Learning the iOS 4 SDK for JavaScript Programmers

Is it possible for JavaScript programmers to learn Apple’s iOS 4 SDK and live to tell the tale? Technology guru Danny Goodman did, and with this book he leaves a well-marked trail for you to follow. An authority on JavaScript since its inception, Goodman understands the challenges you might face in creating native iOS apps with this SDK, and introduces Xcode, Objective-C, and Cocoa Touch in a context you’ll readily understand.

Why bother with the SDK when you can simply build web apps for Apple’s iOS devices? Web apps can’t access an iPhone’s music library, camera, or iOS system software for maps, audio, and more. Nor can you sell web apps in the App Store. If you want to take full advantage of the iPhone and iPad, iOS 4 SDK is your tool—and this is your book.

“Anyone with a web background wanting to jump into native apps should follow Danny’s investigative approach to learning the platform. This is the perfect guide for any seasoned JavaScript programmer wanting to make the journey to the App Store.”

—Todd Moore
Creator of White Noise

Danny Goodman has written more than three dozen books and hundreds of magazine articles on personal computers and consumer electronics. Recently, he’s been programming iPhone and iPod touch apps, including jFeltThat Earthquake, PhotoSize, and BeaconAid-HF.

Previous programming experience is recommended.

US $34.99 CAN $40.99
ISBN: 978-1-449-38845-4

Free online edition for 45 days with purchase of this book. Details on last page.
Learning the iOS 4 SDK for
JavaScript Programmers
Table of Contents

Preface ................................................................................................................. xi

1. Why Go Native? ................................................................................................. 1
   Using an App Offline ......................................................................................... 2
   More Access to the Hardware .......................................................................... 3
   More Access to the Software .......................................................................... 4
   What You Lose
      Distribution .................................................................................................. 6
   Apple iOS Developer Program ....................................................................... 7
   Content .............................................................................................................. 8
   Authoring Platform Choices .......................................................................... 8
   Taking the Plunge ............................................................................................ 9

2. Welcome to the iOS SDK ..................................................................................... 11
   Hardware and OS Requirements ................................................................... 11
   Installing the SDK .......................................................................................... 12
   About iOS Developer Programs ................................................................... 12
   Inside the SDK ............................................................................................... 14
   Viewing Developer Documentation ................................................................. 15
   Loading Code Samples .................................................................................. 18
   Setting the Project’s Base SDK ..................................................................... 21
   Trying the iOS Simulator ................................................................................ 22
   Coming Up... ................................................................................................... 24

3. Creating a Test Workbench ................................................................................. 25
   Creating the Project in Xcode ....................................................................... 26
      Selecting a Project Type ............................................................................... 26
      Naming and Saving the New Project ............................................................. 29
      Welcome to Your Project ............................................................................ 29
      Editing Your First Files ............................................................................... 31
      What the runMyCode: Method Does .......................................................... 34
4. **Structural Overview of an iOS App** ........................................... 51
   Where It All Begins: APIs ...................................................... 51
   APIs You Already Know ..................................................... 51
   The Cocoa Touch APIs ....................................................... 52
   Frameworks ......................................................................... 53
   Foundation Framework ....................................................... 54
   UIKit Framework ............................................................... 54
   CoreGraphics Framework .................................................... 55
   Adding Frameworks ............................................................ 55
   Frameworks Set in Stone ...................................................... 56
   Welcome to Class Files ....................................................... 57
   The JavaScript Way ............................................................. 57
   The Objective-C Way ......................................................... 58
   Header File Details ............................................................. 61
   Using Xcode to Create DGCar Class Files ......................... 65
   Editing the @interface Section ............................................. 68
   Message Passing .................................................................. 70
   Editing the @implementation Section .................................. 70
   Integrating the DGCar Class into Workbench ..................... 75
   Creating Object Instances ................................................ 76
   NSLog() and String Formats ............................................... 77
   Running the Code ............................................................... 78
   What About Accessing Instance Variables? ......................... 79
   Recap ............................................................................... 81

5. **App Execution Flow** ............................................................... 83
   Some C Language Roots in an iOS App ................................. 83
   An Introduction to Delegates .............................................. 85
      How UIApplication Appoints Its Delegate ......................... 87
      The App’s Info.plist File ................................................. 87
      Inside MainWindow.xib ................................................... 88
   iPhone App Development Design Patterns ......................... 92
      The Model-View-Controller Design Pattern ..................... 92
      Other Design Patterns ................................................... 94
   The Importance of Views .................................................. 95
      The App Window—UIWindow ......................................... 96
      Adding Another View to Workbench ................................. 97
6. Central Objective-C Concepts: Pointers, Data Types, and Memory Management 107

Pointers 108
  Pointers and Memory 108
  Pointers and Objective-C Variables 110
  Pointer Notation 111
  Determining Pointer Usage 113
Data Typing 115
  Objective-C Data Types 116
  Cocoa Touch Data Types 116
  Objective-C Variable Declarations 118
  Objective-C Method Declarations 118
  The id Data Type 122
  Converting Objective-C Data Types 123
Memory Management 125
  Cleaning Up After Yourself 125
  The Retain Count 127
  Autorelease Pools 129
  Observing Memory Usage 130
Recap 131

7. C Language Fundamentals .................................................. 133

Variable Names 133
Variable Scope 136
  Instance Variables 137
  Local Variables 137
  Local Variables in Control Structure Blocks 138
  Static Local Variables 140
  Global Variables 140
Constant Values 141
Functions 142
C Structures 148
C Arrays 151
Enumerated Types 152
Operators 153
Program Flow Constructions 153
Boolean Values 154
Math Object Equivalents in C 155
Inserting Comments 157
Recap 157
You don’t have to be an Apple fanboy or fangirl to give Apple Inc. credit for redefining mobile gadgetry and its surrounding industries. First the company used the iPod to reshape the music industry and strongly influence how we acquire and consume tunes. Just count the number of people wearing iPod-connected earbuds in a subway car. Then the iPhone rewrote the cellular telephone industry manual, while opening the world’s eyes to the potential of being connected to the Internet nearly everywhere, all the time. It’s happening again with the iPad, where electronic publishing is evolving right before our eyes.

Although the iPhone was an early success with just the workable but limited set of Apple-supplied applications that came with the phone, programmers couldn’t wait to get their hands on the platform. The first word that Apple let drop about third-party developers, however, landed with a bit of a thud: they were graciously allowed to create web apps. Sure, the iPhone’s WebKit-based browser let creative HTML, CSS, and JavaScript programmers create far more than dull web pages, but the apps still faced frustrating limits compared to Apple’s native apps.

It took some additional months, but Apple eventually released a genuine software development kit (SDK) to allow third-party programmers to create native applications for what was then called the iPhone OS. Part of Apple’s task was also creating the App Store to distribute apps—yet another industry-transforming effort. Many existing Mac OS X developers rejoiced because the iPhone OS was derived from Mac OS X. The iPhone SDK was based on the same Xcode tools that Mac developers had been using for some time. The language of choice was Objective-C.

As a happy iPhone early adopter, I eagerly awaited the iPhone SDK. Unfortunately, despite my years of being a dedicated Mac user since 1984 and a scripter since 1987 and the HyperCard days, I had never done any Mac OS X programming. I didn’t know much about C and next to nothing about Objective-C. Still, I thought perhaps my years of experience in JavaScript would be of some help. After all, at one time I even learned enough Java to write a small browser applet to demonstrate how JavaScript code in a web page can communicate with the applet. At least I knew what a compiler did.
When the iPhone SDK landed on my Mac, I was simply overwhelmed. The old metaphor of trying to sip from a firehose definitely applied. The more I read Apple’s early developer documentation, the more I felt as though I had to know a lot more than I knew just to understand the “getting started” texts. With JavaScript having been the most recent language acquisition for me (albeit back in late 1995), I looked for anything I could borrow from that experience to apply to iPhone app development. I’d see occasional glimmers, but I was basically flying blind, not knowing what I had to discard and what I could keep.

The SDK was evolving during that time as well. I’d read a tutorial here and there, but I wasn’t making much headway at first. Some tools, especially Interface Builder, felt incomplete to me. Frankly, I had a couple of false starts where I walked away until a future SDK version appeared. Finally, I reached a point that was “put up or shut up.” After sticking with it and reading many of the documents many times, I was, indeed, getting tastes from the firehose. Working on iPhone development as a part-time effort over a three-month period, I managed to go from the starting line to submitting my first app to the App Store in January 2009.

Since then I’ve been monitoring the developer communities on both the native app and web app sides. I’ve even sat in online courses for web app developers to see what they’re saying in the chat room. A lot of web app developers seem to look enviously to native iPhone and iPad development. I suspect many have gone through the same false starts that I did. And yet I know from my own experience that it is possible to make the transition from web app to native app developer if you know how to channel your JavaScript knowledge into what is now known as the iOS SDK environment.

What You Need to Start

I have written this book specifically for the web developer who is comfortable in the JavaScript language. Even if you use a bit of JavaScript to glue together apps from third-party JavaScript libraries and frameworks, you should be ready for this book. Unlike most entry-level iOS programming books, this one assumes that you have not necessarily worked in a compiled language before. You probably have little or no experience with C or Objective-C. But you do know what a string and an array are because you use them in your JavaScript work. I will be introducing you to the way Objective-C works by comparing and contrasting what you use in JavaScript. It’s the kind of hand-holding that I wish I had when I started learning iPhone app development.

You will get more from this book if you are the adventurous type. By adventurous, I mean that you will follow the instructions throughout to try things for yourself. Along the way I will help you build an app called Workbench, where you will be able to play and learn by experimenting with little pieces of code here and there. Creating projects, editing files, and building apps is the only way to really get to know the SDK.
Of course, you’ll need a Macintosh running Mac OS X version 10.6 (Snow Leopard) or later. I’ll have more details about getting set up with hardware and SDK software in Chapter 2.

**What’s in This Book**

Perhaps because my programming knowledge has been completely self-taught over the decades, this book does not follow what some might term traditional programming language training. First of all, you already come to the book with specialized knowledge. The goal of the book is to pick up where that knowledge leaves off and fill in the gaps with the new material. There’s no doubt about it: there is a lot of new material for you. But I have tried to establish a learning progression that will make sense and keep you interested while you learn the decidedly unglamorous—but essential—parts of iOS programming.

**Chapter 1** goes into detail about the differences between web app and native app programming for devices running iOS. It’s not all roses for native app development, as you’ll see, but I believe the positives outweigh the negatives. In **Chapter 2**, you will install the iOS SDK, inspect one of the sample apps, and run it on the iOS Simulator. Then in **Chapter 3**, I put you to work to create your first iPhone app—the Workbench app that you’ll use throughout the rest of the book. The steps are intended to help you get more comfortable with Xcode and learn what it’s like to work on an app in the environment.

In **Chapter 4**, you will use the Workbench app to build your first Objective-C object and compare the process against building the same object in JavaScript. You will spend a lot of time in Xcode. And if you’ve used JavaScript frameworks for your web app development, wait until you get a peek at the frameworks you’ll be using in iOS app development.

The focus of **Chapter 5** is understanding how the code you write commands an iOS device to launch your app and get it ready for a user to work with. In the process, you’ll learn a great deal about how an app works. In fact, by the end of this chapter, you will add a second screen to Workbench and animatedly switch between the two.

Sometimes while learning new material, you have to take your medicine. That happens in **Chapter 6**, where you meet three programming concepts that are foreign to what you know from JavaScript: pointers, data typing, and memory management. There will be plenty of sample code for you to try in the Workbench app to learn these new concepts.

Objective-C is built atop the C language. There is still a bit of C that you should know to be more comfortable in the newer language. **Chapter 7** shows you what you need to know from C. The good news is that a fair amount of it is identical to JavaScript. Hooray! And most of the rest it isn’t needed because it’s all covered in more robust
and friendly ways in Objective-C, as covered in Chapter 8. There you’ll learn how Objective-C handles strings, arrays, and other data collections.

The final chapter, Chapter 9, is also the longest. It provides a catalog of programming tasks you’re accustomed to, but implemented in the iOS SDK. Most of the jobs will be familiar to you—formatting numbers, performing date calculations, sorting arrays, working with user-entered text, having Ajax-like communications with a server, and even dragging an item around a screen. I don’t expect you to learn and remember everything described in Chapter 9, but know what’s there and how to find it when the need arises in your own iOS development.

Two appendixes round out the offering. One provides tips on using the iOS SDK’s documentation to its fullest extent. The other presents a list of common Xcode compiler errors that beginners encounter and what the errors really mean. Unintelligible error messages in the early going of learning a new environment can be very frustrating and discouraging. Appendix B makes it possible to learn more quickly from newbie mistakes.

**Conventions Used in This Book**

The following typographical conventions are used in this book:

Plain text
- Indicates menu titles, menu options, menu buttons, and keys.

*Italic*
- Indicates new terms, URLs, email addresses, filenames, file extensions, and directories.

Constant width
- Indicates variables, methods, types, classes, properties, parameters, values, objects, XML tags, the contents of files, and logging output.

**Constant width bold**
- Highlights new code or code of special importance in examples.

*Constant width italic*
- Shows text that should be replaced with user-supplied values.

This icon signifies a tip, suggestion, or general note.

This icon indicates a warning or caution.
About the Author

Danny Goodman has been writing about personal computers and consumer electronics since the late 1970s. A freelance writer and programmer, he’s published hundreds of magazine articles, several commercial software products, and three dozen computer books. His most popular book titles—on HyperCard, AppleScript, and JavaScript—have covered programming environments that are both accessible to non-professionals yet powerful enough to engage experts. He is currently an independent iOS app developer, with three products available on the App Store and more in the pipeline.

Colophon

The dog on the cover of Learning the iOS 4 SDK for JavaScript Programmers is a King Charles Spaniel. Today’s Cavalier King Charles Spaniel is descended from a small, “toy” type of spaniel that was popular in 16th-century England. King Charles II, from whom the breed gets its name, was so fond of these dogs that he decreed that they were to be allowed in any public place, and it was said that “His Majesty was seldom seen without his little dogs.” These spaniels were often referred to as “Comforters”; in the winter, a noble lady riding in a carriage was likely to keep a spaniel in her lap for warmth. While used by some for hunting small game, the King Charles Spaniel was typically valued for its companionship and considered more of a luxury item than a utilitarian pet.

Today’s King Charles Spaniel emerged in part from interbreeding with the pug—which was in fashion in England during the reign of King William III and Queen Mary II—and the longer-nosed spaniels Charles II was so fond of. Their pointed noses, flat heads, and almond-shaped eyes were replaced with the shorter muzzles, domed skulls, and large, round eyes that characterize them today. The turn of the 20th century saw a final attempt to revive the breed as it existed during King Charles’s time, but the modern King Charles Spaniel—named “Cavalier King Charles Spaniel” by the Cavelier Club in 1928—persisted. During World War II, the breed declined significantly (with one registered kennel dropping from 60 to 3 Caveliers), but regained popularity after the war and throughout the 1940s.

Today, the Cavalier King Charles Spaniel is gaining popularity worldwide. There are national Cavalier breed clubs in about a dozen countries, including Finland, Italy, New Zealand, and South Africa. The Kennel Club reports that the Cavalier was the sixth most popular dog in the UK in 2007, and according to statistics from the American Kennel Club, they were the 25th most popular in the US in 2008, particularly in San Francisco, New York City, Boston, and Washington, D.C..

The cover image is from Wood’s Animate Creation, Vol. I. The cover font is Adobe ITC Garamond. The text font is Linotype Birka; the heading font is Adobe Myriad Condensed; and the code font is Free Font Foundry’s TheSansMonoCondensed.