
loadSampleData method, 287–288

loadStatViews method, 380
loadView method, 158
Local analysis, 53–55
Local remote repositories, 87–88
Local variables, 37
Locales, 383–385
Localizable Strings file, 385
Localizable.strings file, 395–396
Localization Debugging: Show non-localized strings, 517
Localizations, 368, 373
  adding, 374–376
  application bundles, 408–409
  base, 374–376
  French. See French localization
  game detail, 376–382
  Info.plist, 406
  locales, 383–385
  modules and namespaces, 382
  overview, 373–374
  strings, 398–399
Locations
  Doxygen, 452
  settings, 553–556
Locks instruments, 509–510
Log Message, 520
Log view for Comparison editor, 99, 101
Logic tests, 260
Login button.png, 404
Logs
  vs. breakpoints, 518–519
  builds, 476–477
lproj system, 373–374
LSApplicationCategoryType key, 410
LSBackgroundOnly key, 410
LSEnvironment key, 410
LSFileQuarantineEnabled key, 411
LSFileQuarantineExcludedPathPatterns key, 411
LSGetAppDiedEvents key, 410
LSMinimumSystemVersion key, 411
LSMinimumSystemVersionByArchitecture key, 411
LSMultipleInstancesProhibited key, 411
LSRequiresIPhoneOS key, 412
LSUIElement key, 411
LSUIPresentationMode key, 411

M

Mac App Store. *See* App Store
Mac Developer identity, 312
Mac OS X. *See* OS X
"Mac OS X Debugging Magic (TN2124)", 538
MAC_OS_X_PRODUCT_BUILD_VERSION, 552
MAC_OS_X_VERSION_ACTUAL, 551
MAC_OS_X_VERSION_MAJOR, 552
MAC_OS_X_VERSION_MINOR, 552
Machine instructions, 49
MacOS directory, 404
MacPorts package manager, 574
MacStatView class, 376
Mailing lists, 568
Main.storyboard file, 112, 389–392
MainInterface.storyboard file, 290
Makefile goals, 459
makeWindowControllers method, 344–345
malloc function, 489
MallocDebug application, 489
Manage Flags, 508
Manage PM Events, 508
Manage Schemes editor, 317

Managed-object classes, 117, 123
creating, 124–128
source control and product files, 136–138
specializing, 132–138
test data, 134–136
Managed object contexts, binding, 352–353
managedObjectContext property, 353
Maps capability, 306–307
Margins in reStructured Text, 456–458
MARK directive, 208
Mark Heap, 538
Mark Selected Files As Resolved, 84
Master branches in version control systems, 102–103
Master-Detail Application template, 161–162
Master tracks instruments, 506
MasterViewController class, 112, 141–143
Matching, 144
Mavericks version, 9–11
Maximum attribute, 120, 368
MDM (Mobile-Device Management) systems, 304
measureBlock method, 246, 276
Measurement and analysis, 264
memory, 277–278
speed. *See* Speed
Meetings, 570
Membership, target, 71–73
Memory, 277–278
instruments, 506–507
problems, 538
RAM, 49
requirements, 10
Memory Monitor instrument, 508
Menus, wiring, 330–331
file types, 336–338
First Responders, 332–333
icons, 339–340
League Files, 338–339
LeagueDocument data, 333–334
managed documents, 334–335
targets and actions, 331–332
testing commands, 335–336
Merge editor, 97
Merge from Branch, 104
Merge into Branch, 104
Merges in version control systems, 89–98
Messages
analysis, 56
Documentation window, 444
logs, 520
Objective-C compilers, 544
Metadata in Git, 82
Method names, refactoring, 142
Min Length setting, 120
Mini instruments, 502
Minimum attribute, 120, 368
missing-braces-and-parentheses warning, 149
MKDirectionsApplicationSupportedModes key, 415
Mobile-Device Management (MDM) systems, 304
Modal scenes, 227–233
Mode settings
Doxygen, 452
object controllers, 352
Model Key Path field, 354, 366–367
Model-to-View support, 210
Model-View-Controller (MVC) design pattern, 107–108
controllers, 110
models, 108
views, 108–110

Models
- attributes, 118–121
- debugging, 139
- entities, 118
- implementing, 117
- managed-object classes. *See* Managed-object classes
- OS X applications, 325–330
- relationships, 120–123

Modified file state, 82–83
module.map file, 66
MODULE_NAME, 557
Modules, 65–66
- localizations, 382
- Swift in, 289
- symbols, 250

Modules extension, 52
mogenerator tool, 126–128, 573
More Developer Tools, 14
Mouse buttons, 5
Mouse pointer variables, 37
Move Breakpoint To, 525
Multiple Values Placeholder, 356
MVC (Model-View-Controller) design pattern, 107–108
- controllers, 110
- models, 108
- views, 108–110

N

Name labels, 169–170
Names
- product, 322
- refactoring, 142–143

Namespaces, localizations, 382
nan (not a number), 35
NATIVE_ARCH, 557
NATIVE_ARCH_32_BIT, 557
NATIVE_ARCH_64_BIT, 557
Navigation panel for gestures, 537–538
Navigators, 21
- Breakpoint, 150
- Debug, 211–212
- detail settings, 56
- Documentation window, 439–440
- Issue navigator, 29
- Library, 444
- Project, 36
- Report, 476
- Symbol, 57

NDEBUG macro, 531
Net resources, 567–570
Network Activity instrument, 509
Network Activity Monitor instrument, 508
Network capabilities, 309
New Branch, 102
New File assistant, 26–27
New Folder
- Doxygen, 452
- subclasses, 124

New OS X Icon, 339–340
New Project assistant, 19–20, 27
- iOS, 110
- OS X, 328

New Scope, 535
New Target assistant, 69, 280
New Trace assistant, 491
newton_sqrt function, 189
Next Run, 494
.nib files, 143
nm tool, 63
No Access, 309
No Action delete rule, 121
No Selection Placeholder field, 356

Normalizing entities, 326
Not a number (nan), 35
not-enough-fields.csv file, 252
Notification Center widget, 290, 294–295
NS_BLOCK_ASSERTIONS macro, 531–532
NSAppleScriptEnabled key, 411
NSApplicationShowExceptions setting, 528
NSArray class, 151
NSArrayController class, 352–353
NSBundle class, 373, 392
NSCoder Night meetings, 570
NSController class, 352
NSError class, 54–55
NSFetchedResultsController class, 113, 129–130, 144, 207
NSFileWrapper class, 402
NSHumanReadableCopyright key, 408, 410
NSLocalizedString class, 395
NSJSONSerialization class, 431
NSLog function, 519–520
NSMainNibFile key, 407
NSManagedObject class, 145
  MVC model, 107
  subclass creation, 124
  widget design, 281
NSManagedObjectCollector class, 353
NSManagedObjectContext class, 289
NSNull class, 431
NSObject class, 107
NSPersistentDocument class, 334, 353
NSPrincipalClass key, 407
NSRTFDPboardType file type, 402
NSScrollView, 544
NSServices key, 412
NSSplitView class, 357
NSSplitViewController class, 358
NSString class, 180
NSSupportsSuddenTermination key, 411
NSTextFields, 370
NSTextView, 544
NSZombieEnabled setting, 518
Null Placeholder, 356
Nullify delete rule, 122
Numbers
  data formatters, 367–368
  property lists, 417–418, 429

O

.o files, 460
-O3 optimization, 540
objc-language list, 568
Object allocations by class, 489
Object controllers, 352–354
OBJECT_FILE_DIR, 555
Object files, 50–51
objectAtIndexPath method, 144
Objective-C programs
  assertions, 261–262
  characteristics, 57–58
  choosing, 111
  compiler messages, 544
  data types, 417–418
  namespaces, 382
  optimization settings, 540
  optional arguments, 395
  renaming symbols in, 141–143
  shared libraries, 286
  source files, 50
  support for, 62
OBJROOT, 554
observeValueForKeyPath method, 360–361
-Ofast optimization, 540
-Onone optimization, 540

Open another project, 18
Open GL ES, 518
Open in instruments, 504
Open Keyboard Shortcut Preferences, 501
Open Quickly, 261, 437–438, 532
Open Recent, 24
OpenCL facility, 53
OpenGL ES Analyzer instrument, 506
Optimization
  compiler, 48–49
  speed, 272–275
  tricks, 540–541
Option key, 537–538
Optional for libraries, 75
Options panel, 19–20
Options tab
  localizations, 388
  schemes, 516
Ordered for relationships, 121
Ordered lists, 417–418
Organization Identifier, 110, 280
Organization Name setting
  iOS projects, 110
  new projects, 19
  OS X projects, 323
Organizational Identifier setting, 19
Organizations in Apple developer programs, 298
Organizer window
  derived files, 554
  snapshots, 95
  trash, 24
Orientation constraints, 197–198
-Os optimization, 540
OS X, 321
  application keys, 409–412
  bindings. *See* Bindings
  bundles. *See* Bundles
  capabilities, 306–307
  entities, 326–330
  frameworks. *See* Frameworks
  goals, 321–322
  localizations. *See* Localizations
  models, 325–330
  porting from iOS, 326–330
  property lists. *See* Property lists
  sandboxing, 308–311
  starting applications, 322–325
  storyboard segues, 343–345
  wiring menus, 330–340
OSAScriptingDefinition key, 412
OTA (over-the-air) installations, 304
OTHER_CFLAGS, 558
OTHER_CODE_SIGN_FLAGS, 553
OTHER_CPLUSPLUSFLAGS, 558
OTHER_SWIFT_FLAGS, 558
otool tool, 63
outlet collections, 201
Outlets
  building views, 163–164
  code completion and snippets, 180–183
  connections, 177–180
  hooking up, 177
  overview, 175–176
  table view, 207–208
Output panel in Doxygen, 452
Over-the-air (OTA) installations, 304
Overlay for instruments, 497

P

@p code-text comment, 448
Package managers, 574

Packages, 401
downloading, 14–15
RTFD, 401–402
PaintCode tool, 573–574
@param parameter comment, 448
Passbook system, 307
Passer Array controller, 363–364
Passer class, 212, 231–232
passer_rating function, 131, 256
Passer ratings project overview
building, 29–30
controllers, 154–155
creating, 25–29
debugging, 32–34
running, 30–32
test case, 35
PasserEditController class, 228, 233–235
PasserEditTableController class, 230, 233
passerGameHeader method, 394
PasserListController class, 160, 162, 231, 233, 236
Passing data to editor, 235–237
Paste for dictionaries, 424
.pch files, 65
Peak Graph style, 496–497
Performance
compiler, 48–49
speed, 272–275
tricks, 540–541
XCTest, 276–277
Performance bar charts, 31
Persistent State: Launch application without state restoration, 517
Personal VPN system, 307
Phases, build, 460–461
PhoneGap framework, 576–577
pinComparables function, 129
pip package, 446
Pixels for icons, 223
Plain style, 232
Planning
apps, 107–110
constraints, 197–200
platform in lldb, 523
PLATFORM_NAME, 551
Playback head in trace document window, 494
Player billboard, 186–188
Playgrounds, 17, 189–191
PLIST_FILE_OUTPUT_FORMAT, 430
Plists. *See* Property lists
plutil tool, 428–430
po command, 526
Point Graph style, 496
Pointers in Cocoa programming, 54
Points for icons, 223
Popovers
Quick Help, 436–437
variable values, 37
Portals for iOS, 304
Porting from iOS, 326–330
POSIX working directory, 517
Precompilation, 64–65
Prefer Margin Relative, 173–174
Preferences window
Apple ID, 298
automatic features, 22
behaviors, 40–42
bindings, 148
code completion, 28, 180
code-folding ribbon, 535–536
controllers, 148
docsets, 444–445
downloads, 11, 15

Preferences window (*continued*)
fonts, 32
indentation, 28
instruments, 501
Issue Navigator detail, 56
key equivalents, 533
navigational gestures, 537–538
remote repositories, 85, 88–89
source trees, 563
team membership, 300–301
version control, 80
warnings and errors, 53
Prefix files, 64–65
Prefix headers, 64–65
prepareForSegue function, 234–236, 283–284, 381
Prepares Content, 352
Preprocessing xcconfig files, 470–471
Preprocessors, 62–63
Prerelease distributions, 304–306
Prerelease versions, 13
Present As Popover connections, 239
Preview, 94–95
Preview assistant, 168, 172
Preview view, 165–166
Previous Run, 494
print_ln function, 191
Private keys for certificates, 303
Private role, 74
Probes, 511
process in lldb, 524
Process instrument, 508
Processor requirements, 10
Product files in managed-object classes, 136–138
PRODUCT_MODULE_NAME, 557
PRODUCT_NAME, 551
Product Name setting, 19, 322
PRODUCT_TYPE, 551
Profile action, 490–491
Profiles
applications, 268
provisioning, 299–303
Program members, 302
PROJECT, 551
PROJECT_DIR, 553
Project editor
library targets, 69–70
localizations, 375
PROJECT_FILE_PATH, 553
-project for xcodebuild, 472
PROJECT_NAME, 551
Project navigator, 36
Project role, 74
ProjectBuilder, 1
Projects list for builds, 460
Projects overview
building, 22–23, 29–30
creating, 19–22, 25–29
debugging, 32–34
deleting, 24
Doxygen settings, 452–453
running, 22–23, 30–32
templates, 112–114
Projects panel
derived files, 554
snapshots, 95
Properties for entities, 117
Property List editor, 406
limitations, 427–429
working with, 422–426
Property lists, 417
binary, 430
data types, 417–419, 429–431
editing, 419–429

limitations, 431
localizations, 392–394
specialized, 431–432
text, 429–430
Protecting assets, 303
Protocol methods, 208–209
Prototype cells, 211, 214
Provide Feedback link, 445
Provisioning, 297, 299
asset protection, 303
capabilities editor, 306–308
distribution builds, 314–318
Gatekeeper and Developer ID, 311–314
OS X Sandboxing, 308–311
prerelease distributions, 304–306
profiles, 299–303
registering apps, 300-303
PROVISIONING_PROFILE, 314, 553
Public role, 74
Pull, 96
Push, 88–89, 96
Push segues, 162
Pushing to remote repositories, 88–89
pwrite function, 511

Q

Quick Help facility, 435
comment syntax, 447–449
generating, 446–447
inspector, 435–436
popovers, 436–437
Quick Help for Selected Item, 436, 439
QuickLook eye, 191
Quit
lldb, 523
OS X, 33
Quit Xcode, 24

R

Raises For Not Applicable Keys, 354–356
RAM, 49
rating_components function, 256–257
RatingTest class, 256–260
Read Access, 309
Read/Write Access, 309
Reads/Writes instrument, 506, 511–512
Recent Executables, 493
Recent for builds, 476
Record button, 501
Record for instruments, 504
Record Options, 502–503
Record Settings tab, 496–497
Recording instruments, 500–503
Rectangles, layout, 171
Refactoring feature, 57
class names, 142–143
method names, 142
Reference Language column, 384–385
References
folders, 533
repositories, 87
Registered developers, 12–13, 302
Registering
app groups, 282–283
apps, 300–303
team membership, 300
Regular expressions
refactoring method names, 142
searches, 89
traps, 542

Relationships, 117–118, 120–123
Relative to Group, 534
Release Notes section, 438
Remote repositories, 84–87
Remotes tab, 88
Removing
 breakpoints, 36
 Xcode, 11–12
Renamed file state, 83
Renaming
 classes, 144
 symbols, 141–143
Renaming service, 534
Render As Template Image, 221
Rentzsch, Jon “Wolf”, 126
Reopening instruments, 503–504
Replace All, 144
Replace All in File, 90–91
Report Navigator, 476–477
Repositories
 cloning, 85
 remote, 84–87
 Xcode Server, 85–86
Required for libraries, 75
Requirements, 9–10
resizableImageWithCapInsets method, 221
Resolve Auto Layout Issues menu, 202–203, 229
Resource forks, 401
Resource Manager, 401
Resources
 books, 565–567
 Developer Technical Support, 568–569
 face to face, 570
 localizing, 392–394
 net, 567–570
 sites and blogs, 569–570
 software, 570–576
Resources directory, 403
Respect language direction, 388
Responder chains, 331
reStructured Text (reST) language, 446, 456–458
restview package, 446
Retain cycles, 277
Retrieving data from editor, 237–239
@return keyword, 448
Return Value, 525
Reveal in Library, 440
Rich text file directory (RTFD) package, 401–402
Right-clicking, 5
Right-side group for labels, 169–170
Right to Left Pseudolanguage, 388, 391
Root view controller segues, 162, 239
Routing App Coverage File, 518
Row Height setting for table cells, 215
RTF editor for localizations, 385–386
RTFD (rich text file directory) package, 401–402
Rules, build, 474–475
Run-Edit-Print Loop (REPL), 189
Run scheme editor, 515
Run Script editor, 135
Running
 Doxygen, 454–455
 instruments, 490–491
 projects, 22–23, 30–32
 tests, 252–254
Running Application settings, 493

S

sample code, 5
sample-data.csv file, 136–138, 252, 265, 333
Sampler instrument, 508
Sandboxing
  benefits, 310
  disadvantages, 310–311
  OS X, 308–311
Save as Template, 503
Save-file dialog for targets, 73
Save for iOS App Store Deployment, 539
Save Screen Shot, 224
Saving instruments, 503–504
Scale for track area, 495
Scan recursively, 452
scanf function, 32, 47–48
Scenes
  modal, 227–233
  view controllers, 158–161
Schedules for instruments, 509
Scheme control, 74
Scheme editor, 76
  instrument templates, 270
  tests, 245
scheme for xcodebuild, 472
Schemes
  builds, 317
  options, 515–518
Scopes, defining, 535
SDKROOT, 553
SDKs (software development kits), 9
  build settings, 469–470
  iOS projects, 112
Search Documentation for Selected Text section, 439
Search paths for settings, 561
Searches
  Documentation window, 440–443
  files, 535
  help, 438
  version control, 90–91
@see documentation comment, 448
Segues
  passer list, 231
  types, 239–240
  unwind, 238
  view controllers, 158–161
  views, 162
Select Statistics to List settings, 497
Selection Indexes setting, 366
Selective commits, 91–93
sender method, 232, 235
Separate by Category, 497
Separate by Thread, 497
Services menu, 501, 542
Set Baseline, 276
  setUp method
  CSV testing, 248
  performance testing, 276
  unit testing, 257
setupPlayers function, 194
Shadow Offset, 170
Shadows for labels, 170
SHALLOW_BUNDLE, 555
Share Breakpoint, 525
Shared libraries in frameworks, 285–290
Shared Memory instrument, 507
Shared User Defaults Controller, 354
sharedGameDB method, 287
Sharing
  defaults, 283–284
  files, 284–285
Shift key, 538

Shortcuts
function keys, 39
instruments, 501
lldb, 524
Show All Results, 441
Show Bounds/Layout Rectangles, 188
Show Bounds Rectangles, 164
Show: Code folding ribbon, 22, 535
Show Definitions, 466, 550
Show Detail connections, 239
Show environment settings in build log, 549
Show Find Options, 542
Show Group Banners, 500
Show/Hide...debugger, 40
Show/Hide navigator, 40
Show HTML output, 454
Show In Finder, 24, 539
Show live issues, 22, 148
Show navigator, 42
Show non-localized strings, 392
Show Package Contents, 10, 402, 405, 445
Show Raw Values & Keys, 432
Show Setting Names, 466, 550
Show Setting Titles, 550
Show Slicing, 221
Show tab named, 42
Show this window when Xcode launches, 18
Show Values, 550
Signals from exceptions, 150
Signatures in iOS provisioning, 299
Signing identities, 299, 487
SimpleCSVFile, 246
Simulate Document, 350
Simulate Location, 39
Sites, 569–570
Size and Size Inspector, 168, 173
columns, 350
constraints, 186–188, 197–200
table cells, 215
views, 164–165
Size classes, 197
SKIP_INSTALL, 555
Sleep/Wake instrument, 509
Snap Track to Fit, 271, 495
Snapshot Now, 507
Snapshots
projects, 95
VM Tracker, 507
Snippets, 181–183
Software development kits (SDKs), 9
build settings, 469–470
iOS projects, 112
Software resources, 570–571
AppCode, 576
assessment, 577–578
Cocoa alternatives, 576–577
helpers, 572–574
package managers, 574
text editors, 571–572
version control, 575–576
Sort Descriptors binding, 365–366
Sorting tables, 365
Source code
description, 45
Doxygen, 452
property lists, 427
Source control. *See* Version control systems
Source Control menu, 81–82, 86, 88–89
Source files with names matching, 475
Source Locations settings, 553
Source trees settings, 562–563
Sources & Binaries, 449

SourceTree version control system, 575
Specialized property lists, 431–432
Speed, 265–266
  Debug navigator, 266–268
  instruments, 268–272
  memory, 277–278
  optimization, 272–275
Spin Monitor instrument, 508
Splash screens, 224
Split views, 357
Spotlight box, 120
SQLite, 117, 336
SRCROOT, 135, 472, 553
Stack Libraries style, 496
Stack Overflow forum, 568
Stack traces, 31
  displaying, 151
  filtering, 151–152
  trace document window, 499
Stacked for instruments, 497
Staged file state, 83
Starting
  iOS projects, 110–112
  Xcode, 17–18
startMeasuring method, 276
States of files, 82–83
Static libraries (.a), 69, 539
Static table cells, 228, 232–233
Statistics to Graph settings, 512
Stats view, 291–292
StatView, 191–197
Step Into (F7), 39, 527–528
Step Out (F8), 39
Step Over (F6), 39, 527
Stepping through code, 37–39, 527–528
Stop
  debugging, 33
  instruments, 501
  iOS, 155
stopMeasuring method, 276
Store in External Record File, 120
Storyboard editor, 240
.storyboardc files, 143
Storyboards, 224
  segues, 240, 343–345
  for view controllers, 158–161
Strategy control, 494
Strings and .strings files
  localizations, 389–392, 394–399
  property lists, 417–418, 429–430
STRINGS_FILE_OUTPUT_ENCODING, 560
Structure
  application bundles, 407, 410, 412–413
  builds, 459–462
Structured directory trees, 403
Style settings
  buttons, 229
  instruments, 496
  models, 118
  table cells, 232
Sublime Text 2 text editor, 571–572
Subscripts, 129–130
Subviews from layouts, 188–197
Sudden Termination instrument, 504
Suggest completions while typing, 180
Supporting Files group, 113
Suppressing warnings, 155
SWIFT_OPTIMIZATION_LEVEL, 557–558
Swift programming language
  assertions, 261–263
  books about, 566–567
  chained expressions, 139

Swift programming language (*continued*)
choosing, 111
class names, 123
class qualifiers, 361
collections, 177
data types, 418
directives, 139
exceptions, 263
HeaderDoc comments, 446
managed-object classes, 127
mangled names, 150
memory management, 277
modules, 65–66, 114, 250, 289
namespaces, 382, 543
Objective-C comparisons, 57–62
optimization, 540
pinner functions, 131
playgrounds, 18, 189
renaming classes, 144
reStructured Text, 456–458
subscripts, 129
support limitations, 63, 126
system libraries, 66
Switch-Branch sheet, 103
Switch to Branch, 103
Symbol navigator, 57
Symbols, 49
modules, 250
renaming, 141–143
tokens, 53–54
SYMROOT, 554
Syntax-aware indenting settings, 22, 28
System Calls instrument, 509
System instruments, 507–509
System Language, 388
SYSTEM_LIBRARY_DIR, 561
System Preferences application
function keys, 39
gestures, 29–30, 440
instruments, 501, 506
localizations, 373, 387
System Processes, 493

T

Tab, 42
Table Cell View, 355
Table of contents sidebar, 440
Table View Cell, 355
Tables and table cells, 207
custom, 214–217
graphics. *See* Graphics
modal scenes, 228
OS X, 347–350
outlets, 207–208
prototype, 211
static, 228, 232–233
table views, 145–146, 174–175, 229–230
tableView function, 207, 209, 216
Tabs
creating, 159
Documentation window, 440
switching, 42
Tags
labels, 215
version control, 542
TARGET_BUILD_DIR, 554
Targets and Target editor
ad-hoc variants, 317
asset-catalog file, 221
build phases, 50, 460–461
build rules, 474

build settings, 462–463
bundles, 543
capabilities, 309
code size, 531
components, 69–70
configuration files, 467–469
dependencies, 76
device families, 414
displaying, 244
icons, 221, 316
images, 221, 414
Info.plist file, 324, 405–406, 408
instruments, 501
iOS projects, 111–112
levels, 464–465
libraries. *See* Library targets
in lldb, 523–524
new projects, 26–27
packages, 405–406
product names, 322
property lists, 419–420
provisioning profiles, 300
Quick Help, 436, 447
registering apps, 300
storyboards, 224
trace document window, 492–493
widget, 280
wiring menus, 331–332
xcodebuild, 471–472

Team Admins in Apple developer programs, 298

Team Agents in Apple developer programs, 298

Team array controller, 352–353

Team class, 327

Team Members in Apple developer programs, 298, 300

Team Provisioning Profiles, 302

TeamDetailController, 358–363

teamGameHeader method, 394–395

teamWithName function, 328

tearDown method, 246, 248, 276

Templates
instruments, 270, 503, 512–513
iOS projects, 112–114

Temporary exceptions, 309

Terminal application, 24

test action for xcodebuild, 471

Test data for unit testing, 252

Test navigator, 244–246

Test suites, 243

testCalculation method, 257

testExample method, 246

testFileReadsCompletely method, 249

TestFlight beta distributions, 305–306

Testing
unit. *See* Unit testing
views, 183–184

testNoSuchFile method, 249

testPerformanceExample method, 246, 276

testTooManyFieldsError method, 253–254

Text Color control, 170

Text Editing panel, 22, 180, 536

Text editors, 571–572

Text for property lists, 417–418, 429–430

TextEdit application, 401–402

TextMate 2 text editor, 571

TextWrangler text editor, 571

3rd Party Mac Developer Application identity, 312

3rd Party Mac Developer Installer identity, 312

Third-party package managers, 575

Thread
in debugging, 527
lldb, 524
Threads instruments, 509–510
Threads perspective, 494
Time Profile commands, 501–502
Time Profilers for instruments, 268–272, 496, 508
Timeline Assistant, 191
Titanium API, 577
Titles
buttons, 228–229
columns, 350
menu items, 331
Today widget, 290–294
TodayViewController, 291–293
@todo documentation comment, 448
Toggle Instruments Recording, 501
Tokens, 53–54
too-many-fields.csv file, 252
Toolbars
modal scenes, 228
trace document window, 492–494
Tools in Interface Builder, 159–161
Top Functions filter, 498
Top Layout Guide, 228
Trace document window, 492
Detail area, 495
Extended Detail area, 495–499
Library area, 499–500
toolbar, 492–494
Track area, 494–495
Trace documents, 268–269
Trace Highlights, 499
Trace instruments, 509
Track area in trace document window, 494–495
Track Display, 497
Trailing edges in views, 228
Transcripts for builds, 476–487
Transient attribute, 120
Trash, 24
Tricks
Assistant editor, 536–538
building, 539–541
code-folding ribbon, 535–536
general, 531–535
instruments and debugging, 538–539
Truncation, 367
Two developer-program memberships, 315
2010-data-calculated.csv file, 252
Type menu for instruments, 497
Typographic conventions, 5

U

UI automation instruments, 510
UI Hierarchy view, 521
UI-layout editors, 158
UI panel, 521
UIAppFonts key, 413
UIApplication class, 260
UIApplicationDelegate protocol, 112
UIApplicationExitsOnSuspend key, 414
UIBackgroundModes key, 414
@UIDesignable views, 292
UIFileSharingEnabled key, 414
UIImage class, 221
UIImageView class, 218
UIInterfaceOrientation key, 413
UILabel class, 171
UILaunchImageFile key, 414
UILaunchImages key, 414
UILaunchStoryboardName key, 412

UIMainStoryboardFile key, 412
UINavigationController class, 161–162
UINewsstandApp key, 414–415
UIPrerenderedIcon key, 414
UIRequiredDeviceCapabilities key, 413
UIRequiresPersistentWiFi key, 413
UIScrollView class, 347
UIStatusBarHidden key, 413
UIStatusBarStyle key, 413
UISupportedExternalAccessoryProtocols key, 413
UISupportedInterfaceOrientations key, 413
UITableView class, 129–130, 145, 215, 229, 347
UITableViewCell class, 145, 211, 214, 217, 231
UITableViewController class, 113, 229–230
UITableViewDataSource class, 208
UITableViewDelegate class, 236
UIView class, 109–110, 166, 215
UIViewController class, 110, 141, 157–158, 175, 207, 229–230
UIViewEdgeAntialiasing key, 414
UIViewGroupOpacity key, 414
Umbrella headers, 66
Undefined attributes, 119
Unformatted field, 368
Uniform Type Identifiers (UTIs), 336–338
Unit testing
  asynchronous tests, 260–261
  CSV Reader, 246–251
  overview, 243–244
  test navigator, 244–246
  testing and debugger, 254–260
  XCTest assertions, 261–264
Universal (fat) binaries, 482
Unknown file state, 84
UNLOCALIZED_RESOURCES_FOLDER_PATH, 556
Unmerged file state, 83
Unmodified file state, 83
Unresolved addresses, back-filling, 51
Untracked file state, 82
Unwind segues, 238
Update Frames, 171, 173, 203
Update Frames menu, 175
URLs for application bundles, 409
Use Base Internationalization, 375
Use Core Data
  iOS projects, 111
  OS X applications, 323
Use dot tool, 452
Use scalar properties for primitive data types, 124
Use Storyboards, 323
USER, 551
User and System Libraries style, 496
User Info settings for models, 120
User information for application bundles, 407–408
User interface instruments, 506, 510
User presentation in application bundles, 413–414
Uses Lazy Fetching, 352
Using popup for build rules, 475
/usr/bin directories, 11
UTExportedTypeDeclarations key, 409, 432
Utilities.swift file, 129–131
Utility area, 72, 159
UTImportedTypeDeclarations key, 409

V

Validates Immediately, 356
Validation field for attributes, 120

Value Transformer field, 354, 364

Value With Pattern binding, 364

valueForKeyPath method, 134

Variables

build. *See* Build settings

Debug area, 32

Variables pane, 37–38, 526

verbal conventions, 5

Version control systems, 25, 79–80

branching, 102–104

commits, 84–85

file state, 82–83

managed-object classes, 136–138

merges and conflicts, 89–98

remote repositories, 84–87

software, 575–576

tags, 542

Version editor, 99–102

working with, 81–82

Xcode with Git, 83–84

***Version Control with Subversion*, 575**

Version editor, 99–102

Versioned bundles, 403

Versioning, 120

Versions covered, 4

Versions version control system, 575–576

vi text editor, 572

View control, 31, 42, 119

View controllers, 141

adding, 157–158

building views. *See* Building views

editing, 144–147

embedded, 229–231

outlets. *See* Outlets

storyboards, scenes, and segues, 158–161

table views, 174–175

View Debugging, 517

View Details, 301

View menu, deleting, 330

View Processes by Queue, 521

View Processes by Thread, 521

View selector, 23

View UI Hierarchy, 521

viewDidLoad method, 231, 233–235, 292, 380, 393–394

Views

Auto Layout. *See* Auto Layout

building. *See* Building views

cleaning up, 171–174

constraints, 168, 186–188

debugging, 521–523

linking, 166–167

MVC model, 107–110

table, 145–146, 174–175, 229–230

testing, 183–184

viewWithTag function, 215

VM Operations instrument, 509

VM Tracker instrument, 507

W

waitForExpectationsWithTimeout method, 261

wAny bar, 200–202

@warning documentation comment, 448

Warnings

compiler, 29–30, 539

disclosure triangles, 208

displaying, 53, 55

suppressing, 155

Watchdog timer, 265
watchpoint commands, 524, 526–527
Welcome to Xcode window, 17–18
What's New in Xcode section, 439
Widget, 279
  build dependencies, 294–295
  data access, 282–285
  designing, 281–282
  extension, 290–294
  result, 295–296
  shared libraries, 285–290
  target, 280
widgetPerformUpdateWithCompletionHandler method, 293
Widths Equally, 203
WiFi instrument, 509
Wildcard patterns in searches, 89
windowDidLoad method, 346–347, 360
Wireless Accessory Configuration, 308
Wiring menus, 330–331
  file types, 336–338
  First Responders, 332–333
  icons, 339–340
  League Files, 338–339
  LeagueDocument data, 333–334
  managed documents, 334–335
  targets and actions, 331–332
  testing commands, 335–336
Wiring OS X applications. *See* Bindings
Wizard tab for Doxygen, 451, 453
WORA (write-once-run-anywhere) apps, 577
Workflows, 4
Working Directory: Use custom working directory, 517
-workspace for xcodebuild, 471–472
Wow feature of Assistant editor, 536
WRAPPER_EXTENSION, 555
WRAPPER_SUFFIX, 555
Write-once-run-anywhere (WORA) apps, 577

X

x-code-users list, 568
X11 package, 454
.xcassets files, 220
  images, 559
  OS X applications, 324
  overview, 113
  xcconfig files, 468–471
.xcdatamodeld file, 324
XCNotificationExpectationHandler class, 261
Xcode icon, 17
Xcode Overview section, 438
xcode-select tool, 11, 472–473
Xcode Server
  Accounts panel, 300
  repositories, 80, 85–86
xcode-users list, 568
XCODE_VERSION_ACTUAL, 552
xcodebuild tool, 463–464, 468–469, 471–472
.xcodeproj package, 471
xcrun tool, 473
XCTAssert assertion, 250
XCTAssertEqual assertion, 250, 262
XCTAssertEqualObjects assertion, 263
XCTAssertEqualWithAccuracy assertion, 257, 262
XCTAssertFalse assertion, 262
XCTAssertGreaterThan assertion, 263
XCTAssertGreaterThanOrEqual assertion, 263
XCTAssertLessThan assertion, 263

XCTAssertLessThanOrEqual assertion, 263
XCTAssertNil assertion, 262
XCTAssertNotEqual assertion, 261
XCTAssertNotEqualObjects assertion, 263
XCTAssertNotEqualWithAccuracy assertion, 262
XCTAssertNoThrow assertion, 263
XCTAssertNoThrowSpecific assertion, 263
XCTAssertNoThrowSpecificNamed assertion, 264
XCTAssertNotNil assertion, 250, 262
XCTAssertThrows assertion, 263
XCTAssertThrowsSpecific assertion, 263
XCTAssertThrowsSpecificNamed assertion, 264
XCTAssertTrue assertion, 262
XCTest assertion macro, 243
XCTest class, 250
  assertions, 261–264
  performance, 276–277
XCTestCase class, 243, 260–261
XCTestExpectation class, 261
XCTFail assertion, 262
XCUnitTest class, 244–245
Xemacs text editor, 572
XIB files, 143, 344
  linking, 158
  owners, 175
  xliff files, 396–397
XML
  property lists, 405, 421, 427–431
  refactoring names, 143
  stores, 336
XPC services, 517–518

Y

Yosemite Server, 80
Yosemite version, 9–10
  command-line tools, 11
  gestures, 380
  modules, 338
  state-restoration feature, 335
  storyboards, 343
  support, 13–14
  Xcode Server, 80

Z

Zombie technique, 518
Zoom In, 228
Zoom Out, 227
zooming
  instruments, 495
  Interface Builder, 159
Zuckerberg, Mark, 578