

Learn Objective-C, the gateway
to programming your iPhone, iPad, or Mac



Learn Objective-C on the Mac

For OS X and iOS

SECOND EDITION

Scott Knaster | Waqar Malik | Mark Dalrymple

Apress®

Learn Objective-C on the Mac

For iOS and OS X



Scott Knaster
Waqar Malik
Mark Dalrymple

Apress®

Learn Objective-C for iOS and OS X

Copyright © 2012 by Scott Knaster, Waqar Malik, Mark Dalrymple

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

ISBN 978-1-4302-4188-1

ISBN 978-1-4302-4189-8 (eBook)

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

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

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

President and Publisher: Paul Manning

Lead Editor: Steve Anglin

Development Editors: Gwenan Spearing, Matthew Moodie

Technical Reviewer: Nick Waynik

Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Louise Corrigan, Morgan Ertel, Jonathan Gennick, Jonathan Hassell, Robert Hutchinson, Michelle Lowman, James Markham, Matthew Moodie, Jeff Olson, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Gwenan Spearing, Matt Wade, Tom Welsh

Coordinating Editor: Brent Dubi

Copy Editor: Heather Lang

Compositor: SPi Global

Indexer: SPi Global

Artist: SPi Global

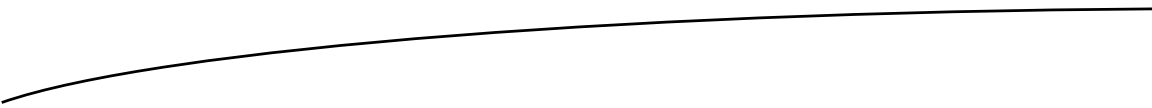
Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales—eBook Licensing web page at www.apress.com/bulk-sales.

Any source code or other supplementary materials referenced by the author in this text is available to readers at www.apress.com. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/.

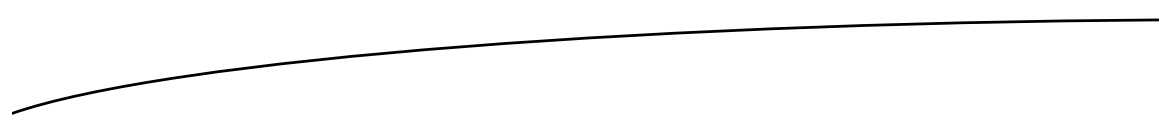


This is for my family—of course!

—Scott

*For my parents, M. Saleem and Kalsoom A. Malik,
who have always provided unconditional encouragement, support and help
in all of my endeavors.*

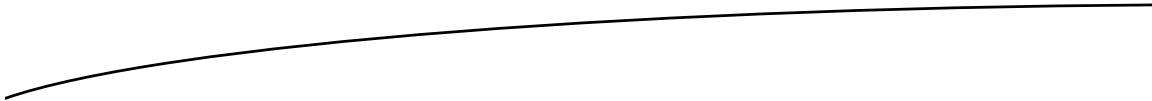
—Waqar



Contents at a Glance

Foreword	xvii
About the Authors.....	xix
About the Technical Reviewer	xxi
Acknowledgments	xxiii
Working with This Book.....	xxv
■ Chapter 1: Hello	1
■ Chapter 2: Extensions to C	7
■ Chapter 3: Introduction to Object-Oriented Programming	21
■ Chapter 4: Inheritance.....	53
■ Chapter 5: Composition	67
■ Chapter 6: Source File Organization	79
■ Chapter 7: More About Xcode.....	91
■ Chapter 8: A Quick Tour of the Foundation Kit	119
■ Chapter 9: Memory Management	145
■ Chapter 10: Object Initialization	177

■ Chapter 11: Properties	195
■ Chapter 12: Categories	209
■ Chapter 13: Protocols.....	227
■ Chapter 14: Blocks and Concurrency	239
■ Chapter 15: Introduction to UIKit.....	255
■ Chapter 16: Introduction to the Application Kit.....	277
■ Chapter 17: File Loading and Saving	293
■ Chapter 18: Key-Value Coding	303
■ Chapter 19: Using the Static Analyzer	319
■ Chapter 20: NSPredicate	329
■ Appendix A	339
Index.....	349



Contents

Foreword xvii

About the Authors..... xix

About the Technical Reviewer xxi

Acknowledgments xxiii

Working with This Book..... xxv

■ **Chapter 1: Hello 1**

 Before You Start 1

 Where the Future Was Made Yesterday..... 2

 What’s Coming Up 2

 Getting Ready 3

 Summary 6

■ **Chapter 2: Extensions to C 7**

 The Simplest Objective-C Program 7

 Building Hello Objective-C..... 7

 Deconstructing Hello Objective-C..... 12

 That Wacky #import Thing 12

Introducing Frameworks	13
NSLog() and @"strings"	13
Are You the Boolean Type?	16
Mighty BOOL in Action	16
Summary	20
■ Chapter 3: Introduction to Object-Oriented Programming	21
It's All Indirection	22
Variables and Indirection	22
Indirection Through Filenames	25
Using Indirection in Object-Oriented Programming	31
Procedural Programming	31
Implementing Object Orientation	36
Time Out for Terminology	41
OOP in Objective-C	42
The @interface Section	42
The @implementation Section	45
Instantiating Objects	48
Extending Shapes-Object	49
Summary	51
■ Chapter 4: Inheritance	53
Why Use Inheritance?	53
Inheritance Syntax	56
Time Out for Terminology	59
How Inheritance Works	59
Method Dispatching	59
Instance Variables	61
Overriding Methods	63
I Feel Super!	64
Summary	65

■ Chapter 5: Composition	67
What Is Composition?	67
Car Talk	68
Customizing for NSLog()	68
Accessor Methods	71
Setting the Engine	72
Setting the Tires	73
Tracking Changes to Car	74
Extending CarParts	75
Composition or Inheritance?	76
Summary	77
■ Chapter 6: Source File Organization	79
Split Interface and Implementation	79
Making New Files in Xcode	80
Breaking Apart the Car	83
Using Cross-File Dependencies	85
Recompiling on a Need-to-Know Basis	86
Making the Car Go	87
Importation and Inheritance	88
Summary	90
■ Chapter 7: More About Xcode	91
One Window to Rule Them All	91
Changing the Company Name	93
Using Editor Tips and Tricks	94
Writing Your Code with a Little Help from Xcode	95
Indentation (Pretty Printing)	95
You Complete Me	96
Kissing Parentheses	99
Mass Edits	99
Navigating Around in Your Code	104

Focus Your Energy	105
The Navigation Bar Is Open	106
Getting Information.....	108
Debugging	111
Crush Puny Bugs	111
Xcode's Debugger.....	111
Subtle Symbolism.....	112
Let's Debug!.....	112
Taking a Look-See	115
Cheat Sheet.....	116
Summary	117
■ Chapter 8: A Quick Tour of the Foundation Kit	119
Solid Foundation	119
Using the Project Boilerplate Code.....	120
Some Useful Types	120
Home on the Range	120
Geometric Types	121
Stringing Us Along.....	122
Build That String.....	122
Class Methods	122
Size Matters.....	123
Comparative Politics.....	124
Insensitivity Training.....	125
Is It Inside?	126
Mutability.....	126
Collection Agency	128
NSArray.....	128
Mutable Arrays	133
Enumeration Nation.....	134
Fast Enumeration	134

NSDictionary	135
Use but Don't Extend	137
Family Values	138
NSNumber	138
NSNumber	139
NSNumber	140
Example: Looking for Files	140
Behind the Sign That Says "Beware of the Leopard"	143
Summary	144
Chapter 9: Memory Management	145
Object Life Cycle.....	146
Reference Counting	146
Object Ownership	148
Retaining and Releasing in Accessors.....	148
Autorelease.....	150
Everyone into the Pool!.....	150
The Eve of Our Destruction	151
Pools in Action	152
The Rules of Cocoa Memory Management.....	154
Transient Objects	155
Hanging on to Objects	156
Take Out Those Papers and the Trash.....	158
Automatic Reference Counting.....	159
Being Exceptional.....	170
Keywords for Exceptions	171
Catching Different Types of Exceptions	172
Throwing Exceptions	172
Exceptions Need Memory Management Too.....	174
Exceptions and Autorelease Pools.....	174
Summary	175

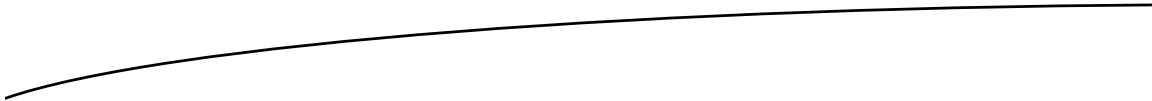
Chapter 10: Object Initialization	177
Allocating Objects	177
Initializing Objects	178
Writing Initialization Methods	178
What to Do When You're Initializing	180
Isn't That Convenient?	180
More Parts Is Parts	182
Init for Tires	182
Updating main()	184
Cleaning Up the Car	186
Car Cleaning, GC and ARC Style	189
Making a Convenience_INITIALIZER	189
The Designated_INITIALIZER	190
The Subclassing Problem	191
Fixing Tire's Initializers	192
Adding the AllWeatherRadial_INITIALIZER	193
Initializer Rules	194
Summary	194
Chapter 11: Properties	195
Shrinking Property Values	195
Shrinking the Interface	196
Shrinking the Implementation	197
Dots Incredible	200
Objecting to Properties	201
Appellation Spring	204
Read-Only About It	206
I'd Rather Do It Myself	207
Alas, Properties Don't Do Everything	208
Summary	208

■ Chapter 12: Categories	209
Creating a Category	209
Let's Create a Category	210
@interface	211
@implementation	212
Bad Categories	214
Good Categories	214
Clint Eastwood Would Love This	215
Splitting an Implementation with Categories	216
Using Categories in our Project	216
Making Forward References with Categories	219
Categories to the Rescue!	220
Informal Protocols and Delegation Categories	221
The iTunesFinder Project	221
Delegates and Categories	224
Responds to Selectors	225
Other Uses for Selectors	225
Summary	226
■ Chapter 13: Protocols	227
Formal Protocols	227
Declaring Protocols	228
Adopting a Protocol	228
Implementing a Protocol	229
Car-bon Copies	229
Copying Engines	230
Copying Tires	231
Copying the Car	232
Protocols and Data Types	235
Objective-C 2.0 Goodies	235
The Delegation Will Come to Order	236
Summary	238

■ Chapter 14: Blocks and Concurrency	239
You're Never Too Old to Play with Blocks	239
Blocks and Function Pointers	239
Objective-C Objects	243
Concurrency, or Keeping Up with Yourself	244
Synchronize	244
Memory Management Is for Queues Too	248
Operation Queues	251
Summary	252
■ Chapter 15: Introduction to UIKit.....	255
View Controllers	260
Adding Items to the Nib File	260
Summary	276
■ Chapter 16: Introduction to the Application Kit.....	277
Making the Project	278
Making the Delegate @interface.....	280
Interface Builder	281
Laying Out the User Interface	282
Making Connections.....	285
Hooking Up the Outlets.....	285
Hooking Up the Actions.....	287
AppDelegate Implementation	289
Summary	291
■ Chapter 17: File Loading and Saving.....	293
Property Lists	293
NSDate.....	293
NSData.....	294
Writing and Reading Property Lists	295
Modifying Objects.....	297

Encoding Objects.....	297
Summary.....	302
Chapter 18: Key-Value Coding	303
A Starter Project.....	303
Introducing KVC.....	305
A Path! A Path!	306
Aggregated Assault	307
Pit Stop	308
Smooth Operator	311
Life's a Batch.....	313
The Nils Are Alive.....	314
Handling the Unhandled	315
Summary.....	316
Chapter 19: Using the Static Analyzer.....	319
Getting Some Static.....	319
Going Into Analysis	320
Assisting the Analyzer	324
On Further Analysis	325
Summary.....	327
Chapter 20: NSPredicate	329
Creating a Predicate.....	330
Evaluate the Predicate	330
Fuel Filters	331
Format Specifiers.....	333
Hello Operator, Give Me Number 9	334
Comparison and Logical Operators	334
Array Operators	335

SELF Sufficient	336
String Operations	337
Like, Fer Sure	338
That's All, Folks	338
■ Appendix A	339
Index.....	349



Foreword

Whenever software developers adopt a new platform, they are faced with the daunting task of familiarizing themselves with the programming language, development tools, design patterns, and standard software libraries available in the new environment.

Typically, this must be done simultaneously with writing code under pressure to deliver it as quickly as possible, and developers are tempted to fall back on approaches they know from previous systems. This often results in code that doesn't really fit the platform, that may duplicate functionality that already existed, and can cause maintenance headaches down the road.

In an ideal world, new developers would have the benefit of colleagues who are already familiar with the platform, who can offer their guidance to get them started and moving in the right direction. Unfortunately, the very success of the iOS platform has made for many cases where this just isn't possible.

If you don't have a mentor handy, what's the next best thing?

The authors of this book are veterans of Apple's Developer Technical Services organization, and they have each answered countless questions from software engineers who are new to Apple's technology. They have been responsible for instilling good habits in those they've helped. That experience results in a book that anticipates the most common misunderstandings and takes care to explain not only the how, but also the why of Apple's development platform.

For example, the conceptual basis provided in Chapter 3, "Introduction to Object-Oriented Programming," gives you the means to place the material that follows it into a coherent picture, instead of just tossing you into the midst of a flurry of unfamiliar classes, methods, and techniques and hoping that you'll somehow sort it all out with practice.

Learn Objective-C on the Mac is a fine guide to the language at the heart of Apple's iOS and OS X development platform.

John C. Randolph

About the Authors



Scott Knaster was at Apple when Apple wasn't cool. Scott worked at Apple helping developers write Mac software in the earliest days of the platform, when Cocoa was just a great idea waiting to be born. Scott now works on Google's Developer Relations team and runs the Google Mac Blog. He lives in Silicon Valley among his nerdy peers.



Waqar Malik is a UNIX nerd and has been for long time. He worked at Apple during the early days of Mac OS X, helping developers with Cocoa and UNIX. He now works for MeLLmo, Inc. in San Diego, writing great iOS Software.



Mark Dalrymple is a longtime Mac and Unix programmer who has worked on cross-platform toolkits, Internet publishing tools, high-performance web servers, and end-user desktop applications. He's also the principal author of *Advanced Mac OS X Programming* (Big Nerd Ranch 2005). In his spare time, he plays trombone and bassoon and makes balloon animals.

About the Technical Reviewer



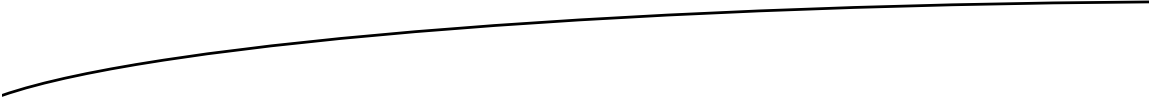
Nick Waynik has been working in the IT field for over 13 years and has done everything from network administration to web development. He started writing iOS apps at the beginning of 2008 when the SDK was released. Since then, he has gone on to start his own business focusing on iOS development. In his spare time, he loves to spend time with his wife and family and play golf. He blogs at nickwaynik.com and can be found on Twitter as @n_dubbs.

Acknowledgments

This book didn't write itself, you know. And we, the authors, didn't do all the work either—not even close. Sure, we typed a bunch of words and code, but without our amazing book team, what would we have? A really long blog post, maybe.

Enormous thanks go to Brent Dubi, who held everything together with relentlessly cheerful e-mails and focused phone calls, both vital tools in coordinating a virtual team of people from around the world. Thanks to Nick Waynik for keeping us technically honest, which is not an easy task given the power and complexity that Apple packs into Xcode and Cocoa nowadays. We thank Gwenan Spearing, who scoured every word we wrote, even the bad puns, to help us make sure we were as clear and concise as possible. And second-edition thanks to our excellent first-edition copy editor, Heather Lang, who returned and once again allowed us to push the limits of proper grammar and style just so we could make yet another nerdy *Star Wars* joke. Each of these folks improved this book immeasurably and so have made us look our best.

Waqar would also like to thank his wonderful children, Adam and Mishal, and his beautiful wife, Irrum, for giving him enough time to work on this book.



Working with This Book

To download source code or report an error, please see this book's page on the Apress site:
www.apress.com/9781430241881.

For best results when using this book, have your computer close by while you read. Having your favorite beverage at hand is also nice, but keep it away from the computer, because liquid damage repair prices will ruin your whole day.

Most of all, have fun while you learn this stuff!

Chapter 1

Hello

Welcome to *Learn Objective-C on the Mac*! This book is designed to teach you the basics of the Objective-C language. Objective-C is a superset of C and is the language used by many (if not most) applications that have a true OS X or iOS look and feel.

In addition to presenting Objective-C, this book introduces you to its companion, Apple's Cocoa (for OS X) and Cocoa Touch (for iOS) toolkits. Cocoa and Cocoa Touch are written in Objective-C and contain all the elements of the OS X and iOS user interfaces, plus a whole lot more. Once you learn Objective-C, you'll be ready to dive into Cocoa with a full-blown project or another book such as *Learn Cocoa on the Mac* (Apress 2010) or *Beginning iOS 5 Development* (Apress 2011).

In this chapter, we'll let you know the basic information you need before you get started with Objective-C itself. We'll also serve up a bit of history about Objective-C and give you a thumbnail sketch of what's to come in future chapters.

Before You Start

Before you read this book, you should have some experience with a C-like programming language such as C++, Java, or venerable C itself. Whatever the language, you should feel comfortable with its basic principles. You should know what variables, methods, and functions are and understand how to control your program's flow using conditionals and loops. Our focus is the features Objective-C adds to its base language, C, along with some goodies chosen from Apple's Cocoa toolkits.

Are you coming to Objective-C from a non-C language? You'll still be able to follow along, but you might want to take a look at this book's Appendix or check out *Learn C on the Mac* (Apress 2009).

Where the Future Was Made Yesterday

Cocoa and Objective-C are at the heart of Apple's OS X and iOS operating systems. Although OS X and especially iOS are relatively new, Objective-C and Cocoa are much older. Brad Cox invented Objective-C in the early 1980s to meld the popular and portable C language with the elegant Smalltalk language. In 1985, Steve Jobs founded NeXT, Inc., to create powerful, affordable workstations. NeXT chose Unix as its operating system and created NextSTEP, a powerful user interface toolkit developed in Objective-C. Despite its features and a small, loyal following, NextSTEP achieved little commercial success.

When Apple acquired NeXT in 1996 (or was it the other way around?), NextSTEP was renamed Cocoa and brought to the wider audience of Macintosh programmers. Apple gives away its development tools—including Cocoa—for free, so any programmer can take advantage of them. All you need is a bit of programming experience, basic knowledge of Objective-C, and the desire to dig in and learn stuff.

You might wonder, "If Objective-C and Cocoa were invented in the '80s—in the days of *Alf* and *The A-Team*, not to mention stuffy old Unix—aren't they old and moldy by now?" Absolutely not! Objective-C and Cocoa are the result of years of effort by a team of excellent programmers, and they have been continually updated and enhanced. Over time, Objective-C and Cocoa have evolved into an incredibly elegant and powerful set of tools. Over the past few years, iOS has become the hottest development platform in computing, and Objective-C is the key to writing great iOS applications. So now, twenty-some years after NeXT adopted Objective-C, all the cool kids are using it.

What's Coming Up

Objective-C is a superset of C: it begins with C and then adds a couple of small but significant additions to the language. If you've ever looked at C++ or Java, you may be surprised at how small Objective-C really is. We'll cover Objective-C's additions to C in detail in this book's chapters:

- Chapter 2, "Extensions to C," focuses on the basic features that Objective-C introduces.
- In Chapter 3, "An Introduction to Object-Oriented Programming," we kick off the learning by showing you the basics of object-oriented programming.
- Chapter 4, "Inheritance," describes how to create classes that gain the features of their parent classes.
- Chapter 5, "Composition," discusses techniques for combining objects so they can work together.
- Chapter 6, "Source File Organization and Using Xcode 4," presents real-world strategies for creating your program's sources.
- Chapter 7, "More about Xcode," shows you some shortcuts and power-user features to help you get the most out of your programming day.
- We take a brief respite from Objective-C in Chapter 8, "A Quick Tour of the Foundation Kit," to impress you with some of Cocoa's cool features using one of its primary frameworks.

- You'll spend a lot of time in your Cocoa applications dealing with Chapter 9's topic, "Memory Management and ARC".
- Chapter 10, "Object Initialization," is all about what happens at that magical time when objects are born.
- Chapter 11, "Properties," gives you the lowdown on Objective-C's dot notation and an easier way to make object accessors.
- Chapter 12, "Categories," describes the super cool Objective-C feature that lets you add your own methods to existing classes—even those you didn't write.
- Chapter 13, "Protocols," tells about a form of inheritance in Objective-C that allows classes to implement packaged sets of features.
- Chapter 14, "Blocks and Concurrency" shows you how to use a new Objective-C feature that enhances functions into blocks that can include data as well as code.
- Chapter 15, "Introduction to UIKit" gives you a taste of the gorgeous applications you can develop for iOS using its primary framework.
- Chapter 16, "Introduction to AppKit," is similar to Chapter 15 except that it introduces the basic framework for OS X applications.
- Chapter 17, "File Loading and Saving," shows you how to save and retrieve your data.
- Chapter 18, "Key-Value Coding," gives you ways to deal with your data indirectly.
- Chapter 19, "Using the Static Analyzer" shows you how to use a powerful Xcode tool to find common mistakes programmers make.
- And finally, in Chapter 20, "NSPredicate," we show you how to slice and dice your data.

If you're coming from another language like Java or C++, or from another platform like Windows or Linux, you may want to check out this book's Appendix, "Coming to Objective-C from Other Languages," which points out some of the mental hurdles you'll need to jump to embrace Objective-C.

Getting Ready

Xcode is the development environment provided by Apple for creating iOS and OS X applications. Macs don't come with Xcode preinstalled, but downloading and installing it is easy and free. All you need is a Mac running OS X 10.7 Lion or later.

The first step on the long and awesome road to programming for OS X or iOS is acquiring a copy of Xcode. If you don't have it already, you can download it from the Mac App Store. To get there, click the App Store icon in the dock (see Figure 1-1), or find the App Store in the *Applications* folder.

In the Mac App Store, click in the search box in the upper right, and search for Xcode (see Figure 1-2).



Figure 1-1. App Store icon in the dock

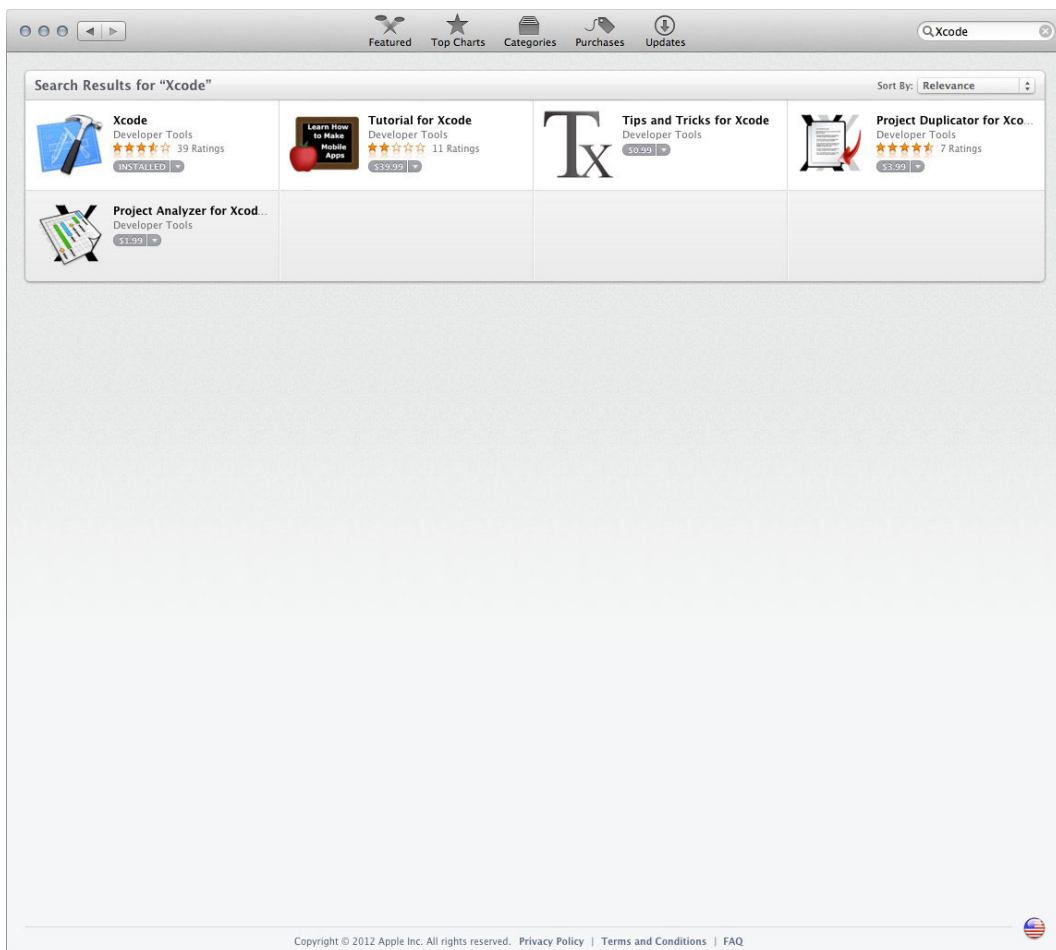


Figure 1-2. Search for Xcode in the Mac App Store

Or, click *Categories* and then *Developer Tools*, and you'll see Xcode on the top left (see Figure 1-3) or somewhere nearby. Click *Xcode* to see its download page (see Figure 1-4).



Figure 1-3. Developer Tools Apps



Figure 1-4. Xcode download page in Mac App Store

Click *Free* and then *Install App*. The App Store installs Xcode in your Applications folder.

Now, you're ready to start your journey. Good luck! We'll be there with you for at least the first part of your trip.

Summary

OS X and iOS programs are written in Objective-C, using technology from way back in the 1980s that has matured into a powerful set of tools. In this book, we'll start by assuming you know something about C programming or another general-purpose programming language and go from there.

We hope you enjoy your adventure!

Extensions to C

Objective-C is nothing more than the C language with some extra features drizzled on top—it's delicious! In this chapter, we'll cover some of those key extras as we take you through building your first Objective-C program—and your second one too.

The Simplest Objective-C Program

You've probably seen the C version of the classic Hello World program, which prints out the text "Hello, world!" or a similar pithy remark. Hello World is usually the first program that neophyte C programmers learn. We don't want to buck tradition, so we're going to write a similar program here called Hello Objective-C.

Building Hello Objective-C

As you work through this book, we're assuming you have Apple's Xcode tools installed. If you don't already have Xcode, or if you've never used it before, an excellent section in Chapter 2 of Dave Mark's *Learn C on the Mac* (Apress 2008) walks you through the steps of acquiring, installing, and creating programs with Xcode.

In this section, we'll step through the process of using Xcode to create your first Objective-C project. If you are already familiar with Xcode, feel free to skip ahead; you won't hurt our feelings. Before you go, be sure to expand the Learn ObjC Projects archive from this book's archive (which you can download from the Source Code/Download page of the Apress web site). This project is located in the 02.01 - Hello Objective-C folder.

To create the project, start by launching Xcode. You can find the Xcode application in `/Developer/Applications`. We put the Xcode icon in the Dock for easy access. You might want to do that too.

Once Xcode finishes launching, you'll see the Welcome screen, as shown in Figure 2-1. On the left side, you can select the next thing you want to do. Or, you can choose to open a recent project from the list on the right. (If you're brand new with Xcode, you won't see any recent



Figure 2-1. Xcode Welcome screen

projects.) If you don't see the Welcome screen, you can always show it by selecting "Welcome to Xcode" on the Window menu or by typing `⌘⇧1`.

On the Welcome screen, click "Create a new Xcode project" (see Figure 2-1), or just choose **File ► New ► New Project**. Xcode shows you a list of the various kinds of projects it can create. Use your focus to ignore most of the intriguing project types there, and choose Application on the left-hand side of the window and Command Line Tool on the right-hand side, as shown in Figure 2-2. Click Next.

On the next screen (Figure 2-3), you'll select options for your new project. For Product Name, enter the timeless classic "Hello Objective-C". For Company Identifier, you'll typically enter a reverse DNS version of your company or website name, such as `com.mywebsite`; for now, you can just enter `com.thinkofsomethingclever`.

This screen saves the best for last, as the most important option is the type of command line tool you want to create: be sure to choose Foundation. Once you're done, your screen should look a lot like Figure 2-3. After you've done this, click Next.

Xcode drops a sheet and asks you where to save your project (see Figure 2-4). We're putting it into one of our Projects directories here to keep things organized, but you can put it anywhere you want.

After you click Save, Xcode shows you its main window, called the project window (see Figure 2-5). This window displays the pieces that compose your project along with an editing pane. `main.m` is the source file that contains the code for Hello Objective-C.

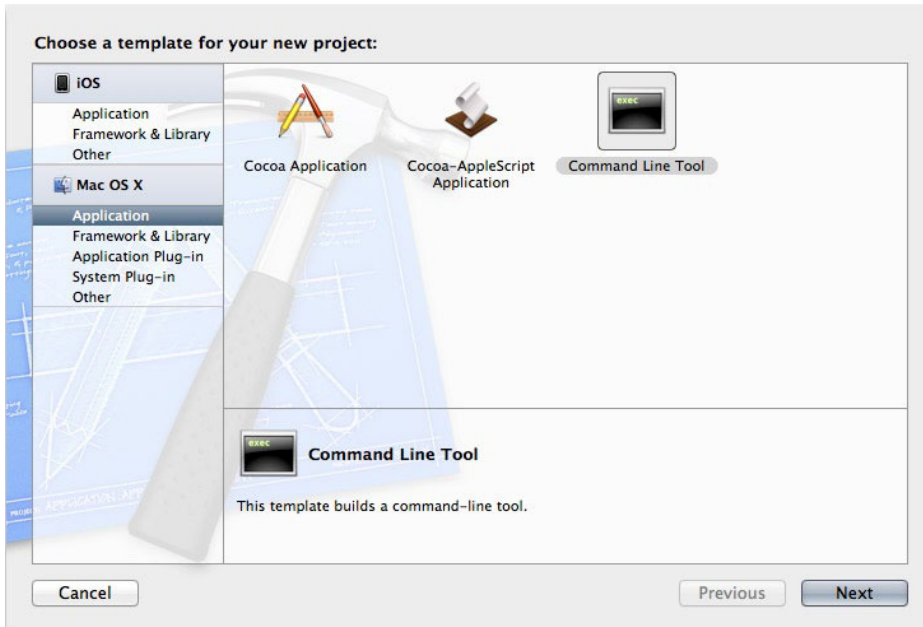


Figure 2-2. Making a new command line tool

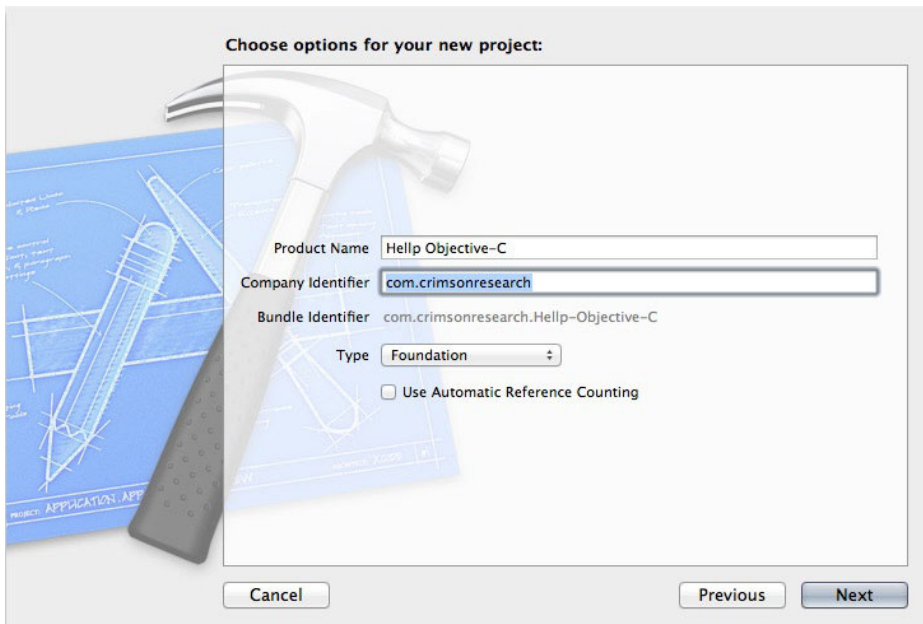


Figure 2-3. Set your project's options