



Fritz Anderson

Xcode 5

Start to Finish

iOS and OS X Development



FREE SAMPLE CHAPTER

SHARE WITH OTHERS



Xcode 5

Start to Finish

Developer's Library Series



◆◆ 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 5 Start to Finish

iOS and OS X Development

Fritz Anderson

◆◆Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • 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.

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 5 start to finish : iOS and OS X development / Fritz Anderson.

pages cm

Includes index.

ISBN 978-0-321-96720-6 (pbk. : alk. paper)—ISBN 0-321-96720-8

(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.M33A53 2014

005.3'82—dc23

2014000229

Copyright © 2014 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, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

ISBN-13: 978-0-321-96720-6

ISBN-10: 0-321-96720-8

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

First printing, May 2014

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

Andrea Fox

Technical Reviewers

Duncan Champney

Chuck Ross

Dan Wood

Editorial Assistant

Olivia Basegio

Cover Designer

Chuti Prasertsith

Compositor

LaurelTech



*For Magdalen Jeanette Anderson (1952–2013),
the mother of my children*



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 **63**
- 7 Version Control **73**

II The Life Cycle of an iOS Application 101

- 8 Starting an iOS Application **103**
- 9 An iOS Application: Model **113**
- 10 An iOS Application: Controller **133**
- 11 Building a New View **147**
- 12 Autolayout in a New View **173**
- 13 Adding Table Cells **187**
- 14 Adding an Editor **205**
- 15 Unit Testing **221**
- 16 Measurement and Analysis **237**
- 17 Provisioning **255**

III Xcode for Mac OS X 275

- 18 Starting an OS X Application **277**
- 19 Bindings: Wiring an OS X Application **295**
- 20 A Custom View for OS X **323**
- 21 Localization **337**
- 22 Bundles and Packages **365**
- 23 Frameworks **381**
- 24 Property Lists **395**

IV Xcode Tasks 409

- 25 Documentation in Xcode **411**
- 26 The Xcode Build System **431**
- 27 Instruments **459**
- 28 Debugging **485**
- 29 Continuous Integration **499**
- 30 Snippets **511**

V Appendixes 525

- A Some Build Variables **527**
- B Resources **539**

Index **555**

Contents

Acknowledgments xxiii

About the Author xxv

Introduction 1

How This Book Is Organized 1

First Steps 2

The Life Cycle of an iOS Application 2

Xcode for Mac OS X 3

Xcode Tasks 4

Appendixes 4

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 18

A New Project 18

Quieting Xcode Down 21

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 31
- 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
- Linking 50
- Dynamic Loading 52
- Xcode and Clang 53
 - Local Analysis 54
 - Cross-Function Analysis 56
 - Indexing 57
- Compiler Products 58
 - Intermediate Products 58
 - Precompilation 60
- Summary 62

6 Adding a Library Target 63

- Adding a Target 63
 - Targets 64
- Target Membership 65
 - Adding Files to a Target 65
 - Headers in Targets 68

A Dependent Target	68
Adding a Library	69
Debugging a Dependent Target	70
Summary	70

7 Version Control 73

Taking Control	74
Creating a Git Repository by Hand	74
The State of Your Files	76
How Xcode Works with Git	77
Your First Commit	78
Working with Remote Repositories	79
Cloning an Existing Repository	79
Creating a Repository with Xcode Server	79
Adding a Reference to a Repository	80
Setting Up a “Remote”—Locally	80
Pushing to the Remote	83
Merges and Conflicts	83
User A	84
User B	87
Back to User A	90
The Version Editor	93
Comparison	93
Blame	95
Log	95
Branching	96
Summary	98

II The Life Cycle of an iOS Application 101

8 Starting an iOS Application 103

Planning the App	103
Model-View-Controller	103
The Model	104
The Views	104
The Controllers	106
Starting a New iOS Project	106
Target Editor	107

What's in the Project 108

One More Thing 110

Summary 112

9 An iOS Application: Model 113

Implementing the Model 113

Entities 114

Attributes 114

Relationships 117

Managed-Object Classes 120

Creating the Classes—the Wrong Way 120

Why Doing It Xcode's Way Is a Mistake 122

The Right Way—`mogenerator` 123

Extending the Classes 124

Some Test Data 126

Source Control and Product Files 128

Making the Model Easier to Debug 131

Summary 132

10 An iOS Application: Controller 133

Renaming Symbols 133

Refactoring a Method Name 134

Refactoring a Class Name 134

Editing the View Controller 136

The Table View 136

Setting Up the Passer List 137

Creating a New Passer 138

Live Issues and Fix-it 138

The Real Passer Rating 140

Another Bug 140

Running Passer Rating 144

Summary 145

11 Building a New View 147

The Next View Controller 147

If You Want to Add a View Controller 147

Storyboards, Scenes, and Segues 148

Building a View 152

Outlets and Assistants, in Passing 153

The Billboard View	154
Autolayout for the Nonce	157
Lots of Labels	158
The Table View	161
Outlets	161
Hooking Up the Outlets	164
Checking Connections	165
Connecting <code>PRGameListController</code>	165
Code Completion and Snippets	168
Testing the Billboard View	170
Summary	171

12 Autolayout in a New View 173

Why Autolayout?	173
Limitations of Autosizing	173
Autolayout	174
The Thing to Remember	174
The Player Billboard, Revisited	175
Why You Should Do More	175
Constraints for Real	176
The Label System	179
Summary	185

13 Adding Table Cells 187

The Game Table	187
Outlets in the Table View	187
Adding Required Protocol Methods	188
Adding Model-to-View Support	189
A Prototype Cell	190
The Game Table: First Run	191
A Custom Table Cell	193
Adding Some Graphics	196
A Cell with an Image in It	196
Hooking the Image View to the Images	197
The Assets Catalog	198
Adding Images to the Assets Catalog	199
Icons and Launch Images	201
Summary	202

14 Adding an Editor 205

- The Plan 205
- Adding a Modal Scene 205
 - An Embedded View Controller 206
 - Linking the Editor to the Passer List 208
 - Static Table Cells 209
- The Editor View Controllers 210
 - The Editor Table 210
 - Passing the Data to the Editor 213
 - Getting the Data Back 215
- Segues 218
- Summary 219

15 Unit Testing 221

- The Test Navigator 222
- Testing the CSV Reader 224
 - The CSV Test Code 224
 - Test Data 226
 - Running the Tests 227
- Testing and the Debugger 229
- Application Tests 232
- TestKit Assertions 233
 - Simple Tests 234
 - Equality 234
 - Exceptions 235
- Summary 236

16 Measurement and Analysis 237

- Speed 237
 - The Debug Navigator 238
 - Instruments 240
 - Optimization: First Try 243
 - Optimization: Second Try 245
 - Optimization: Third Try 246
- Memory 247
 - Allocations: First Look 248

Focusing on One Object Type	250
Cleaning Up the Transients	252
Summary	253

17 Provisioning 255

Apple Developer Programs	255
Organizations	256
Individuals	256
The Enterprise Program	256
Provisioning for iOS	257
What You'll See	258
Registering Your App	258
Protecting Your Assets	261
Submitting an iOS Application	261
The Capabilities Editor	262
Capabilities for Both iOS and OS X	262
iOS Capabilities	263
OS X Capabilities	263
OS X Sandboxing	264
Why Sandbox?	265
Why Not Sandbox?	266
Gatekeeper and Developer ID	266
Getting a Developer ID	266
Using Developer ID	267
Limitations	269
Distribution Builds	269
Basic Build Settings	269
Adjusting the Build Settings	270
The Build	272
Summary	273

III Xcode for Mac OS X 275

18 Starting an OS X Application 277

The Goal	277
Getting Started	278
Model	281

- Porting from iOS 282
- Adding an Entity 282
- Wiring a Menu 287
 - Target/Action 288
 - First Responder 289
 - Loading Data into `MPRDocument` 289
- Summary 293

19 Bindings: Wiring an OS X Application 295

- Laying Out the Document Window 295
 - A Table View 296
 - Autosizing 299
- Filling the Table—Bindings 301
 - Object Controllers 302
 - Binding the Team Table 305
 - Running Bindings 305
- Layering `NSControllers` 307
 - The Passer and Game Array Controllers 309
 - How Object Controllers Chain 310
 - Binding the Passer Table 311
 - Data Formatters—Numbers 313
 - Data Formatters—Strings and Dates 314
- Running a Popover with Bindings 315
 - Another Excursion into Autolayout 318
- Running the Near-Final App 321
- Summary 322

20 A Custom View for OS X 323

- A Graphing View 325
- Back to the View Controller 328
 - Using `MPRPasserGraphController` 330
- QuickLook in the Debugger 332
- Custom View Properties 334
- Summary 336

21 Localization 337

- How Localization Works 337
- Adding a Localization 338

Base Localization	338
Localizing for French	339
Trying It Out	345
Localizing <code>MainMenu.xib</code>	347
Bringing a File into Localization	352
Localizing <code>Info.plist</code>	353
Strings in Code	355
Showing Mac Passer Rating in Finder	359
Adding the Icons	359
Summary	364

22 Bundles and Packages 365

A Simple Package: RTFD	365
Bundles	367
Application Bundles	367
The <code>Info.plist</code> File	369
Localizing <code>Info.plist</code>	370
<code>Info.plist</code> Keys for Applications	371
Keys for Both iOS and OS X	371
Keys for OS X	373
Keys for iOS	376
<code>Info.plist</code>	379
Summary	379

23 Frameworks 381

Adding a Framework Target	381
Populating the Framework	382
Using the Framework	383
Installing a Framework	383
Running the Application Alone	383
Where Frameworks Go	385
Putting the Framework in the Application	386
Building Mac Passer Rating	387
One More Thing	388
Debugging a Framework	388
The Debugger and External Targets	389
Postmortem on Postmortem Debugging	393
Summary	394

24 Property Lists 395

- Property List Data Types 395
- Editing Property Lists 396
 - The Property List Editor 399
 - Why Not the Property List Editor? 404
- Other Formats 406
 - Text Property Lists 406
 - Binary Property Lists 407
- Specialized Property Lists 407
- Summary 408

IV Xcode Tasks 409**25 Documentation in Xcode 411**

- Quick Help 411
 - Inspector 411
 - Popover 412
- Open Quickly 413
- Help 414
- The Documentation Window 415
 - The Navigator Sidebar 415
 - The Table of Contents Sidebar 415
 - Class Info 416
 - Searching and Navigation 417
- Keeping Current 419
- Your Own Quick Help 421
 - How to Generate Quick Help 421
 - Documentation Comment Syntax 422
- Your Own Docsets 424
 - Preparation 425
 - Configuring Doxygen: Basic Settings 425
 - Configuring Doxygen: Expert Settings 427
 - Running Doxygen 428
 - Installing a Docset 429
- Summary 430

26 The Xcode Build System 431

- How Xcode Structures a Build 431
- Build Variables 434
- Settings Hierarchy 435
 - Levels 436
- Editing Build Variables 437
- Configurations 438
 - Adjusting Configurations 438
- Configuration Files 439
 - Creating a Configuration File 439
 - SDK- and Architecture-Specific Settings 441
 - Preprocessing `xconfig` Files 442
- Command-Line Tools 443
 - `xcodebuild` 444
 - `xcode-select` 445
 - `xcrun` 446
- Custom Build Rules 446
- The Build Log 448
- A Simple Build Transcript 450
- Summary 458

27 Instruments 459

- What Instruments Is 459
- Running Instruments 460
 - The Trace Document Window 461
 - The Library 467
 - Instrument Configuration 468
 - Recording 470
 - Saving and Reopening 472
- The Instruments 474
 - Behavior 474
 - Core Data 474
 - Dispatch 474
 - Filesystem 475
 - Garbage Collection 475
 - Graphics 475
 - Input/Output 475

Master Track	476
Memory	476
System	477
System—iOS Energy Instruments	478
Threads/Locks	479
Trace	479
UI Automation	480
User Interface	480
Custom Instruments	480
The Templates	482
All Platforms	482
iOS Only	483
Mac Only	483
Summary	484

28 Debugging 485

Scheme Options	485
Info	485
Arguments	486
Options	486
Diagnostics	488
Doing More with Breakpoints	488
The lladb Command Line	491
Tips	493
Summary	497

29 Continuous Integration 499

Xcode Server	500
Repositories	501
Settings	503
Turn It On	503
Bots	503
Creating a Bot in Xcode	503
Create a Bot on the Web	505
Running a Bot	506
Seeing the Results	507

Building for Distribution	508
Summary	509

30 Snippets 511

Tricks	511
General	511
Code-Folding Ribbon	515
The Assistant Editor	515
Instruments and Debugging	518
Building	518
Workspaces	520
Traps	522

V Appendixes 525

A Some Build Variables 527

Useful Build Variables	528
Environment	528
Code Signing	529
Locations	530
Source Locations	530
Destination Locations	530
Bundle Locations	532
Compiler Settings	533
Search Paths	535
Info.plist	535
The DEVELOPER_ Variables	536
Source Trees	537

B Resources 539

Books	539
On the Net	540
Forums	540
Mailing Lists	541
Developer Technical Support	542
Sites and Blogs	542
Face to Face	544
Meetings	544
Classes	544

- Other Software 544
 - Text Editors 545
 - Helpers 546
 - Package Managers 548
 - Version Control 549
 - AppCode 550
 - Alternatives to Cocoa 551
 - Alternatives to Objective-C 553

Index 555

Acknowledgments

Only part of the effort that went into putting *Xcode 5 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 guided me through the from-scratch planning process that a new Xcode book required, and advocated for it when the need was not obvious.

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

The reviewers—Duncan Champney, Chuck Ross, and Dan Wood—were friends to the book even through the burden of trying to make sense of a work that spiraled, rather than marched, to its conclusion. They 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.

Julie Nahil, the production manager, and Stephanie Geels, the copy editor, made it exceptionally easy to give you a book that is as close as it can be to what I meant it to be. The process was never smoother.

A full-time day job is not an author's best friend (except for the part about paying the rent), but Emerging Technologies, in the IT Services department of The University of Chicago, was a friend to me. My boss, Alan Takaoka, got me three-day weekends while I wrote, even when I ran over by a couple of weeks. I promised to keep all my Monday meetings and deadlines while I worked on the book, but my colleague, Cornelia Bailey, made the deadlines on our projects disappear.

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. 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 Emerging Technologies and Communications division at The University of Chicago. He has two daughters.

This page intentionally left blank

Introduction

Welcome to *Xcode 5 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 5 is the descendant of a family of development tools dating back 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 5 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 5. 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 for *Xcode 5 Start to Finish*. 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.

The full code for the example programs is available online from informit.com/9780321967206 (register your book to gain access to the code). In the interest of space, I’ll be showing only excerpts.

Xcode supports some technologies, like Core Data and OS X bindings, that are *not* for beginners. *Xcode 5 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. Again, I’ll be showing you 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 5; 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 for a description of 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, Autolayout in a New View**—In Xcode 5, autolayout 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, 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 18, 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 19, 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 20, A Custom View for OS X**—Add a custom view to your app, and see how Interface Builder can lay it out and configure it, even though it's not a standard Cocoa component.
- **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, Frameworks**—Package and share a complete subprogram you can incorporate into any OS X application.
- **Chapter 24, 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 25, 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 26, The Xcode Build System**—I'll show you the fundamental rules and tools behind how Xcode goes from source files to executable products.
- **Chapter 27, 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 28, 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 29, Continuous Integration**—Mavericks Server complements Xcode 5 with a sleek continuous-integration system that can synthesize your code analysis, perform cross-platform unit tests, and generate product archives. I'll show you how to get started, and how to put it to best use.
- **Chapter 30, 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 2013, shortly after Apple released iOS 7, OS X Mavericks, and Xcode 5 to the public. *Xcode 5 Start to Finish* is written to the first-bugfix versions of all three.

About the Code

Xcode 5 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 5 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/9780321967206` (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 many, many 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; 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

Kicking the Tires

Now you have Xcode. It's time to start it up and see what it looks like.

Starting Xcode

You'll find Xcode in the `/Applications` directory, just like any application. You'll be using it constantly, so you'll want to keep it in the Dock at the bottom of your main screen. Drag Xcode to the Dock—take care to drop it *between* icons, and not *on* one.

Now click on the Xcode icon. It bounces to show Xcode is being launched. The first time you run any of Apple's developer tools—even through the command line—you'll be asked to read and accept a license agreement for the tools and SDKs. It's no different from any other click-through license process.

Next, Xcode will ask you for permission to install “additional components” it needs. Permit it, and present an administrator's credentials. Those components overlap the iTunes frameworks, so you may be asked to close iTunes.

Once the progress window clears, you are greeted with the “Welcome to Xcode” window (see Figure 2.1).

If this is the first time you've ever run Xcode, the table on the right will be empty (“No Recent Projects”); as you accumulate projects, the table will contain references to them, so you have a quick way to get back to your work. When you accumulate projects in this list, you'll be able to select one, but Xcode doesn't reveal any way to open it. The trick is to double-click the item, or press the Return key.

You have three other options:

- **Create a new Xcode project.** This is obvious enough; it's how you'd start work on a new product. You're about to do this, but hold off for the moment. You could also select **File** → **New** → **New Project...** (⌘ ⌘ N).
- **Check out an existing project.** Xcode recognizes that source control management is essential to even the most trivial of projects. Your development effort might start not with your own work, but with collaborative work pulled in from a source repository. Use this link to get started.



Figure 2.1 When you launch Xcode, it displays a “Welcome” window with options for creating a new project, reopening a recent one, or fetching a project from a source-control repository.

- **Open Other. . .** (at the bottom of the “recents” list). This will get you the standard get-file dialog so you can select any Xcode project file you want. You can do the same thing with the **File** → **Open. . .** (**⌘ O**) command.

If you need to get back to the Welcome window, select **Window** → **Welcome to Xcode** (**⌘ 1**). If you’re tired of seeing this window, uncheck **Show this window when Xcode launches**.

Note

“Show this window when Xcode launches” is not quite accurate. If you had a project open when you last quit Xcode, it will reopen automatically when you start it up again, and the Welcome window won’t appear.

Hello World

Just to get oriented, I’m going to start with the simplest imaginable example project—so simple, you won’t have to do much coding at all.

A New Project

Click the **Create a new Xcode project** link. Xcode opens an empty Workspace window, and drops the New Project assistant sheet in front of it (see Figure 2.2). Select **OS X** → **Application** from the list at left, and then the **Command Line Tool** template from the icons that appear at right. Click **Next**.

The next panel (Figure 2.3) asks for the name of the project and the kind of command-line tool you want.

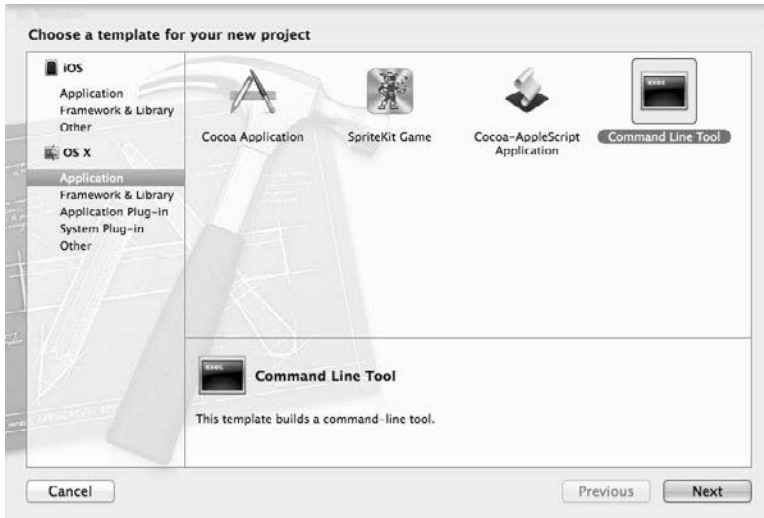


Figure 2.2 The New Project assistant leads you through the creation of an Xcode project. First, you specify the kind of product you want to produce.

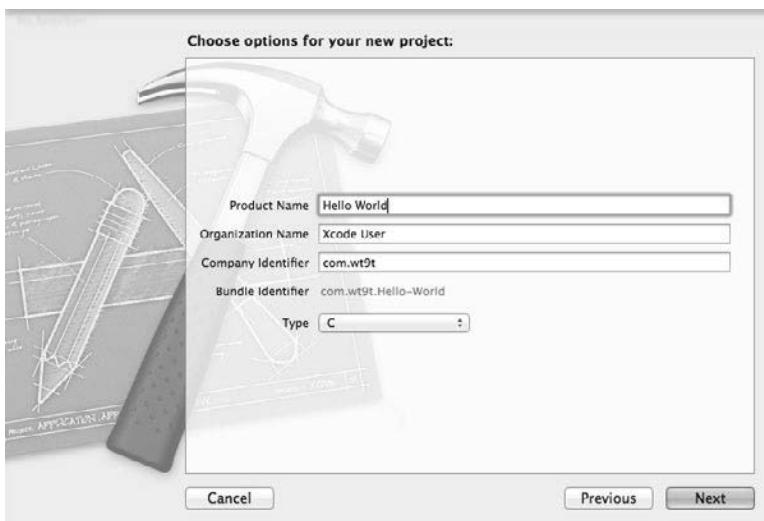


Figure 2.3 The **Options** panel of the New Project assistant lets you identify the product and what support it needs from system libraries.

1. Type **Hello World** in the **Product Name** field. This will be used as the name of the project and its principal product.
2. Xcode should have filled the **Organization Name** in for you, from your “me” card in the Address Book. If you listed a company for yourself, that’s what will be in the field; otherwise, it’s your personal name. Xcode will use this as the name of the copyright holder for all new files.
3. Virtually all executable objects in the OS X and iOS world have unique reverse-DNS-style identifiers that are used to uniquely identify them. The **Company Identifier** is the leading portion of those reverse-DNS names, to be used by every product of this project. For this example, I used `com.wt9t`.
4. By default, Xcode infers the unique **Bundle Identifier** from the company identifier and the name of the product. You’ll see later how to change this if you have to.
5. The **Type** popup prompts Xcode on how to fill in the system libraries the tool will use. This is just a plain old C program, with no need for C++ or Apple-specific support, so choose **C**.

Finally, a put-file sheet appears, so you can select a directory to put the project into. For this example, select your Desktop. One more thing—*uncheck* the box labeled **Create local git repository for this project**. Source control (Chapter 7, “Version Control”) is a Good Thing, but let’s not deal with it in this trivial example. Click **Create**.

If you look on your Desktop, you’ll find that Xcode has created a folder named `Hello World`. The project name you specified is used in several places.

- It’s the name of the project *directory* that contains your project files.
- It’s the name of the project *document* (`Hello World.xcodeproj`) itself.
- It’s the name of the *product*—in this case a command-line tool named `Hello World`.
- It’s the name of the *target* that builds the product. I’ll get into the concept of a target later; for now, think of it as the set of files that go into a product, and a specification of how it is built.
- It’s the name of the *target’s* directory inside the *project’s* directory.
- Suitably modified, it’s the name of the man-file template for the tool (`HelloWorld.1`).

When you’ve made your choices, Xcode un.masks the workspace for the Hello World project (Figure 2.4). Don’t look at it too closely just yet. Xcode starts you off with a view of the settings that control how `Hello World` is to be built. This is useful information, but for now, it’s just details.

More interesting is the program code itself. The left column of the window is called the *Navigator area*. Find `main.c` in the list, and click it (see Figure 2.5). The Editor area, which fills most of the window, now displays the contents of `main.c`. This is the code for the canonical simplest-possible program, known as “Hello, World.”

The Navigator area displays many different things in the course of development—it’s not just a file listing. It can display analyses, searches, and build logs. Which list you see often depends on what Xcode wants to show you; you can make the choice yourself by

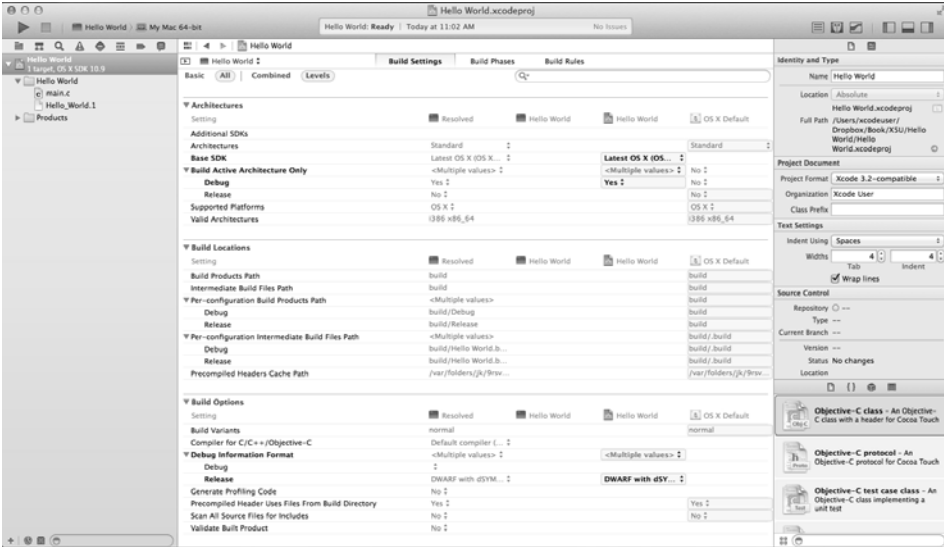


Figure 2.4 Once set up, the Hello World project window fills in with a list of source files and a display of the options that control how the application will be built.



Figure 2.5 Clicking the name of an editable file in the Project navigator displays its contents in the Editor area.

clicking the tiny icons in the bar at the top of the Navigator area. Hovering the mouse pointer over them will show you the names of the various views.

As this book goes on, you'll meet all of them. For now, you care only about the "Project" navigator, the file list Xcode starts you out with. Feel free to click the other icons, but to keep up with this example, be sure to return to the Project navigator, as shown in Figure 2.5.

Quieting Xcode Down

But first.

Xcode is a toolset that contains everything its creators could think of to provide a powerful, helpful environment for writing iOS and OS X applications. Often, you barely need to begin a task, and Xcode will offer to finish it for you. It will usually be right. I use these features all the time. I recommend them.

You're going to turn them all off.

Automatic completions and indentations and code decorations and code fixes are great, once you know what's going on, but an automaton that snatches your work out of your hands, however helpfully, is straight out of *The Sorcerer's Apprentice*. Better to start with what *you* want to do; once you're confident of what that is, then you have the discretion and control to direct Xcode as it helps you.

So you're going to damp Xcode down a bit. You'll do all of this in Xcode's Preferences window, which you can summon with **Xcode** → **Preferences...** (⌘ comma). The Preferences window is divided into panels, which you select with the icons at the top of the window.

To start, make sure the **General** panel is visible. Under **Issues**, uncheck **Show live issues**.

Next, select the **Text Editing** panel, which has two tabs. Select the **Editing** tab, and uncheck **Show: Code folding ribbon**, and all the options under **Code completion**.

In the **Indentation** tab, turn off **Line wrapping: Wrap lines to editor width** and the **Syntax-aware indenting** section.

Now Xcode will mostly stay out of your way as you explore.

Building and Running

The program in `main.c` would run as is, but we have to trick Xcode into keeping its output on the screen long enough to see it. Add a few lines after the `printf` call so it looks like this:

```
int main(int argc, const char * argv[])
{

    // insert code here...
    printf("Hello, World!\n");

    /*****
     * Pause, so the console doesn't disappear
     *****/
    char    dummy[128];
    fgets(dummy, sizeof(dummy), stdin);

    return 0;
}
```

Now we can run it. Select **Product** → **Run** (⌘ R).

In the ordinary course, Xcode would then build and run `Hello World`. However, if this is the first time you've run any application, there is a security problem: Running an app from Xcode puts it under the observation of a debugger, which will have access to the internal data and running state of the app. Crossing process boundaries like that is technically a security breach, and you have to authorize it.

Xcode posts an alert, **Enable Developer Mode on this Mac?** It explains that you could be asked for an administrator’s password every time you run the debugger (“**Developer Tools Access needs to take control of another process. . .**”), or, with Developer Mode, you could do the authorization once and forget about it. Click **Enable**, enter an administrator’s credentials, and carry on.

Note

You can turn Developer Mode off, or on again, from the Devices organizer (⌘⌘2). Select your Mac and click the button **Disable Developer Mode**.

With authorization taken care of, a heads-up window (“bezel”) appears almost instantly, to tell you “Build Succeeded.” (If Xcode is in the background, a notification banner will appear saying the build succeeded, and identifying the project and product involved.)

So. What happened?

Hello World is a console application; it just writes out some text without putting up any windows. Xcode captures the console of the apps it runs in the Debug area, which popped into view when you ran the program (Figure 2.6). The Debug area includes a console view on the right. It says Hello, World! (Figure 2.7).

Click in the console to make it ready for text input, and press the Return key. Hello World exits, and the Debug area closes.

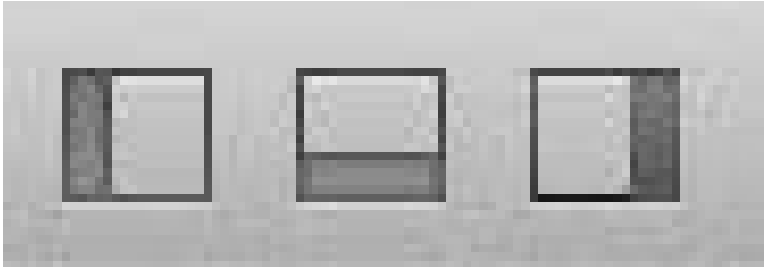


Figure 2.6 The **View** selector in the toolbar shows and hides the Navigator, Debug, and Utility areas (left to right) of the project window. Clicking a button so it is highlighted exposes the corresponding area. Here, the Navigator and Debug areas are selected.

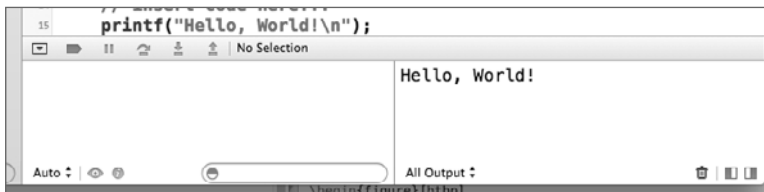


Figure 2.7 Opening the Debug area after running Hello World shows the eponymous output.

Note

If the Debug area didn't hide itself as soon as an application terminated, we wouldn't have had to add that `fgets()` call. That's easy to change; see the "Behaviors" section of Chapter 4, "Active Debugging."

The Real Thing

What Xcode just produced for you is a real, executable application, not a simulation. To prove it, open the Terminal application (you'll find it at `/Applications/Utilities/Terminal`, and you'd be well advised to add Terminal to your Dock). In Xcode, find the `Hello World` product in the Project navigator by clicking the disclosure triangle next to the Products folder icon. Drag the `Hello World` icon into the Terminal window, switch to Terminal, and press the Return key. (The path to a file deep in a directory for build products is remarkably long, but Terminal takes care of the necessary escaping.) "Hello, World!" appears.

If you want access to the executable file itself, select it in the Project navigator, then **File** → **Show in Finder**—also available in the contextual menu you get by right-clicking the `Hello World` icon. A window will open in the Finder showing you the file.

You're done! You can close the Workspace window (**File** → **Close Project**, `⌘W`) or quit Xcode entirely (**Xcode** → **Quit Xcode**, `⌘Q`).

Getting Rid of It

There is nothing magical about an Xcode project. It's just a directory on your hard drive. If you don't want it any more, close the project, select its enclosing folder in the Finder, and drag it to the Trash. It's gone. It won't even show up in the Recents list in the Welcome to Xcode window, or in the **File** → **Open Recent** menu.

That's it.

Okay, yes, the build products of the project will still stick around in a warren of directories inside `~/Library/Developer/Xcode/DerivedData`. They aren't many or large in this case, but there's a principle involved.

If you want them gone, the best way is to close the project window, open the Organizer window (**Window** → **Organizer**, `⌘2`), select the **Projects** panel, select the "Hello World" project, press Delete, and confirm the deletion in the ensuing alert sheet. All trace of the build products is gone.

Summary

In this chapter, you had your first look at Xcode, and you discovered that it doesn't bite. You saw how to create a simple project, one you didn't even have to edit. You saw what happens when you run a project in Xcode, how to close a project and quit Xcode, and at last how to get rid of the project entirely.

Next, we'll start doing some real work.

Index

A

- .a (static libraries), 63, 519**
- Accessibility package, 14**
- Accessory setting**
 - details, 209
 - editors, 191
- Accounts panel**
 - Apple ID, 256
 - bots, 504
 - Developer ID, 266
 - iOS provisioning, 258–259, 261
 - remote repositories, 82, 84
 - version control, 74
- Actions**
 - for wiring menus, 288, 291
 - xcodebuild, 444
- Activate/Deactivate Breakpoints, 39**
- Active Allocation Distribution style, 469**
- Activity Monitor instrument, 477**
- actool compiler, 456**
- Ad hoc (beta) distributions, 257, 262, 270–272**
- AD.HOC.CODE.SIGNING.ALLOWED, 529**
- Add, 33**
- Add an Account, 259**
- Add Apple ID, 256, 258**
- Add Breakpoint at Current Line, 36**
- Add Build Rule, 446–447**
- Add Constraints, 161, 178–179, 182, 184**

- Add Copy Files Build Phase, 386**
- Add Entity, 114**
- Add Exception Breakpoint, 141, 307**
- Add Files to**
 - availability, 282
 - get-file sheets, 110, 124, 227
 - mogenerated directory, 284
 - projects, 432, 520
 - target pickers, 67
- Add Item, 400–401**
- Add Localization, 339**
- Add Missing Constraints in Container, 160**
- Add Missing Constraints in Game List Controller, 157**
- Add Other, 65**
- Add Relationship, 117**
- Add Remote, 80, 82**
- Add/Remove Breakpoint at Current Line, 495**
- Add Run Script Build Phase, 127**
- Add shortcut, 471**
- Add Target, 63, 222, 381**
- Add to targets table, 111**
- Add User-Defined Setting, 527**
- Added file state, 77**
- Adobe PhoneGap, 551**
- Advanced attributes for models, 116**
- Agent applications, 375**
- Alert sheets**
 - in debugging, 33
 - version control, 90
- Alerts in localizations, 357–358**
- Alignment**
 - labels, 159, 181–182
 - text-cell objects, 309
- All in builds, 435, 449**
- All Entities, 427**
- All Frames in Container, 178**
- All Issues, 449**
- All Messages, 449**
- All Objects Created, 250**
- All Processes, 462**
- Allocation Density style, 469**
- Allocation Lifespan settings, 250–251**
- Allocation of memory, 248–250**
- Allocations & Leaks service, 471**
- Allocations instruments, 469, 476, 518**
- Allocations list, 252**
- Allow Location Simulation, 487**
- alltargets, 444**
- Always use deferred mode, 471**
- Analysis and measurement, 237**
 - memory, 247–253
 - speed. *See* Speed
- Analysis message display, 56**
- analyze action for xcodebuild, 444**
- Antecedents in makefile goals, 431**
- Antialiasing, 378**
- .app directory, 367**
- App Store**
 - archives for, 508
 - Enterprise program, 256, 270–271
 - installer packages, 387
 - OS X applications, 264, 279
 - program members, 259
 - provisioning, 255, 257
 - sandboxing, 264–266
 - Xcode downloads, 10–13
 - Xcode Server, 500
 - Xcode updates, 445
- .app suffix, 445**
- Appcelerator Titanium, 551–552**
- AppIcon image set, 201**
- AppKiDo tool, 546**
- Apple Developer Forums, 541**
- Apple developer programs, 12–13, 255–256**

- Application Data popup, 487**
- Application icon, 359**
- Application IDs, 257**
- Applications**
 - bundles, 367–369
 - iOS. *See* iOS
 - Info plist keys for, 371–373
 - registering, 258–260
 - tests, 232–233
- /Applications directory, 10, 17**
- apropos in lldb, 491**
- ARC (Automatic Reference Counting), 193**
- arch in clang, 452**
- Architecture-specific build settings, 441–442**
- archive action in xcodebuild, 444**
- Archives organizer, 267, 273, 387**
- ARCHS, 533**
- ARCHS_STANDARD, 438, 533**
- Arguments Passed On Launch table, 307**
- Arguments tab**
 - binding debugging, 307
 - localizations, 346
 - schemes, 486
- ARM processors, 53**
- Assembly listings, 59–60**
- assert macro, 511**
- Assertions**
 - description, 221
 - TestKit, 233–235
- Asset catalog, 456**
- Assets, protecting, 261**
- Assets catalog, 198**
 - adding images to, 199–201
 - image sets, 198–199
- Assignments (=) in Boolean contexts, 140**
- Assistant editor**
 - assembly display, 59
 - caller display, 58
 - connection checks, 165
 - Editor area, 153
 - Interface Builder, 150
 - jump bar, 155–156, 315
 - localizations, 347
 - Option-key navigation, 413
 - overview, 515–517
 - Preview, 156–157
 - views, 153, 162, 164
- Associative arrays, 396, 407**
- At sign (@) notation, 490**
- atIndexPath method, 137**
- ATSApplicationFontsPath key, 374**
- Attach to Process, 388, 463**
- att Color setting, 334–335**
- Attributes for models, 114–117**
- Audio package, 14**
- @author keyword, 423**
- Authorization in iOS provisioning, 257**
- Authorized devices in iOS provisioning, 257**
- Autolayout, 173**
 - labels, 160, 179–185
 - localizations, 341–342
 - overview, 174
 - purpose, 173–174
 - size constraints, 175–179
- Automatic code completion, 22, 28, 168–170**
- Automatic for Assistant editor, 153**
- Automatic Reference Counting (ARC), 193**
- Automatic Snapshotting, 477**
- Automation instrument, 480**
- Autorelease pools, 252–253**
- Autoresize**
 - document window, 299–301
 - limitations, 173–174
 - localizations, 341

Auxiliary tools, 14
awakeFromNib, 334

B

@b debug format, 423

B2B program, 256

Badges for test navigator, 222

Bar graphs, 32

CPU and memory, 291

Debug navigator, 191–192

Base localization, 337–339

Basic button, 435

BBEdit text editor, 349, 545

**Beta (ad hoc) distributions, 257, 262,
 270–272**

Billboard view

label constraints, 179–185

size constraints, 175–179

testing, 170–171

view, 154–158

Binaries, fat, 454

Binary property lists, 407

Bindings, 295

data formatters, 313–315

filling, 301–307

NSController layering, 307–315

object controllers, 302–304, 310–311

popovers, 315–320

running, 305–307

Blame view in Comparison editor, 93, 95

Block Graph style, 469

Blogs, 542–543

Bluetooth instrument, 478

Bookmark navigator, 415

Books, 539–540

Borders for buttons, 206

Bots, 503

creating, 503–506

results, 507–508

running, 506

Branching in version control systems, 96–98

breakpoint in lldb, 492

Breakpoint navigator, 141

breakpoint set, 496

Breakpoints, 35

bindings, 307

listing, 141

removing, 36

setting, 36–37

tips, 493

unit testing, 229

working with, 488–491

Browse All Versions, 305

@bug keyword, 423

build action in xcodebuild, 444

Build Configuration popup, 438

Build for popup, 323

Build For Running, 448

Build New Instrument, 482

Build Phases tab

description, 64

extending classes, 127

frameworks, 382–383, 386

iOS projects, 110

Link Binaries with Libraries, 69, 110

targets, 50, 65–66, 432–433

Build Rules tab, 64, 446–447

Build settings, 434–435, 527–528

code signing, 529–530

Compiler, 533–535

DEVELOPER, 536–537

environment, 528–529

Info.plist, 535–536

locations, 530–532

- search paths, 535
- source trees, 537
- Build Settings tab**
 - build settings, 434–438, 527–528
 - code size, 511
 - configuration files, 441
 - flags, 534
 - frameworks, 386–387, 393
 - garbage collection, 534
 - hierarchy, 435–436
 - packages, 370
 - product names, 279
 - Quick Help, 412, 422
 - release size, 511
 - SDK, 108
 - targets, 64, 272
- Building views, 152**
 - labels, 158–160
 - outlets and Assistant editors, 153–154
- Builds and build system, 431**
 - command-line tools, 443–446
 - configuration files, 439–443
 - configurations, 438–439
 - custom rules, 446–448
 - distribution, 269–273, 508–509
 - logs, 448–449
 - projects, 22–24, 29–31
 - settings, 434–435, 437–438
 - settings hierarchy, 435–437
 - structures, 431–434
 - transcript, 450–458
 - tricks, 518–520
- BUILT_PRODUCTS_DIR, 531**
- Bumgarner, Bill, 518**
- Bundle Identifier setting**
 - new projects, 20
 - OS X applications, 279

- Bundles, 365, 367**
 - application, 367–369
 - Info.plist keys and file, 369–373, 379
 - location settings, 532
 - .strings files, 356
 - targets, 523
- Button borders, 206**

C

- .c files, 434**
- @c keyword, 423**
- C++ language, 554**
- Call-tree constraints, 466**
- Call Tree listing, 241**
- Cancel Integration, 508**
- Canvas**
 - displaying, 177
 - labels, 181, 183
 - segues on, 218
 - view controllers, 149–150
- Canvas menu, 160, 176**
- Capabilities editor, 262–263**
- Capabilities tab, 264**
- Carbon Events instrument, 480**
- Cartesian coordinates for views, 324**
- Cascade delete rule, 118**
- CC, 434**
- Cell Based, 296**
- cellForRowAtIndexPath method**
 - custom cells, 194–196
 - images, 197–198, 299
 - outlets, 188
 - prototype cells, 190–191
 - table view, 137
- Cells. *See* Tables and table cells**
- Centering constraint, 183**

Certificates

- code signing, 529
- Developer ID, 12, 266–268
- distribution builds, 269–270, 508–509
- iOS provisioning, 257, 261
- private keys, 261
- team membership, 258–260

Certificates, Identifiers & Profiles site, 258**Change color, 427****Check and Install Now, 421****Check for and install updates automatically, 421****Check out an existing project, 17, 74, 79, 83****Check Out, 83****Choose Target, 463****clang compiler, 47**

- builds, 452
- cross-function analysis, 56–57
- indexing, 57–58
- local analysis, 54–56
- modules, 60–62
- overview, 53–54
- precompilation, 60

Class Info settings, 416–417**Class Prefix, 107****Classes (educational), 544****Classes (objects)**

- document localizations, 360
- extending, 124–126, 162, 189
- managed-object. *See* Managed-object classes
- name refactoring, 134–136
- object allocations by, 459

clean action for xcodebuild, 444**Cleaning up transients, 252–253****Clear Constraints, 177****Cloning repositories, 79****Close Project, 24****cocoa-dev list, 542****Cocoa Events instrument, 480****Cocoa language application frameworks**

- alternatives, 551–552
- Core Data, 107
- libraries, 69
- pointers, 54

Cocoa Touch framework, 103, 168**CocoaHeads meetings, 544****CocoaPods package manager, 549****Code completion, 22, 28, 168–170****Code completion: Automatically insert closing “{”, 28****Code-folding ribbon, 515–516****Code Signing Identity, 238****Code signing settings, 529–530****Code snippets, 169–170****Color**

- labels, 159
- tables, 308
- views, 155, 334–335

Color controls

- palette, 155
- picker, 308
- well, 155

Column Sizing setting, 297**Combined for build settings, 435****Combo fields for property lists, 408****Command Line Developer Tools package, 12****Command Line Tool template, 18****Command-line tools, 11**

- arguments, 347
- builds, 443–446
- package, 14

Comments, 422–424**Commit editor, 78–79**

- Commit sheet, 86**
- Commits**
 - selective, 85–87
 - version control systems, 78–79, 92–93
- Company Identifier setting**
 - iOS, 106
 - new projects, 20
 - OS X, 279
- Compare Call Trees, 245**
- Comparison editor, 93–95**
 - Blame view, 95
 - Log view, 95–96
- compColor property, 335**
- Compile for controllers, 138–139**
- Compile Sources build phase, 50, 432–433**
- CompileAssetCatalog phase, 456**
- Compilers and compiling, 45**
 - build settings, 533–535
 - clang, 53–54
 - controllers, 138–140
 - cross-function analysis, 56–57
 - dynamic loading, 52–53
 - indexing, 57–58
 - intermediate products, 58–62
 - linking, 50–52
 - local analysis, 54–56
 - precompilation, 60–62
 - process, 45–50
 - warnings, 29–30, 518
- Completes action, 42**
- Completion, code, 22, 28, 168–170**
- componentsSeparatedByCharactersInSet method, 229**
- componentsSeparatedByString method, 252**
- Condition field for breakpoints, 490**
- Conditionally Sets Editable, 310**
- Configuration files, 439–443**
- Configuration tab for bots, 505**
- Configure Repository sheet, 79–80**
- configureAtIndexPath method, 137**
- configureView method, 165–168**
- Conflicted file state, 77**
- Conflicts**
 - assignments, 513
 - version control systems, 83–93
- Connect to a Git Repository, 502**
- Connect to a Subversion Repository, 502**
- Connecting outlets, 153–154**
- Connection inspector for First Responders, 289**
- Connections for outlets, 164–168**
- Connections instrument, 477**
- Console applications, 23**
- Console windows, 494**
- Constraints**
 - description, 175–176
 - labels, 179–185
 - size, 176–179
 - trace document window, 466
 - views, 157, 174–176
- Content Set, 310**
- Contents directory, 367, 369**
- Continue, 39**
- Controller Key setting, 304–305**
- Controller layers, 133**
 - MVC model, 104, 106
 - object, 302–304
 - view. *See* View controllers
- Converting**
 - data types, 168, 406
 - .iconset to .icns, 361–363
 - property list formats, 407
- Coordinates for views, 324**
- Copy**
 - configuration files, 441
 - dictionaries, 401

- Copy Bundle Resources build phase, 129, 432, 448**
 - folder references, 513
 - Info.plist file, 379
 - targets, 66, 68
- Copy items into destination group's folder (if needed), 111, 282, 520–521**
- Copy only when installing, 386**
- Copy Source Changes, 95**
- Copy Transcript for Shown Results, 450**
- Core Animation instrument, 475**
- Core Data**
 - events, 460
 - model objects, 113
- Core Data Cache Misses instrument, 474**
- Core Data Faults instrument, 474**
- Core Data Fetches instrument, 474**
- Core Data Saves instrument, 474**
- Counters instrument, 477**
- CPU Activity instrument, 478**
- CPU bar for speed analysis, 238–239**
- CPU Monitor instrument, 477–478**
- CPU Usage style, 469**
- Create a new Xcode project, 17, 25**
- Create Bot, 504–505**
- Create Document-Based Application, 279**
- Create folder references for any added folders, 512**
- Create git repository on, 25, 74, 279**
- Create groups for any added folders, 111, 282**
- Create local git repository for this project, 20**
- Create New Remote, 79**
- Create NSManagedObject Subclass, 120**
- Create Symbolic Breakpoint, 495**
- Created & Destroyed, 251**
- Created & Still Living, 251**

- Credits.rtf file**
 - localizations, 341–346
 - OS X applications, 280
- Cross-function analysis, 56–57**
- CSResourcesFileMapped key, 374**
- .csv data files, 224**
- CSV Reader, 224–228**
- CSVFileTests class, 226**
- Current Bytes style, 469**
- Current Views, 40**
- Custom build rules, 446–448**
- Custom instruments, 480–482**
- Custom script, 447**
- Custom segues, 218**
- Custom table cells, 193–196**
- Custom views**
 - Graphing, 325–328
 - overview, 323–325
 - properties, 334–336
 - view controller, 328–332

D

- DarwinPorts package manager, 549**
- Dash styles tool, 546–547**
- Dashcode package, 14**
- Data formatters**
 - numbers, 313–314
 - strings and dates, 314–315
- Data Model editor, 114**
- Data Model inspector, 116–117, 119**
- Data Protection, 263**
- Data tips, 332–333**
- Data types for property lists, 395–396, 406–407**
- dataSource property, 187**
- Date attribute, 115**

Date data type

- data formatters, 314–315
- property lists, 395–396, 406

Date Style popup, 315**Debug area**

- breakpoints, 36
- components, 23, 31–34
- exception traces, 305
- hiding, 23–24, 40–41
- variables, 37, 143, 229, 493

Debug area actions, 42**DEBUG macro, 511****Debug navigator, 32**

- Game table, 191–192
- speed analysis, 238–239
- stack trace, 141–142

Debug Workflow menu, 518**Debug XPC services used by this application, 487****Debugging, 485**

- bindings, 307
- breakpoints. *See* Breakpoints
- controllers, 140–144
- dependent targets, 70
- frameworks, 388–393
- lldb command line, 491–493
- models, 131
- problem fixes, 39–42
- projects, 32–34
- QuickLook feature, 332–334
- scheme options, 485–488
- stepping through code, 37–39
- tips, 493–497
- tricks, 518
- unit testing, 229–232
- variables pane, 37–38

Debugging-symbol archive, 456**Decrease Deck Size, 470****Deepest Stack Libraries style, 469****Default attribute, 115****Default - Property List XML, 405****Deferred Mode, 472****#define directive, 58****Definitions, 514****Delegate design pattern, 136****delegate property, 187****Delete Rule for relationships, 118****Deleting**

- menus, 287
- projects, 24

Deny delete rule, 118**Dependencies**

- implicit, 69–70
- makefile goals, 431

Dependent targets, 68–70**Deployment Target field, 108****@deprecated keyword, 423****DERIVED_FILE_DIR, 448, 531****Derived files, 530–531****description method, 494****Descriptions**

- document localizations, 361
- exceptions, 142

Destination locations

- Doxygen, 426
- settings, 530–532

destination for xcodebuild, 445**Destination popup for frameworks, 386****Detail area**

- Call Tree listing, 241
- trace document window, 465–466

Detail Disclosure, 209**DEVELOPER_APPLICATIONS_DIR, 536****DEVELOPER_BIN_DIR, 536–537****DEVELOPER_DIR, 536**

- `/Developer` directory, 10
- `DEVELOPER_FRAMEWORKS_DIR`, 537
- Developer ID, 266–269
- Developer ID Application identity, 267
- Developer ID Installer identity, 267
- `DEVELOPER_LIBRARY_DIR`, 537
- Developer mode, disabling, 23
- Developer programs, 12–13, 255–256
- `DEVELOPER_SDK_DIR`, 537
- Developer Technical Support (DTS), 12, 542
- `DEVELOPER_TOOLS_DIR`, 537
- `DEVELOPER_USR_DIR`, 537
- Development process in iOS applications, 261
- Device Family setting, 107
- Devices settings, 260
- Diagnostics tab, 488
- Diagrams panel, 427
- Diamond badges, 222
- Dictionaries
 - object properties, 211
 - property lists, 395–396, 400–401, 407–408
- Direction popup, 178
- Directories
 - iOS projects, 111
 - localization, 337
- Directory I/O instrument, 475
- Disable Developer Mode, 23, 494
- Discard All Changes, 94, 121
- Discard Changes, 94
- Disclosure triangles in trace document window, 464
- Disk image (.dmg) files, 13–14
- Disk Monitor instrument, 477
- Disk space requirements, 10
- Dispatch instruments, 474
 - `dispatch_once` function, 169
- Display Brightness instrument, 478
- Display Pattern field, 312
- Display PostScript engine, 324
- Display requirements, 10
- Distribute for Developer ID, 267–268
- Distributed source-control systems, 79
- Distribution
 - builds, 269–273, 508–509
 - iOS applications, 261–262
 - .dmg (disk image) files, 13–14
- Do not show this message again, 34
- Dock, 17
- DOCSET BUNDLE ID setting, 428
- DOCSET FEEDNAME setting, 428
- DOCSET PUBLISHER ID setting, 428
- DOCSET PUBLISHER NAME setting, 428
- Docsets (documentation sets), 419–421, 424–425
 - Doxygen settings, 425–428
 - installing, 429
 - preparation, 425
- Document Extension setting, 279
- Document Outline sidebar, 163
- Document Outline view, 150
- Document types for localizations, 360–363
- Document Versions: Allow debugging when using document Versions Browser, 487
- Document window
 - autoresizing, 299–301
 - laying out, 295–301
- Documentation, 411
 - docsets, 419–421, 424–429
 - Documentation window, 415–419
 - downloading, 10–11
 - Help menu, 414–415

Open Quickly, 413–414
 Quick Help, 411–413, 421–424
Documentation and API Reference settings, 414
Documentation sets (docsets), 419–421, 424–425
 Doxygen settings, 425–428
 installing, 429
 preparation, 425
Documents
 application bundles, 373
 icons, 361–363
 OS X, 277–278
Dollar sign (\$) setting, 512
Dot panel in Doxygen, 428
dot tool, 425
Downloading
 docsets, 420–421
 packages, 14–15
 Xcode, 13–14
Downloads panel, 11, 15, 421
Doxygen generator, 421
 basic settings, 425–427
 comments, 423–424
 docset installation, 429
 docsets, 424–425
 expert settings, 427–428
 running, 428–429
Drive & Record, 470
DSTROOT, 531
DTPerformanceSession framework, 473
DTrace Data Import, 482
DTrace tool, 482
DTS (Developer Technical Support), 12, 542
DYLIB_INSTALL_NAME_BASE, 386
Dynamic libraries (.dylib), 385–386, 519
Dynamic loading, 52–53

E

@e keyword, 423
Edges for views, 178, 206
Edit Active Target, 463
Edit All in Scope, 57
Edit Breakpoint, 490–491
Edit Find Options, 84
Edit Instrument sheet, 481
Edit ‘Reads/Writes’ Instrument, 480
Edit Scheme, 70, 346
Editing
 build settings, 437–438
 property lists, 396–406
 view controllers, 136–138
Editing tab, 22, 28, 168
Editor area, 32, 153
Editor control, 42, 302
Editor menu, adjusting, 512
Editor Style control, 114
Editor table, 210–213
 passing data to, 213–215
 retrieving data from, 215–217
Editor view controllers, 210–213
Editors, 205
 Assistant. *See* Assistant editor
 Capabilities, 262–263
 Commit, 78–79
 Comparison, 93–96
 Data Model, 114
 linking, 208–209
 Merge, 91
 Project, 339
 Property List, 370, 399–406
 RTF, 342
 segues, 218
 static table cells, 209–210

Editors (continued)

- Target. *See* Targets and Target editor text, 545–546
- Version, 93
- @em keyword, 423**
- emacs text editor, 546
- Email Link, 419
- Embedded view controllers, 206–208
- en.lproj directory, 337–338
- Enable Developer Mode on this Mac? alert, 23
- Enable Developer Mode, 494
- Enable for Development button, 260
- ENABLE_NS_ASSERTIONS macro, 512
- Enable Performance Analysis popup, 488
- @encode directive, 234**
- Energy-impact report, 291–293
- Energy Usage instrument, 479
- Enterprise developer program
 - Apple developer programs, 256
 - build settings, 270–271
 - iOS distributions, 262
- Entities, 113**
 - models, 114
 - OS X applications, 282–286
- Environment settings, 528–529**
- Equality assertions, 234–235**
- Errors**
 - compiler, 29–30
 - debugging. *See* Debugging
 - displaying, 54, 56
 - unit testing, 229
- Errors Only, 449**
- Escape key shows code completions, 169
- Event Profiler instrument, 477
- Events, 460, 477
- EXC_BAD_ACCESS message, 32

- Exception breakpoints, 141
- @exception keyword, 422**
- Exceptions, 141–144**
 - assertions, 235
 - temporary, 265
- EXECUTABLE_FOLDER_PATH, 532
- EXECUTABLE_PATH, 386, 532
- existingPassersWithLastName function, 243–246, 286
- Expand Variables Based On popup menu, 486
- Expert tab in Doxygen, 425, 427
- Export Accounts, 261
- Export button in Documentation window, 419
- Export Developer ID-signed Application, 267, 387
- Export Items, 261
- Export Snapshot, 89
- Exported UTIs settings, 361
- expression, 492–493
- expression interpreter, 492
- Extended Detail view**
 - stack traces, 482
 - trace document window, 467
- Extensions**
 - classes, 124–126, 162, 189
 - document localizations, 360–361
- extern keyword, 514**

F

- F-keys, 39**
- Face to face resources, 544
- Family popup for labels, 159
- Fat (universal) binaries, 454
- Features, turning off, 21–22
- Fetches Properties table, 114
- fetchResultsController method, 137

- File Activity instrument, 475**
- File Attributes instrument, 475**
- File inspector tab, 65**
- File Locks instrument, 475**
- File Types column, 341**
- Files**
 - adding to targets, 65–68
 - configuration, 439–443
 - renaming, 514
 - searching, 514
 - states, 76–77
- File's Owner setting, 288, 304**
- Filesystem instruments, 475**
- Fill With Test Data, 291, 316**
- Filled Line Graph style, 469**
- Filling bindings, 301–307**
- fillWithData, 288–291**
- Filtering stack trace, 142**
- Find for property lists, 401, 403**
- Find and Replace, 85, 403**
- Find Implicit Dependencies, 70, 383**
- Find in Project, 87–88**
- Find in Workspace/Project, 403**
- Finder**
 - bundles, 367
 - command-line arguments, 347
 - docset versions, 421
 - instruments, 460
 - iOS apps, 369
 - localizations, 354
 - packages, 366
 - PNG files, 454
- Fink package manager, 549**
- First Responder, 288–289**
- Fix Issue, 107**
- Fix-it popover, 139–140**
- Fix Misplacement, 180**
- Flatten Recursion, 466**
- Folders**
 - new projects, 20
 - references, 512–513
- Folders: Create groups for any added folders, 124**
- Font field for labels, 159**
- Fonts & Colors panel, 32**
- Format menu, deleting, 287**
- Format specifiers in localizations, 358**
- Formats tab for localizations, 354**
- Formatters**
 - numbers, 313–314
 - strings and dates, 314–315
- Forums, 540–541**
- FOSS (free and open-source) software, 548**
- Foundation command-line program, 54**
- Fraction Digits setting, 314**
- frame in lldb, 492**
- Frames for labels, 180–181**
- Frameworks, 60, 381**
 - in applications, 386–387
 - debugging, 388–393
 - installing, 383–387
 - location, 385–386
 - populating, 382–383
 - targets, 381–383
- Frameworks directory, 369**
- FRAMEWORKS_FOLDER_PATH, 532**
- Frameworks groups in iOS projects, 109–110**
- Free and open-source (FOSS) software, 548**
- free function, 459**
- French localization, 338**
 - base, 338–339
 - process, 339–345
 - trying out, 345–347
- Function keys, 39**

G

Game Array controller bindings, 309–311

Game Center mediator, 262

Game table, 187

- data formatters, 314–315
- first run, 191–193
- Model-to-View support, 189–190
- outlets, 187–188
- protocol methods, 188–189
- prototype cells, 190–191
- table cells, 193–196

gameTableClicked action, 316–317

Garbage Collection instrument, 475

Gatekeeper, 12, 123, 266–269

gcc compiler, 54

GCC prefix, 533

GCC_ENABLE_CPP_RTTI, 434

GCC_ENABLE_OBJC_GC, 534

**GCC_PREPROCESSOR_DEFINITIONS,
533–534**

**GCC_PREPROCESSOR_DEFINITIONS_NOT_
USED_IN_PRECOMPS, 534**

GCC_TREAT_WARNINGS_AS_ERRORS, 534

GCC_VERSION, 533

GCC_VERSION_IDENTIFIER, 533

GCC_WARN_, 535

GCC_WARN_INHIBIT_ALL_WARNINGS, 534

General editor for iOS projects, 107

General settings

- automatic features, 22
- controllers, 139
- icons, 200, 359
- images, 200
- instruments, 471
- iOS projects, 107
- libraries, 69

Navigator detail, 56

registering apps, 258

GENERATE_DOCSET setting, 428

Generate Test Data build phase, 129

Generic apps, 262

Generic team provisioning profiles, 260

genstrings utility, 357–358

Gestures for navigation, 517

Get-file sheets, 110–111

Git version-control system, 25, 550

- OS X applications, 279
- repositories. *See* Repositories
- servers, 82
- Xcode with, 77–78

.gitignore files, 75

Global hot key combinations, 471

Goals for makefiles, 431

GPS instrument, 478

Grand Central Dispatch facility, 169

Graphics, 196

- assets catalog, 198–201
- icons and launch images, 201–202
- image views, 197–198
- table cells, 196–197

Graphics instruments, 475

Graphics package, 14

Graphing views, 325–328

Graphs in Debug navigator, 191–192

GraphViz package, 425

Gray Scale slider, 308

Group from Selection, 121, 150

Group popup, 26

GROUP, 528

Grouped style, 209

Groups

- bots, 507
- frameworks, 109–110

H

Hardware capabilities, 264
Hardware IO package, 14
HEADER_SEARCH_PATHS, 535
Headers, 29
 library targets, 68
 prefix, 60
Heads-up display (HUD) window, 163
Height setting for views, 178
Hello World project, 18
 building and running, 22–24
 creating, 18–21
 deleting, 24
Help
 application bundles, 374
 Help menu, 414–415
 lldb, 491–492
 Quick Help, 411–413, 421–424
help, 491
help breakpoint, 492
Help menu, 414–415
HFS+ filesystem, 524
Hide Missing Symbols, 466
Hide/Show Debug Area button, 37
Hide System Libraries, 243, 252, 466
Hide Toolbar, 494
Hiding Debug area, 41
Highlight style, 308
HOME, 528
Homebrew package manager, 549
Hooking up outlets, 164–165
Hopper Disassembler tool, 47, 547
Horizontal Center in Container, 182
Host a Git Repository, 502
Hosted repositories can be created by setting, 501
Hot key combinations, 471

HTML and Doxygen, 427–428
HUD (heads-up display) window, 163

I

@i keyword, 423
I/O Activity instrument, 475
IBAction type
 linking controls to actions, 165
 menu actions, 291
 table clicks, 315–316
 unit testing, 233
 unwind segues, 215–216
IBOutlet type
 array controllers, 302
 bindings, 307
 constraints, 176
 outlets, 161–165
 removing, 522
 view controllers, 154
ibtool tool, 455
iCloud capabilities, 262
.icns files, 359, 361, 363
Icons
 launch images, 201–202
 localizations, 359–363
.iconset directory, 361–363
iconutil tool, 363
Identifier popup, 206
Identifier setting in localizations, 360–361
Ignored file state, 76
Image sets, 198–199
Image Views, 197–198
Images. *See* Graphics
Images.xcassets catalog, 199
 contents, 362
 icons, 359

Images.xcassets catalog (*continued*)

- OS X applications, 280
- overview, 109

Implicit dependencies, 69–70**#import directive**, 58**Import Energy Diagnostics from Device**, 478**In-app purchases**, 263**In-house distributions**, 262, 270**In Project**, 514**In Workspace**, 514**#include directive**, 58, 442**Include Spotlight Importer**, 279**Increase Deck Size**, 470**Indentation tab**, 22, 28**Indexed setting**, 244**Indexing**, 57–58**Individuals in Apple developer programs**, 256**Info.plist file**, 408

- application keys, 371–373
- background modes, 263
- builds, 270–272, 456
- bundles, 367, 369–370, 374, 379
- gloss effect, 201
- localizations, 353–354, 360, 370
- OS X applications, 280
- packages, 365–366
- property lists, 397, 399
- settings, 535–536
- signatures, 262

Info tab

- allocations, 248
- application keys, 373–374
- builds, 438–439, 441
- Info.plist file, 369
- localizations, 338, 360, 362
- property lists, 382, 396–397
- Quick Help, 412

schemes, 485–486

tests, 222, 504

InfoPlist.EXPAND.BUILD.SETTINGS, 536**INFOPLIST.FILE**, 270, 370, 535**INFOPLIST.OUTPUT.FORMAT**, 407, 536**InfoPlist.strings**, 340, 345, 370**Inherited setting**, 512**initialize function**, 170**initWithFrame method**, 332, 334**INPUT.FILE.BASE**, 448**INPUT.FILE.DIR**, 448**INPUT.FILE.NAME**, 448**INPUT.FILE.PATH**, 448**Input/output instruments**, 475**Insert Pattern**, 85**insertNewObject method**, 138**Inspection Range control**, 251, 463–464**install action for xcodebuild**, 444**INSTALL.PATH**, 386**Installing**

- docsets, 429
- frameworks, 383–387
- Xcode, 10–11

installsrc action, 444**Instruction**, 495**Instrument Specific**, 463**Instruments**, 473

- behavior, 474
- configuration, 468–470
- Core Data, 474
- custom, 480–482
- Dispatch, 474
- filesystem, 475
- Garbage Collection, 475
- input/output, 475
- iOS energy, 478–479
- Library window, 467–468
- master tracks, 476

- memory, 476–477
- overview, 459–460
- recording, 470–472
- running, 460–461
- saving and reopening, 472–473
- speed analysis, 240–243
- system, 477–478
- templates, 482–483
- threads/locks, 479
- trace, 479–480
- trace document window, 461–467
- tricks, 518
- UI automation, 480
- user interface, 480
- Integration, 499**
 - bots, 503–508
 - building for distribution, 508–509
 - Xcode Server, 500–503
- Intentions for views, 157**
- Inter-App Audio service, 263**
- Interface Builder**
 - autolayout, 174
 - class names, 135
 - constraints, 175–176, 179
 - control actions, 291
 - labels, 180–181
 - localizations, 341
 - outlets, 161, 165, 188
 - property editing, 289
 - table views, 296, 298
 - view controllers, 148–151, 154, 216
- Interface Builder tab, 42, 150**
- Intermediate compiler products, 58–62**
- Interpreted languages, 54**
- Intrinsic sizes, 181**
- Invert Call Tree, 242, 252, 466**
- iOS**
 - application bundles, 371–373, 376–379
 - application submissions, 261–262
 - autolayout. *See* Autolayout
 - capabilities, 262–263
 - controllers. *See* Controllers
 - energy instruments, 478–479
 - measurement and analysis. *See* Measurement and analysis
 - model. *See* Models
 - MVC design pattern, 103–106
 - as packages, 369
 - porting from, 282–286
 - provisioning. *See* Provisioning
 - scheme options, 487–488
 - starting projects, 106–108
 - table cells. *See* Tables and table cells
 - templates, 108–110
 - unit testing. *See* Unit testing
 - view controllers. *See* View controllers
- “iOS Debugging Magic (TN2239)”, 518**
- iOS Enterprise developer program**
 - Apple developer programs, 256
 - build settings, 270–271
 - iOS distributions, 262
- iOS icon is pre-rendered, 201**
- iOS Simulator**
 - layouts, 156
 - limitations, 191, 238
 - memory, 248, 476
 - speed analysis, 238
 - starting, 110
 - templates, 482–483
 - tests, 227
- IPHONEOS_DEPLOYMENT_TARGET, 535**
- ISO-standard languages, 337**
- Issue Navigator Detail, 56**

Issues: Show live issues, 54
Items of New Constraints, 161, 178–179

J

JavaThread instrument, 479
Join a Program, 256
Jump bars

- Assistant editor, 315
- description, 150–151
- object controllers, 302

K

Kaleidoscope tool, 547–548
Keep in Dock, 461
Key Bindings panel

- controllers, 139
- Preferences window, 513

Key Equivalent field, 288
Key paths, 334
Key-Value Coding (KVC), 167, 211, 301, 334
Key-Value Observing (KVO), 301–302
Key-value pairs

- localizations, 347
- property lists, 400–401

Keyboard panel for shortcuts, 39, 376
Keyboard Shortcuts tab, 376
Keychain sharing, 263
Keys for applications, 371–376
KVC (Key-Value Coding), 167, 211, 301, 334
KVO (Key-Value Observing), 301–302

L

Labels

- building views, 158–160

- constraints, 179–185
- tags, 194
- Language & Region panel, 337**
- Language & Text panel, 354**
- Language tab, 337**
- Languages, 337**
- Launch behavior for bundles, 374–375**
- Launch due to a background fetch event, 488**
- Launch images, 201–202**
- Launch Services, 360**
- Layering NSControllers, 307–315**
- Laying out document window, 295–301**
- Layout guides for views, 157**
- Layout rectangles, 308**
- LD_DYLIB_INSTALL_NAME, 386**
- Leading edges of views, 178, 206**
- Leaks instrument, 476, 518**
- Left-side group for labels, 158–160**
- .Jemom files, 448**
- Levels for build settings, 435–437**
- libcrypto API, 9**
- Libraries**
 - adding, 69–70
 - dynamic, 52–53
 - frameworks, 385–386
 - instruments, 467–468, 474
 - Interface Builder, 296
 - object files, 51
 - static, 63, 519
 - targets. *See* Library targets
 - trace document window, 463
- /Library/Developer directory, 11–12**
- Library navigator, 420**
- Library palette, 474**
- LIBRARY_SEARCH_PATHS, 535**
- Library targets, 63**
 - adding, 63–65
 - debugging, 70

- dependent, 68–70
- description, 64–65
- headers, 68
- membership, 65–68
- Library window, 467–468**
- Licenses for MonoTouch, 554**
- Line Break popup, 314–315**
- Line Graph style, 469**
- Line wrapping: Wrap lines to editor width, 22**
- Link Binary With Libraries build phase, 50, 52, 61, 110, 432**
- Linking and linkers**
 - editing, 51
 - editors, 208–209
 - process, 50–52
 - tricks, 519
- Lion, 264**
- lipo tool, 454**
- Live Autoresizing, 309**
- lldb debugger**
 - command line, 491–493
 - overview, 389–392
- LLDB Quick Start Guide, 493***
- .lldbinit files, 493**
- llvm library, 53–54, 140**
- Loading**
 - dynamic, 52–53
 - MPRDocument data, 289–293
- loadView method, 148**
- Local analysis, 54–56**
- Local remote repositories, 80–82**
- Local variables, 37**
- Locales, 340**
- Localizable.strings file, 357–358**
- Localizations, 337**
 - adding, 338–347
 - application bundles, 373
 - base, 338–339
 - bringing files into, 352
 - document types, 360–363
 - icons, 359–363
 - Info.plist, 353–354, 370
 - MainMenu.xib, 347–351
 - overview, 337–338
 - process, 339–345
 - strings, 355–358
 - trying out, 345–347
- localizedStringForKey method, 356**
- Locations**
 - Doxygen, 426
 - frameworks, 385–386
 - settings, 530–532
 - workspaces, 522
- Locks instruments, 479**
- Log Message, 490**
- Log navigator for builds, 449**
- Log view for Comparison editor, 93, 95–96**
- Logic tests, 232**
- Login button.png, 368**
- Logs**
 - vs. breakpoints, 488–489
 - builds, 448–449
- Look up API Documentation, 467**
- lproj system, 337–338**
- LSApplicationCategoryType key, 374**
- LSBackgroundOnly key, 374**
- LSEnvironment key, 375**
- LSFileQuarantineEnabled key, 375**
- LSFileQuarantineExcludedPathPatterns key, 376**
- LSGetAppDiedEvents key, 375**
- LSMinimumSystemVersion key, 375**
- LSMinimumSystemVersionByArchitecture key, 375**

LSMultipleInstancesProhibited key, 375
LSRequiresiPhoneOS key, 376
LSUIElement key, 375
LSUIPresentationMode key, 375

M

Mac App Store. *See* **App Store**
Mac Developer identity, 267
Mac Installer Package, 388
Mac OS X. *See* **OS X**
MAC_OS_X_PRODUCT_BUILD_VERSION, 529
MAC_OS_X_VERSION_ACTUAL, 528–529
MAC_OS_X_VERSION_MAJOR, 529
MAC_OS_X_VERSION_MINOR, 529
Machine instructions, 49
MacOS directory, 369
MacPorts package manager, 549
Mailing lists, 541–542
main.m file, 280
Main.storyboard file, 108
MainMenu.xib file, 287

- localizations, 347–351
- OS X applications, 280

Makefile goals, 431
malloc function, 459
MallocDebug application, 459
Manage Flags, 477
Manage PM Events, 477
Manage Schemes editor, 223

- bots, 504
- listing schemes, 223
- new schemes, 272

Managed-object classes, 113, 120

- creating, 120–124
- extending, 124–126
- source control and product files, 128–131
- test data, 126–128

Managed Object Context binding, 303–304
Mark Heap, 518
Mark Selected Files As Resolved, 77
Master branches in version control systems, 97
Master-Detail Application template, 152
Master track instruments, 476
Mavericks, 10

- arguments, 307
- command-line tools, 11
- energy-saving strategies, 293
- garbage collection, 475
- state restoration feature, 291
- support, 14
- system libraries, 61

Mavericks Server, 74, 502
Maximum attribute, 115, 314
Measurement and analysis, 237

- memory, 247–253
- speed. *See* **Speed**

Meetings, 544
Membership, target, 65–68
Memory, 247–248

- allocations, 248–250
- bar graphs, 291
- instruments, 476–477
- object type focus, 250–252
- problems, 518
- RAM, 49
- reports, 291–293
- requirements, 10
- transients, 252–253

Memory Monitor instrument, 477
Menus, wiring, 287–288

- First Responder, 289

- MPRDocument data, 289–293
- targets and actions, 288
- Merge editor, 91**
- Merge from Branch, 97**
- Merge into Branch, 97**
- Merges in version control systems, 83–93**
- Messages**
 - analysis, 56
 - Documentation window, 419
 - logs, 490
 - Objective-C compilers, 524
- Metadata in Git, 76**
- Method names, refactoring, 134**
- MIME Type setting, 360–361**
- Min Length setting, 117**
- Mini instruments, 471–472**
- Minimum attribute, 115, 314**
- missing-braces-and-parentheses warning, 140**
- MKDirectionsApplicationSupportedModes key, 379**
- Modal scenes, 205–210**
- Mode settings**
 - Doxygen, 426–427
 - object controllers, 302
- Model controllers in OS X applications, 280**
- Model Key Path setting, 304–305**
- Model-to-View support, 189–190**
- Model-View-Controller (MVC) design pattern, 103–104**
 - controllers, 106
 - models, 104
 - views, 104–106
- Models**
 - attributes, 114–117
 - debugging, 131
 - entities, 114
 - implementing, 113
 - managed-object classes. *See* Managed-object classes
 - OS X applications, 281–286
 - relationships, 117–119
- Modern bundles, 367**
- Modified file state, 76–77**
- module.map file, 61**
- Modules, 60–62**
- Modules extension, 52**
- mogenerator tool, 123–124, 548**
- MonoTouch framework, 553–554**
- More Developer Tools, 15**
- motion tool, 553**
- Mountain Lion, 8, 10**
 - command-line tools, 11
 - Gatekeeper, 266
- Mouse pointer variables, 37**
- Move Breakpoint To, 493**
- MPRDocument class**
 - loading data into, 289–293
 - OS X applications, 279–280
- MPRDocument.xcdatamodeld file, 280**
- MPRDocument.xib file**
 - attributes, 308
 - bindings, 315
 - compiling, 280
 - object controllers, 302
 - table view, 296
- MPRGameViewController.xib file, 318, 355**
- MPRPassCompletionView class, 324–325**
- MPRPasserGraphController class, 323–325, 328–332, 338, 352**
- Multithreading, 169**
- MVC (Model-View-Controller) design pattern, 103–104**
 - controllers, 106
 - models, 104
 - views, 104–106

N

Name labels, 158–159

Names

localizations, 360

product, 279

refactoring, 134–136

nan (not a number), 35

NATIVE_ARCH, 533

NATIVE_ARCH_32_BIT, 533

NATIVE_ARCH_64_BIT, 533

Navigation panel for gestures, 517

Navigators, 20

Breakpoint, 141

Debug, 191–192

detail settings, 56

Documentation window, 415

Issue navigator, 29

Library, 420

Log, 449

Project, 36

Symbol, 57

NDEBUG macro, 511

Net resources, 540–543

Network Activity instrument, 479

Network Activity Monitor instrument, 477, 479

Network capabilities, 264

New Branch, 97

New File assistant, 26–27

New Folder

Doxygen, 426

subclasses, 121

New Project assistant, 18–19, 26

iOS, 106

OS X, 278

New Scope, 514

New Tab, 42, 150

New Target assistant, 63, 381

.nib files, 135

Nil pointers, 169

nm tool, 60

No Access, 264

No Action delete rule, 118

No Selection Placeholder field, 312

Normalizing entities, 282

Not a number (nan), 35

not-enough-fields.csv file, 226

“not key value coding-compliant” exceptions, 307

NS_BLOCK_ASSERTIONS macro, 511–512

NSAlert, 357

NSAppleScriptEnabled key, 376

NSApplicationMain function, 280

NSApplicationShowExceptions setting, 497

NSArray class, 143

NSArrayController class, 302–303

NSBindingDebugLogLevel setting, 307

NSBundle class, 337, 356

NSCoder Night meetings, 544

NSControl class, 315

NSController class, 303, 307–315

NSDateFormatter class, 168

NSError class, 54–55, 228

NSFetchedResultsController class, 109, 136, 187, 189

NSFileWrapper class, 367

NSHumanReadableCopyright key, 372, 374

NSLocalizedString class, 357

NSLog function, 489–490

NSMainNibFile key, 372

NSManagedObject class

MVC model, 103

subclass creation, 120–121

NSManagedObjectCollector class, 303

NSNumberFormatter class, 168
NSObject class, 103
NSObjectController class, 318
NSPopover, 323
NSPrincipalClass key, 371
NSRTFDPboardType file type, 366
NSScrollView, 523
NSServices key, 376
NSSortDescriptor class, 137
NSString class, 168
NSSupportsSuddenTermination key, 375
NSTableView class, 315
NSTextView, 523
NSViewController class, 318
NSZombieEnabled setting, 488
Null Placeholder field, 312
Nullify delete rule, 118
Numbers
 data formatters, 313–314
 property lists, 395–396, 406

O

.o files, 432
objc-language list, 542
Object allocations by class, 459
Object controllers, 302–304
Object controllers chain, 310–311
OBJECT_FILE_DIR, 532
Object files, 50–51
Object Graph instrument, 476
Objective-C
 alternatives, 553–554
 class names, 120
 compiler messages, 524
OBJROOT, 531
Omni Group, 542
Open in instruments, 473
Open Anyway, 425
Open Keyboard Shortcut Preferences, 471
Open Link in New Tab, 417
Open Other, 18
Open Quickly dialog, 413–414
Open Recent, 24
OpenCL facility, 53
OpenGL Driver instrument, 475
OpenGL ES Analyzer instrument, 475
OpenGL ES Driver instrument, 475
OpenGL ES Frame Capture, 488
Optimization
 compiler, 48–49
 speed, 243–247
 tricks, 519–520
Option key, 517
Optional for libraries, 69
Options for trace document window, 463
Options panel, 19
Options tab
 schemes, 486–487
 state-restoration feature, 291
Ordered lists, 395–396, 407
Ordered for relationships, 117
Organization Name setting
 iOS projects, 106
 new projects, 20
 OS X projects, 279
Organizations in Apple developer programs, 256
Organizer window
 derived files, 531
 snapshots, 89
 trash, 24
 workspaces, 521
Orientation setting, 179
Origin control for views, 324
-Os optimization, 519

OS X, 275, 277

- autolayout, 318–320
- bindings. *See* Bindings
- bundles. *See* Bundles
- capabilities, 262–263
- command-line arguments, 347
- custom views. *See* Custom views
- entities, 282–286
- frameworks. *See* Frameworks
- goals, 277–278
- localizations. *See* Localizations
- models, 281–286
- porting from iOS, 282–286
- property lists. *See* Property lists
- running, 321
- sandboxing, 264–266
- starting applications, 278–281
- wiring menus, 287–293

“OS X Debugging Magic (TN2124)”, 518

OSAScriptingDefinition key, 376

OTHER.CFLAGS, 534–535

OTHER.CODE.SIGN.FLAGS, 530

otool tool, 60

Outlets

- building views, 153–154
- code completion and snippets, 168–170
- connections, 164–168
- hooking up, 164–165
- overview, 161–164
- table view, 187–188

Output panel in Doxygen, 427

Overlay for instruments, 470

P

@p keyword, 423

Package managers, 548–549

Packages, 365

- downloading, 14–15
- RTFD, 365–367

PaintCode tool, 548

@param keyword, 422

Passer Array Controller, 311–312

Passer controller bindings, 309–310

Passer ratings project overview

- building, 29–31
- controllers, 144–145
- creating, 25–29
- debugging, 32–34
- running, 31–32
- test case, 35

Passer table, binding, 311–312

PasserGroup framework, 382–383

Passing data to editor, 213–215

Paste for dictionaries, 401

Pboard Types setting, 361

.pch files, 60, 452

Peak Graph style, 469

Performance bar charts, 32

Performance optimization

- compiler, 48–49
- speed, 243–247
- tricks, 519–520

Permissions settings for bots, 505

Persistent State: Launch application without state restoration, 487

Phases, build, 432–433

PhoneGap framework, 551

Pin popover, 177–178, 184

Pixels for icons, 201

Plain style, 209

Planning apps, 103–106

platform in lldb, 492

PLATFORM_NAME, 528

- Playback head in trace document window, 464**
- Player billboard**
 - label constraints, 179–185
 - size constraints, 175–179
- PLIST_FILE_OUTPUT_FORMAT, 407**
- Plists. *See* Property lists**
- Plugins, 334**
- plutil tool, 406–407**
- po command, 494–495**
- Point Graph style, 469**
- Pointers**
 - Cocoa programming, 54
 - nil, 169
- Points for icons, 201**
- Popovers**
 - bindings, 315–320
 - Quick Help, 412–413
 - segues, 218
 - variable values, 37
- Populating frameworks, 382–383**
- Portals for iOS, 261**
- Porting from iOS, 282–286**
- POSIX working directory, 487**
- #pragma mark directive, 188**
- PRAppDelegate class, 108**
- PRDetailViewController class, 109, 147**
- Precompilation, 60–62**
- Preferences window**
 - Apple ID, 256
 - archives, 261
 - automatic features, 22
 - behaviors, 40–42
 - bindings, 139
 - bots, 504
 - code completion, 28, 168
 - code–folding ribbon, 515–516
 - controllers, 139
 - Developer ID, 266
 - docsets, 420
 - downloads, 11, 15
 - fonts, 32
 - indentation, 28
 - instruments, 471
 - key equivalents, 513
 - navigational gestures, 517
 - Navigator detail, 56
 - remote repositories, 79, 82, 84
 - snapshot locations, 522
 - source trees, 537
 - team membership, 258–259
 - version control, 74
 - warnings and errors, 54
- Prefix files, 60, 280**
- Prefix headers, 60**
- Prepares Content, 302, 310**
- Preprocessing xcconfig files, 442–443**
- Preprocessors, 58–59**
- Prerelease versions, 13**
- Preview assistant, 155**
- Preview, 88–89**
- Preview view, 156–157**
- PRGameListController class, 150, 165–168, 189–190, 194**
- Priority setting for centering, 183**
- Private keys for certificates, 261**
- Private role, 68**
- PRMasterViewController class, 109, 133–146**
- Pro Git version control system, 549**
- Probes, 481**

- process in lldb, 492**
- Process instrument, 477**
- Processor requirements, 10**
- Product files in managed-object classes, 128–131**
- Product Name setting, 20, 278–279**
- Profiles**
 - applications, 240
 - provisioning, 257–260
- Program members, 259**
- PROJECT_DIR, 530**
- Project editor**
 - library targets, 63–64
 - localizations, 339
- PROJECT_FILE_PATH, 530**
- PROJECT_NAME, 528**
- Project navigator, 36**
- project for xcodebuild, 444**
- Project role, 68**
- PROJECT, 528**
- Projects list for builds, 432**
- Projects organizer for workspaces, 521**
- Projects overview**
 - building, 22–24, 29–31
 - creating, 18–21, 25–29
 - debugging, 32–34
 - deleting, 24
 - Doxygen settings, 425–428
 - running, 22–24, 31–32
 - templates, 108–110
- Projects panel**
 - derived files, 531
 - snapshots, 89
- Properties**
 - custom views, 334–336
 - entities, 113
- Property List editor, 370**
 - limitations, 404–406
 - working with, 399–404
- Property lists, 395**
 - binary, 407
 - data types, 395–396, 406–407
 - editing, 396–406
 - specialized, 407–408
 - text, 406–407
- Protecting assets, 261**
- @protocol, 325**
- Protocol methods, 188–189**
- Prototype cells, 190–191, 193**
- Provide Feedback link, 421**
- Provisioning, 255, 257**
 - asset protection, 261
 - capabilities editor, 262–263
 - distribution builds, 269–273
 - Gatekeeper and Developer ID, 266–269
 - OS X sandboxing, 264–266
 - profiles, 257–260
 - registering apps, 258–260
 - submitting applications, 261–262
- PROVISIONING_PROFILE, 270, 530**
- PRPasser class, 187–188, 192, 208–210**
- PRPasserEditController class, 205, 210, 213–214**
- PRPasserEditTableController class, 208, 210**
- PRPasserListController class, 150, 152, 187, 208, 210, 214–217**
- PRTeam class, 283**
- Public role, 68**
- Pull, 90**
- Push, 83, 90**
- Push segues, 152**
- Pushing to remote repositories, 83**
- pwrite function, 481**

Q

- Quick Help facility, 411–413**
 - comment syntax, 422–424
 - generating, 421–422
- Quick Help for Selected Item, 412, 415**
- QuickLook feature, 332–334**
- Quit**
 - lldb, 491
 - OS X, 33
- Quit Xcode, 24**

R

- Raises For Not Applicable Keys, 304**
- rake, 553**
- RAM, 49**
- RatingTest class, 230**
- Read Access, 264**
- Read/Write Access, 264**
- Reads/Writes instrument, 475, 480–481**
- Recent for builds, 449**
- Record**
 - instruments, 473
 - trace document window, 463
- Record Options, 472–473**
- Recording instruments, 470–472**
- Rectangles**
 - bounds, 308
 - layout, 160, 308
- Refactoring feature, 57**
 - class names, 134–136
 - method names, 134
- Reference Language column, 340**
- Reference URL setting, 361**
- References**
 - folders, 512–513
 - repositories, 80
- Region setting, 354**
- Registered developers, 13, 259**
- Registering**
 - apps, 258–260
 - team membership, 258
- Regular expressions**
 - refactoring method names, 134
 - searches, 85
 - traps, 523
- Relationships, 113–114, 117–119**
- Relative to Group, 513**
- Relaunch, 354**
- Release build configuration, 452**
- Release Notes section, 415**
- Remote repositories, 79–83**
- Remotes tab, 82**
- Removing**
 - breakpoints, 36
 - Xcode, 11–12
- Renaming service, 514**
- Renaming symbols, 133–136**
- Rentzsch, Jon “Wolf”, 123**
- Reopening instruments, 472–473**
- Replace All in File, 84–85**
- Replace segues, 218**
- Repositories**
 - cloning, 79
 - remote, 79–83
 - settings, 503
 - turning on, 503
 - Xcode Server, 79–80, 501–502
- Repositories tab, 500**
- Required for libraries, 69**
- Requirements, 10**
- Reset to Suggested Constraints, 180**
- resizableImageWithCapInsets method, 200**
- Resolve Auto Layout Issues menu, 206**
- Resource forks, 365**

Resource Manager, 365**Resources**

- books, 539–540
- Developer Technical Support, 542
- face to face, 544
- Net, 540–543
- sites and blogs, 542–543
- software, 544–554

Resources directory, 369**Responder chains, 288****Retrieving data from editor, 215–217****@return keyword, 422****Return Value, 493****Reveal in Library, 417****Rich text file directory (RTFD) package, 365–367****Right-side group for labels, 158–159****Role setting for localizations, 360****Root view controller segues, 152****Routing App Coverage File, 488****Row Height setting for table cells, 194****RTF editor for localizations, 342****RTFD (rich text file directory) package, 365–367****RubyMotion framework, 553****Rules, build, 446–448****Run Browser, 463****Run scheme editor, 485****Run Script editor, 127****Running**

- bindings, 305–307
- bots, 506
- Doxygen, 428–429
- instruments, 460–461
- OS X applications, 321
- projects, 22–24, 31–32
- tests, 227–228

S

sample-data.csv file, 128–130, 227, 237, 252, 289, 291**Sampler instrument, 477–478****Sandboxing**

- benefits, 265
- disadvantages, 266
- OS X, 264–266

Save as Template, 472**Save as Workspace, 520****Save-file dialog for targets, 67****Saving instruments, 472–473****Scan recursively, 426****scanf function, 32, 47–48****Scenes**

- modal, 205–210
- view controllers, 148–151

Schedules

- bots, 504
- instruments, 479

Scheme control, 68**Scheme editor, 70**

- binding debugging, 307
- instrument templates, 241
- state-restoration feature, 291
- tests, 222

scheme for xcodebuild, 445**Schemes**

- bots, 503
- options, 485–488

Scopes, defining, 514–515**Scroll View, 324****SDKROOT, 530****SDKs (software development kits), 9**

- build settings, 441–442
- iOS projects, 108

- Search Documentation for Selected Text section, 415**
- Search in help, 414**
- Search paths for settings, 535**
- Searches**
 - Documentation window, 417–419
 - files, 514
 - trace document window, 463
 - version control, 84–85
- Security & Privacy panel, 425**
- @see keyword, 423**
- Segues**
 - passer list, 208
 - types, 218
 - unwind, 215–216
 - view controllers, 148–151
 - views, 152
- Select and Edit, 183**
- Selective commits, 85–87**
- sender method, 209, 213**
- Separate by Category, 465–466**
- Separate by Thread, 466**
- Services menu, 471**
- Settings tab**
 - repositories, 503
 - Xcode panel, 500
- Shadow Offset, 159**
- Shadows for labels, 159**
- SHALLOW_BUNDLE, 532**
- Share Breakpoint, 493**
- Shared Memory instrument, 476**
- Shared User Defaults Controller, 304**
- Shift key, 517**
- Shortcuts**
 - function keys, 39
 - instruments, 471
 - lldb, 493
 - Show All Results, 417
 - Show Bounds/Layout Rectangles, 176
 - Show Bounds Rectangles, 308
 - Show: Code folding ribbon, 22, 515
 - Show Definitions, 438, 528
 - Show environment settings in build log, 527
 - Show Find Options, 522
 - Show Group Banners, 468
 - Show/Hide...debugger, 40
 - Show/Hide...navigator, 40
 - Show HTML output, 429
 - Show In Finder, 24, 519
 - Show Layout Rectangles, 160, 308
 - Show live issues, 22, 139
 - Show navigator, 42
 - Show Obj-C Only, 466
 - Show Package Contents, 10, 366, 369, 421
 - Show Raw Values & Keys, 408
 - Show Resize Knobs, 181
 - Show Setting Names, 438, 528
 - Show Setting Titles, 528
 - Show Slicing, 201
 - Show tab named, 42
 - Show Vertical Scroller, 324
 - Signals from exceptions, 141
 - Signatures in iOS provisioning, 257
 - Signing identities, 257, 458
 - SimpleCSVFile, 224
 - Simulate Document, 299
 - Simulate Location, 39
 - Simulated Metrics attribute, 179
 - Sites, 542–543
 - 64-bit applications, 52–53
 - Size and Size Inspector, 155, 157
 - columns, 297
 - constraints, 175–179
 - document window, 299–301
 - labels, 183

Size and Size Inspector (*continued*)

- table cells, 194
- views, 324, 716

Skip Install, 387**Sleep/Wake instrument**, 479**SMAuthorizedClients key**, 376**SMPrivilegedExecutables key**, 376**Snap to Guides**, 309**Snap Track to Fit**, 248**Snapshot Now**, 477**Snapshots**

- projects, 89
- VM Tracker, 477

Snippets, 169–170**Software development kits (SDKs)**, 9

- build settings, 441–442
- iOS projects, 108

Software resources, 544–545

- AppCode, 550–551
- assessment, 552
- Cocoa alternatives, 551–552
- helpers, 546–548
- package managers, 548–549
- text editors, 545–546
- version control, 549–550

sortDescriptors property, 321**Source code**

- description, 45
- Doxygen, 426
- property lists, 404

Source control. *See* Version control systems**Source Control menu**, 75–76, 80, 82**Source Control**, 522**Source files with names matching**, 447**Source Locations settings**, 530**Source trees settings**, 537**Sources & Binaries**, 425**SourceTree version control system**, 550**Specialized property lists**, 407–408**Speed**, 237–238

- Debug navigator, 238–239
- instruments, 240–243
- memory, 247–253
- optimization, 243–247

Spin Monitor instrument, 478**Splash screens**, 201**Springs**, 300**SQLite**, 113**SRROOT**, 127–128, 444, 530**Stack Libraries style**, 469**Stack traces**, 32

- displaying, 141–142
- Extended Detail view, 482
- filtering, 142
- trace document window, 467

Stacked for instruments, 470**Staged file state**, 76–77**Standard Windowing**, 518**Starting**

- iOS projects, 106–108
- Xcode, 17–18

State-restoration feature, 291**States of files**, 76–77**Static libraries (.a)**, 63, 519**Static table cells**, 205, 209–210**Statistics to Graph settings**, 470, 482**Step Into (F7)**, 39, 495–496**Step Out (F8)**, 39**Step Over (F6)**, 39, 495–496**Stepping through code**, 37–39, 495–496**Stop**

- debugging, 33
- instruments, 242, 470
- iOS, 145

Storyboard editor for segues, 218

- .storyboardc files, 135**
- Storyboards for view controllers, 148–151**
- Strings and .strings files**
 - builds, 454
 - data formatters, 314–315
 - localizations, 347–349, 355–358
 - property lists, 395–396, 406–407
- STRINGS_FILE_OUTPUT_ENCODING, 536**
- Structure**
 - application bundles, 371–372, 374, 376–377
 - builds, 431–434
- Structured directory trees, 367**
- Struts, 300**
- Style settings**
 - buttons, 206
 - dates, 315
 - instruments, 469
 - models, 114
 - table cells, 209
- Sublime Text 2 text editor, 546**
- Submit to the Mac App Store Package, 387**
- Submitting iOS applications, 261–262**
- Subpath field for frameworks, 386**
- Sudden Termination instrument, 474**
- Suggest completions while typing, 168**
- Summary tab**
 - property lists, 397
 - targets, 369, 374
- Supporting Files group, 109**
- Suppressing warnings, 145**
- Switch-Branch sheet, 98**
- Switch to Branch, 97–98**
- Symbol navigator, 57**
- Symbols, 49**
 - renaming, 133–136
 - tokens, 54

- SYMROOT, 531**
- Syntax-aware indenting settings, 22, 28**
- System instruments, 477–478**
- System Calls instrument, 479**
- system keychain, 509**
- SYSTEM_LIBRARY_DIR, 536**
- System Preferences application**
 - function keys, 39
 - gestures, 30, 417
 - instruments, 471, 476
 - localizations, 337, 345, 354
 - security, 425
 - services, 376

T

- Table of contents sidebar, 415–416**
- Tables and table cells, 187**
 - custom, 193–196
 - graphics. *See* Graphics
 - modal scenes, 205
 - OS X, 296–299
 - outlets, 187–188
 - prototype, 190–191
 - static, 205, 209–210
 - table views, 136–137, 161, 207
- tableView property, 187**
- Tabs**
 - creating, 150
 - Documentation window, 417
 - switching, 42
- Tags**
 - labels, 194
 - version control, 523
- TARGET_BUILD_DIR, 531**
- target in lldb, 492**
- target for xcodebuild, 444**

Targeted for iPad, 148**Targets and Target editor**

- ad hoc variants, 272
- asset-catalog file, 200
- build phases, 50, 432–433
- build rules, 447
- build settings, 434–435
- bundles, 523
- capabilities, 264
- code size, 511
- components, 63–64
- configuration files, 439, 441
- dependencies, 70
- device families, 377
- displaying, 222
- frameworks, 381–383, 386
- icons, 200, 272, 359–360
- images, 200, 378
- Info.plist file, 280, 369–370, 372–374, 382
- instruments, 468
- iOS projects, 107–108
- levels, 436–437
- libraries. *See* Library targets
- localizations, 362
- new projects, 26–27
- packages, 370
- product names, 279
- property lists, 396–397
- provisioning profiles, 258
- Quick Help, 412, 422
- registering apps, 258
- trace document window, 462–463
- wiring menus, 288

Team Admins in Apple developer programs, 256

Team Agents in Apple developer programs, 256

Team array controller, 302–303, 310–311

Team class, 283

Team Members in Apple developer programs, 256, 258

Team popup for iOS projects, 107

Team Provisioning Profiles, 260

Team table for bindings, 305

tearDown method, 224

Templates

- instruments, 241, 472, 482–483
- iOS projects, 108–110

Temporary exceptions, 265

Terminal application, 24

test action for xcodebuild, 444

Test data for unit testing, 226–227

Test navigator, 222–223

Test suites, 221

Test tab for bots, 508

testCalculation method, 231

testExample method, 224

Testing

- unit. *See* Unit testing
- views, 170–171

TestKit assertions, 233–235

testTooManyFieldsError method, 225, 228–229

Text

- containers, 183
- property lists, 395–396, 406–407

Text Color control, 159

Text Editing panel, 22, 516

Text editors, 545–546

TextEdit application, 365–366

TextMate 1.5 text editor, 545

TextMate 2 text editor, 545–546

TextWrangler text editor, 545

Third-party package managers, 549

3rd Party Mac Developer Application identity, 267
3rd Party Mac Developer Installer identity, 267
32-bit applications, 52–53
Thread
 in debugging, 495–496
 lldb, 492
Threads instruments, 479
Time Profilers for instruments, 240–241, 244, 469, 471, 478
Titanium API, 551–552
Titles
 buttons, 206
 columns, 298
 menu items, 288
@todo keyword, 423
Toggle Instruments Recording, 471
Tokens, 54
too-many-fields.csv file, 226
Toolbars
 modal scenes, 205–206
 trace document window, 462–464
Tools in Interface Builder, 149–151
Top Layout Guide, 178, 206
Trace Call Duration, 467, 482
Trace document window, 461–462
 Detail area, 465–466
 Extended Detail area, 467
 toolbar, 462–464
 Track area, 464–465
Trace Highlights, 467
Trace instruments, 479–480
Track area in trace document window, 464–465
Track Display, 470
Trailing edges in views, 178, 206
Transcripts for builds, 450–458

Transient attribute, 115
Transients, cleaning up, 252–253
Traps, tricks for, 522–524
Trash, 24
Tricks
 Assistant editor, 515–517
 building, 518–520
 code-folding ribbon, 515–516
 general, 511–515
 instruments and debugging, 518
 traps, 522–524
 workspaces, 520–522
Truncate Middle, 314–315
Truncation, 313
Two developer-program memberships, 270
2010-data-calculated.csv file, 226–227
Type menu
 instruments, 470
 new projects, 20

U

UI automation instruments, 480
UI-layout editors, 149
UIFonts key, 377
UIApplication class, 233
UIApplicationDelegate protocol, 108
UIApplicationExitsOnSuspend key, 378
UIApplicationMain method, 242
UIBackgroundModes key, 378
UIDeviceFamily key, 377
UIFileSharingEnabled key, 378
UIImage class, 200–201
UIImageView class, 197
UIInterfaceOrientation key, 377
UIKit framework, 301
UILabel class, 160
UILaunchImageFile key, 378

- UILaunchImages key, 378
- UIMainStoryboardFile key, 376
- UINavigationController class, 152
- UIPrerenderedIcon key, 378
- UIRequiredDeviceCapabilities key, 377
- UIRequiresPersistentWiFi key, 377
- UISlider controls, 233
- UIStatusBarHidden key, 377
- UIStatusBarStyle key, 377
- UISupportedExternalAccessoryProtocols key, 377
- UISupportedInterfaceOrientations key, 377–378
- UITableView class, 136, 194, 207
- UITableViewCell class, 136, 191, 193, 196, 208
- UITableViewController class, 109, 207–208
- UITableViewDataSource class, 188
- UITableViewDelegate class, 188, 214
- UIView class, 105–106, 194
- UIViewController class, 106, 133, 147–148, 161, 187, 207
- UIViewEdgeAntialiasing key, 378
- UIViewGroupOpacity key, 378
- Umbrella headers, 61
- Undefined attributes, 115
- Undo, 183
- Unformatted field, 314
- Unit testing
 - application tests, 232–233
 - CSV Reader, 224–228
 - overview, 221–222
 - test navigator, 222–223
 - testing and debugger, 229–232
 - TestKit assertions, 233–235
- Universal (fat) binaries, 454
- Universally unique identifiers (UUIDs), 392
- Unknown file state, 78
- UNLOCALIZED_RESOURCES_FOLDER_PATH, 532
- Unmerged file state, 77
- Unmodified file state, 77
- Unresolved addresses, back-filling, 51
- Untracked file state, 76
- Unwind segues, 215
- Update All Frames in Container, 160
- Update All Frames in Game List Controller, 185
- Update Constraints, 180
- Update Frame, 180
- Update Frames menu, 161, 178
- Update Frames, 179
- URLs for application bundles, 373
- Use Autolayout, 296
- Use Base Internationalization, 338
- Use Core Data
 - iOS projects, 107
 - OS X applications, 279
- Use dot tool from the GraphViz package, 427
- Use scalar properties for primitive data types, 121
- User and System Libraries style, 469
- User Defined Runtime Attributes settings, 334–335
- User Info settings for models, 116
- User information for application bundles, 372–373
- User interface instruments, 476, 480
- User presentation in application bundles, 377–378
- USER, 528
- Using popup for build rules, 447
- /usr/bin directories, 11
- UTExportedTypeDeclarations key, 373, 408
- Utility area, 65, 150

UTImportedTypeDeclarations key, 373
UUIDs (universally unique identifiers), 392

V

Validation field for attributes, 115

Value Transformer setting, 304

valueForKeyPath method, 126

Variables

build. *See* Build settings

data tips, 332–333

Debug area, 32

Variables pane, 37–38, 143, 333, 495

Version control systems, 25, 73–74

branching, 96–98

commits, 78–79

file state, 76–77

managed-object classes, 128–131

merges and conflicts, 83–93

remote repositories, 79–83

software, 549–550

tags, 523

Version editor, 93–96

working with, 74–76

workspaces, 522

Xcode with Git, 77–78

Version Control with Subversion system, 549

Version editor, 93–96

Versioned bundles, 367

Versions version control system, 550

vi text editor, 546

View control, 31, 42, 115

View controllers, 133, 328–332

adding, 147–148

building views. *See* Building views

editing, 136–138

embedded, 206–208

OS X applications, 280

outlets. *See* Outlets

storyboards, scenes, and segues, 148–151

table views, 161

View Details, 259

View menu, deleting, 287

View selector, 23

viewDidLoad method, 208, 210–211, 213–214

Views

autolayout in. *See* Autolayout

building. *See* Building views

constraints, 157, 174–176

custom. *See* Custom views

graphing, 325–328

MVC model, 103–106

table, 136–137, 161, 207

testing, 170–171

VM Operations instrument, 479

VM Tracker instrument, 476

W

@warning keyword, 423

Warnings

compiler, 29–30, 518

disclosure triangles, 188

displaying, 54, 56

suppressing, 145

Watchdog timer, 237

watchpoint command family, 495

Web, bot creation on, 505–506

Welcome to Xcode window, 17–18

What's New in Xcode section, 415
WiFi instrument, 478
Wildcard patterns in searches, 85
Wiring menus, 287–288
 First Responder, 289
 MPRDocument data, 289–293
 targets and actions, 288
Wiring OS X applications. *See* Bindings
With XIB for user interface, 148, 162, 323
Wizard tab for Doxygen, 425, 427
WORA (write-once-run-anywhere) apps, 552
Working Directory: Use custom working directory, 487
workspace for xcodebuild, 444
Workspace (or Project) Settings, 522
Workspace tricks, 520–522
Wow feature of Assistant editor, 516
WRAPPER_EXTENSION, 532
WRAPPER_SUFFIX, 532
Write-once-run-anywhere (WORA) apps, 552

X

x-code-select, 11
X coordinates for views, 324
X11 package, 428
.xcarchive packages, 389, 393
.xcassets files, 199
 assets catalog contents, 362
 icons, 359
 OS X applications, 280
 overview, 109
 xcconfig files, 439–443
Xcode Archive, 388
Xcode icon, 17
Xcode Overview section, 415
xcode-select tool, 443, 445–446
Xcode Server
 Accounts panel, 258
 overview, 500
 registering, 503
 repositories, 74, 79–81, 501–502
xcode-users list, 541–542
XCODING_VERSION_ACTUAL, 529
xcodebuild tool, 435–436, 441, 443–445
.xcodeproj package, 444
xcrun tool, 443, 446
XCTAssert macros, 233, 235
XCTAssertEqual assertion, 226, 234–235
XCTAssertEqualObjects assertion, 235
XCTAssertEqualWithAccuracy assertion, 234
XCTAssertFalse assertion, 234
XCTAssertNil assertion, 234
XCTAssertNotEqual assertion, 234
XCTAssertNotEqualObjects assertion, 235
XCTAssertNotEqualWithAccuracy assertion, 234
XCTAssertNoThrow assertion, 235
XCTAssertNoThrowSpecific assertion, 235
XCTAssertNoThrowSpecificNamed assertion, 235
XCTAssertNotNil assertion, 226, 234
XCTAssertThrows assertion, 235
XCTAssertThrowsSpecific assertion, 235
XCTAssertThrowsSpecificNamed assertion, 235
XCTAssertTrue assertion, 226, 234
XCTest assertion macro, 221, 235
XCTest class, 226
XCTestCase class, 221

XCTFail assertion, 234

XCTest class, 222–223

Xemacs text editor, 546

XIB files, 135

linking, 148

owners, 161–162

XML

property lists, 369–370, 399, 404–408

refactoring names, 135–136

XPC services, 487

Y

Y coordinates for views, 324

Z

Zombie technique, 488

zooming

instruments, 470

Interface Builder, 149

Zuckerberg, Mark, 552