



iPhone IN ACTION

**Introduction to Web
and SDK Development**

SAMPLE CHAPTER

Christopher Allen
Shannon Appelcline

 MANNING



iPhone in Action

Introduction to Web and SDK Development

by Christopher Allen
and Shannon Appelcline

Chapter 2

Copyright 2009 Manning Publications

brief contents

PART 1	INTRODUCING IPHONE PROGRAMMING.....	1
	1 ■ Introducing the iPhone	3
	2 ■ Web development or the SDK?	16
PART 2	DESIGNING WEB PAGES FOR THE IPHONE	29
	3 ■ Redeveloping web pages for the iPhone	31
	4 ■ Advanced WebKit and textual web apps	55
	5 ■ Using iUI for web apps	80
	6 ■ Using Canvas for web apps	102
	7 ■ Building web apps with Dashcode	124
	8 ■ Debugging iPhone web pages	143
	9 ■ SDK programming for web developers	154
PART 3	LEARNING SDK FUNDAMENTALS	165
	10 ■ Learning Objective-C and the iPhone OS	167
	11 ■ Using Xcode	190
	12 ■ Using Interface Builder	206

- 13 ■ Creating basic view controllers 221
- 14 ■ Monitoring events and actions 240
- 15 ■ Creating advanced view controllers 264

PART 4 PROGRAMMING WITH THE SDK TOOLKIT..... 283

- 16 ■ Data: actions, preferences, files, SQLite, and addresses 285
- 17 ■ Positioning: accelerometers and location 324
- 18 ■ Media: images and sounds 344
- 19 ■ Graphics: Quartz, Core Animation, and OpenGL 366
- 20 ■ The web: web views and internet protocols 396

Web development or the SDK?

This chapter covers

- The two types of iPhone development
- Ways to program for the iPhone
- Integrated project development

There are two ways you can develop for the iPhone. One approach is to write web pages for mobile Safari, using HTML, CSS, JavaScript, and your favorite dynamic language. The other is to write native applications to run directly on the iPhone, using Objective-C and the iPhone SDK.

We strongly believe that each programming method has its own place, and that you should always ensure you're using the correct one before you get started with any iPhone project.

2.1 Comparing the two programming styles

One of the things that might surprise programmers coming to the iPhone for the first time is the fact that web development and SDK programming can produce similar user-level experiences and support much of the same functionality. We've

highlighted this in figure 2.1, which kicks off our discussion of the fact that web development and SDK programming are both reasonable alternatives when creating applications for the iPhone.

Figure 2.1 depicts what iPhone developers call a “utility,” a two-page iPhone application that contains the actual application on the front page and setup information on the back page. Within the illustration, we’ve included a snippet of the code that allows the utility to flip between the pages when the info button is pushed. It’s done in

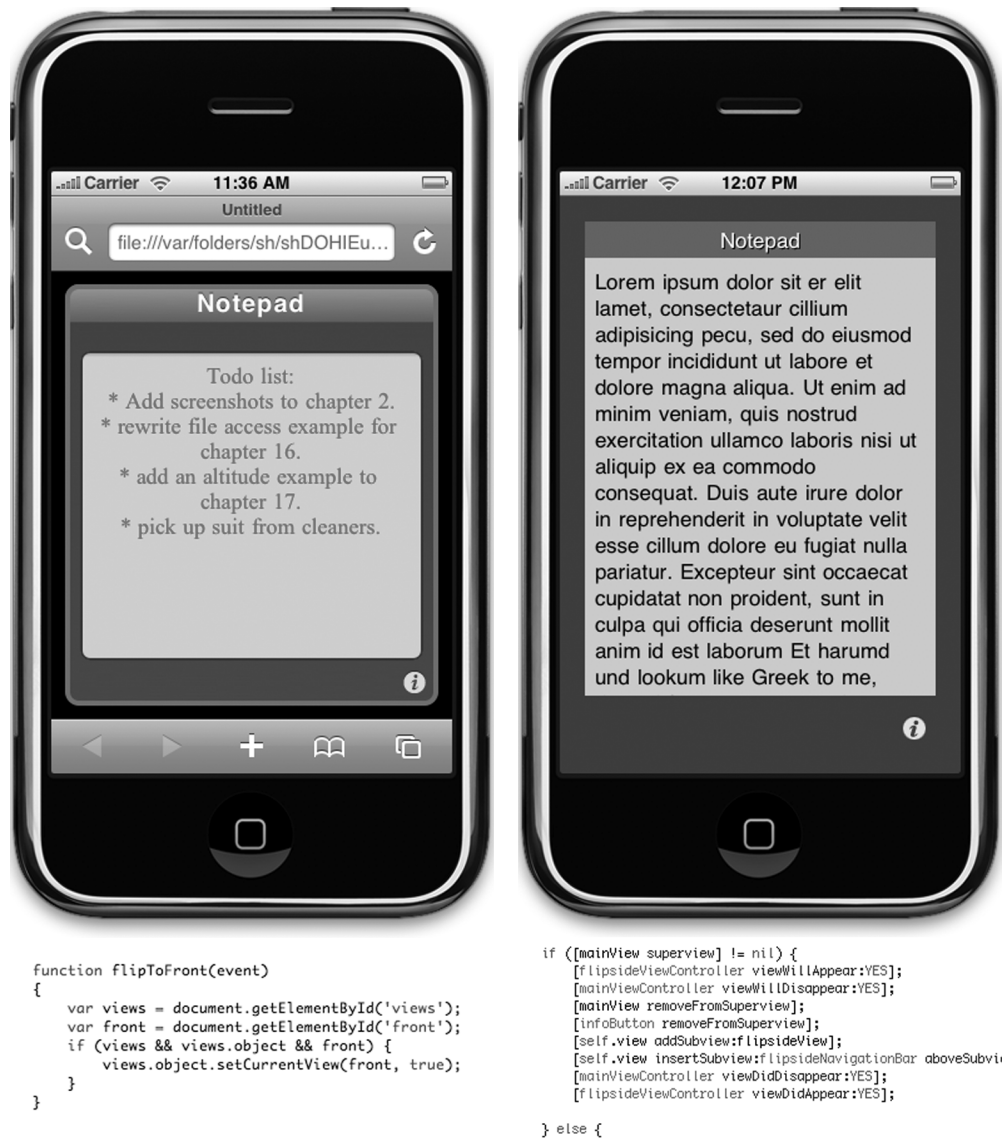


Figure 2.1 Though not identical, web programming (left) and SDK development (right) can produce similar output with similar underlying programming models.

JavaScript in the web example and in Objective-C in the SDK example. Each one produces an attractive rotating animation when the active screen is changed. We'll explain more about how the code for each programming style works in the chapters ahead, but we wanted to give you a preview now.

There's a further similarity between the two programs: each one features on its front page an editable "text view." This view can be used to display text, and can be edited using the iPhone's built-in keyboard.

We'll be talking a lot more about the similarities between the programming styles throughout this book. For now, we mainly want to highlight that neither style is a poor cousin: each has considerable functionality and can be used to create attractive UIs quickly and simply.

2.2 A central philosophy: the continuum of programming

Although we think that both styles of iPhone programming are useful, we're well aware that many of you are coming to this book from one of two directions. Either you're a web developer and want to learn how to optimize your web pages for viewing on the iPhone, or you're a programmer and you want to extend your C (or C++ or C# or J2ME) programming experience to the iPhone. We welcome you all, and we're certain that in this book you'll find a great introduction to your style of programming on the iPhone. Even if you've never programmed before but are simply intrigued by this new iDevice that you have, you'll be able to follow this book right through its introductory tutorials. Whichever route you've taken here, we encourage you to read the entire book, because we believe that by understanding—and using—the entire continuum of iPhone programming you'll literally get twice as much out of the experience.

For the *web developer*, we're going to tell you everything you need to know about the specifics of iPhone programming, including digging into some web features that you're probably not familiar with, such as the newest WebKit features, iUI, and Canvas. We hope you'll keep reading from there, as our material on the SDK is all quite introductory, and even if you've never worked with a compiled programming language, you can use this book to move up to SDK programming.

Chapter 9 is the foundation of this transition. It'll provide you with the basis of how a compiled programming language differs from the PHP, Perl, or Ruby on Rails that you might be familiar with. Starting from there, you should be able to learn the same lessons as a more experienced C programmer when we dive into the SDK itself.

For the *SDK programmer*, we're going to provide you with a complete introduction to all of the SDK's coolest features, including all of the unique iPhone features that we've already touched on, such as its GPS, its accelerometers, and its unique input device. However, we hope you won't consider SDK programming the be-all, end-all of iPhone software. We feel there are genuinely places where web development is a better choice.

We'll argue the reasons that you might select web development over SDK right here, in this chapter. Even if you opt not to do any web development, we invite you to

at least skim through the web chapters, because we end each with a look at the lessons that web development can teach you about the iPhone as a whole.

Generally, no matter what type of programmer you are, you should think of this book as a toolbox. It's divided into two large compartments, but every tool within has the same goal: creating great iPhone programs. You just need to make sure that you're always using the right tool for the job at hand.

2.3 Advantages and disadvantages

Each of the web and SDK development models has its own advantages and disadvantages. We've summarized the biggest advantages for each model of development in table 2.1.

Table 2.1 Each model of development has its own advantages; for any project, you should use the model that best matches your needs.

Web development advantages	SDK advantages
Ease of development	Sophisticated development environment
Ease of first-time user access	Improved language depth
Rapid deployment	Integration with iPhone libraries
Automated updating	Improved graphics libraries
Access to dynamic data	Ease of continued user access
Access to existing web content	No downloading
Offline server access	Native speed
Integration with external web content	Improved privacy
Access to other users	Built-in economic model

We're going to look at each of these topics in more depth, starting with the web side of things. Afterward we'll offer some more precise suggestions on which programs we think should be developed using each type of programming.

2.3.1 Web development

The bywords of web development are *simplicity*, *dynamism*, and *globalization*.

Simplicity. Frankly, web development is simpler than using a low-level programming language like C. Although some of the dynamic programming languages are pretty sophisticated, you don't usually have to worry about things like memory management or even (for the most part) object modeling. If you're just outputting plain data, the ease factor in web development goes up by a factor of at least 10 times when compared to doing the same thing using the SDK's tables and other data outputs. Beyond that, when you're done developing a web program, all you need to do is upload your

pages to your server. There are no hoops to jump through (other than those your individual company might impose).

It's also a lot simpler for users to begin working with your web program. They're much more likely to randomly view a web page than to pay to purchase your SDK program from the iPhone App Store, and thus it's a great way to attract many more users.

Dynamism. Hand in hand with that is the fact that you can literally update your program at any time. You don't have to worry about when or how your users will get a new program. You just change your server's source code, and the next time your users access the page (or, in the worst case, the next time they update their cache), they'll see all your bug fixes and other changes.

Similarly, you can constantly give users access to your newest data. Whereas data stored within an SDK program is more likely to be static, for a web program a user's view of data changes whenever you update it. This leads us to the next strength...

Globalization. When you create a web-based iPhone program, you become part of a global network that starts at your own server. This has a lot of advantages.

First, it means you can just create one program for use by both iPhone and desktop users (or, at worst, just one back-end, if you decide to program separate front-ends for each class of users). This will improve usability for your users if there's data they want to access (or even update) from both their desktop and their iPhone.

Second, it gives you direct access to your web server. This is particularly important because the iPhone keeps you from running programs in the background due to energy concerns. If you need to keep something running, you can hand it off to your server.

Third, it gives you rapid access to the rest of the web through URLs, Really Simple Syndication (RSS) feeds, and other data links. Granted, these advantages could be mimicked using web access from the SDK. However, if you're depending on the internet anyway, at some point you should just go full out and write your program for the web, allowing you to take advantage of the other strengths listed here.

Fourth, it's also a lot easier for your users to interact with other users, which might be particularly important for chats or multiplayer games.

Looking across the internet, there are numerous examples of superb iPhone web apps programmed by companies who felt that it was the superior medium. Google in particular is developing an entire suite of web apps, mimicking many of their desktop-centric web pages. The Hahlo twitter client is another example of a great iPhone program developed for the web: it makes use of online data but presents it in a tabbed, river format that should look somewhat familiar to iPhone users despite being a web app.

Overall, we feel that web development is the superior programming tool to use when

- You're programming a simple data-driven interface, with little need for the iPhone's bells and whistles
- You're expecting frequent updates to the program's data or to the program itself
- You're depending on the internet for data, users, or other access

2.3.2 SDK development

The bywords of SDK development are *sophistication*, *accessibility*, and *monetization*.

Sophistication. Just as web programs are a lot easier to develop and deploy, the flip side is that SDK programs allow for improved depth. This offers two important advantages.

First, you have greater depth implicit to the SDK itself. The development environment is a wonder of error reporting, complemented by well-integrated documentation and a built-in profiling package. This sophistication is also represented in the programming language, which is a well-considered object-oriented language. Although dynamic web languages are moving in that direction, Objective-C has already been doing OOP for over 20 years. Given that some sophisticated web languages like Java aren't available on the iPhone, the SDK's depth differentiates it that much more from the web.

Second, this depth shows up in the frameworks that you'll have access to when you use the SDK. They'll give you much deeper access to the iPhone's internals than any web page could. Apple has made some unique events, like orientation change, and some multifinger gestures available to web developers through the WebKit, but if you want to use the address book or GPS or want to take a deeper look at the accelerometers, you have to use the SDK. You can also access better graphics when you use the SDK.

Accessibility. Once users buy a program, it's available on their iPhone screen. Although a similar function is available for saving Safari bookmarks, it's likely only a percentage of users will take advantage of it.

That program is also usable wherever a user is, whether that be inside a subway tunnel or in a cell phone dead zone. The iPhone has an always-on internet, yet there are inevitably times and places when it's not available—but a native program will be. Even an occasionally connected application might benefit from being native to the iPhone, as they can provide constant access to "old" data.

This goes hand in hand with the fact that the applications will always be running at iPhone speed, not constrained by the internet, a remote server, or some combination of the two.

Finally, because the program is sitting on their iPhone, users might feel more comfortable about using it to save their personal records than they would be if they knew the data was going out to the internet and thus potentially vulnerable (though the iPhone doesn't actually guarantee that its information won't go out onto the net, thus making this a mainly visceral advantage).

Monetization. We don't want to be entirely mercenary, but at the same time we think it's important to note that Apple is making it easy to sell your iPhone SDK programs through their iPhone App Store. Certainly you could depend on advertisements or even subscriptions for a web-developed program, but you don't have to worry about any of that if you write a program using the SDK.

At the time of this writing, the iPhone App Store is just getting started, but it's already shown off some excellent programs that clearly work better using the SDK than the web, primarily due to sophisticated graphics. This mirrors the web programs

that we've already seen, like those designed by Google, which are excellent due to their web-based origins.

Overall, we feel that SDK development is the superior programming tool to use when

- You're creating a particularly sophisticated program or suite of programs
- You need to use any function (such as the address book, the accelerometers, the GPS, the camera, or animation) that isn't well supported on the web
- You want to monetize your program but don't have the web infrastructure to do so

2.3.3 *To each program its platform*

At this point you should be able to make your own decisions about which of the two programming platforms will best support the software that you want to write. To supplement your own thinking, we've listed a variety of programs in table 2.2, sorted by the method we'd suggest for programming them.

Table 2.2 Different programs can each benefit from one of the main developmental models.

Web programs	SDK programs
Chat programs	Accounting
Data wrappers (general)	Address books and other contacts
Data wrappers (frequently changing data)	Animated graphics
Games (simple multiplayer)	Data wrappers (critical information)
Inventory lists	Games
Schedules (multiperson)	Location-aware programs
Schedules (services)	Photo/graphic programs

The line between web development and SDK programming doesn't have to be as black and white as we make it out here. As you'll see momentarily, models exist for integrating the two types of development. But before we get there, we'd first like to outline our models for programming using just one of the programming packages, as these web development and SDK models will form the basis for most of this book.

2.4 *Stand-alone iPhone development*

The topic of iPhone development isn't just as simple as web versus SDK. We've divided those topics further by highlighting six ways you can develop iPhone pages using the web and two ways you can develop iPhone pages using the SDK. These methods are all summarized in table 2.3, complete with chapter references.

As shown in table 2.3, these models of development ultimately form the skeleton of this book. We'll summarize the methodologies here, and then expand on them in upcoming chapters.

Table 2.3 This book includes details on eight ways that you can program for the iPhone.

Method	Type	References
iPhone incompatible	Web	Brief mentions only
iPhone compatible	Web	Brief mentions only
iPhone friendly	Web	Chapters 3, 8
iPhone optimized	Web	Chapters 3, 8
iPhone web apps	Web	Chapters 4–6, 8
Dashcode	Web	Chapter 7
SDK native apps	SDK	Chapters 10–19
SDK web apps	SDK	Chapter 20

2.4.1 Web development models

We classify web pages into three types:

- Those that haven't received any special development work for the iPhone
- Those that have received normal development work
- Those that have received development work using Apple's Dashcode program

NONDEVELOPED WEB PAGES

The purpose of this book is to talk about how to develop web pages and programs for the iPhone, yet web developers will have a baseline that they're starting from: those web pages that haven't been developed with the iPhone in mind but that are viewed there anyway. We divide these non-developed pages into two general categories.

A web site is *iPhone incompatible* if the web developer has done no work to improve the site for the iPhone and it doesn't work very well. This may be due to a dependence on unsupported plug-ins like Flash and Java. It could also be due to the way in which CSS is used: a web site that uses a microscopically tiny font that's absolutely defined will be unreadable on the iPhone unless a user zooms and scrolls a lot. Very wide columns without viewport tags are another common reason for incompatibility. If you've got an iPhone-incompatible site, that might be why you picked up this book: to figure out how to improve the look and feel of your existing web site for the iPhone.

A web site is *iPhone compatible* if the web developer has done no work to improve the site for the iPhone and it sort of works anyway. It doesn't necessarily look that great, but at least it's usable, and so iPhone users won't actively avoid it. If you've got an iPhone-compatible site, it's probably because you're already making good use of CSS and you understand a core concept of HTML: that it's a markup language in which you can't accurately control how things are viewed.

DEVELOPED WEB PAGES

As an iPhone web developer, however, you want to do better than these undeveloped web pages. That's what the first part of this book is about (though we'll also touch

on some SDK tools as we go). We categorize iPhone-specific web development into three types.

A web site is *iPhone friendly* if the web developer has spent a day of time—or maybe less—improving the experience for iPhone users. This involves simple techniques such as using the viewport tag, making good use of columns, using well-designed style sheets, and making use of iPhone-specific links. The basic techniques required to create iPhone-friendly web sites will be covered in depth in chapter 3.

A web site is *iPhone optimized* if the web developers have gone all-out to create pages that look great on the iPhone. They've probably inserted commands that deal with the iPhone chrome and have thought about iPhone gestures. They may link in unique iPhone style sheets when the device is detected. All of this requires a great understanding of how the iPhone works, but also provides a better experience for users. Many times the view that an iPhone user sees on an iPhone-optimized web site may be dramatically different than that experienced by a desktop user; despite the iPhone's claim to be a fully featured browser, there are some things that just don't work as well, and an iPhone-optimized site recognizes that. The slightly more advanced techniques needed to develop iPhone-optimized web sites will also be discussed in chapter 3.

Finally, some web sites may actually be *iPhone web apps*. These are web pages that are intended only to work on the iPhone, and in fact will probably look quite ugly if viewed from a desktop browser. We'll talk about using the functions of Apple's advanced WebKit in chapter 4. Then we'll discuss how to make pages that look like iPhone natives apps in chapter 5, including a look at the iUI library. Finally we'll look at the Canvas graphic library in chapter 6.

DASHCODE PAGES

As part of the SDK, Apple distributes a tool called *Dashcode*. It can be used to package JavaScript and HTML into a format specifically intended for the iPhone. Dashcode can also be used as development platform for many of the web app libraries and presages some of the functionality of the SDK. We'll cover it in chapter 7.

2.4.2 SDK development models

iPhone web apps represent a transition for iPhone developers. When you're engaging in simpler types of iPhone development—making existing web sites iPhone friendly or iPhone optimized—you're just considering the iPhone as one of many platforms that you're supporting. However, when creating iPhone web apps, you're instead developing specifically and exclusively for the iPhone.

Some developers will be happy staying with the web development techniques discussed in the first half of this book. Other developers will want to take the next step, to learn the SDK for those programs that could be better developed using that toolkit. Chapter 8 will offer some advice on making the jump from web development to the SDK—even if you've never programmed in a compiled programming language before. Then, the latter half of the book covers the SDK in depth.

THE SDK CONTINUUM

SDK development is even more of a continuum than web programming. As you learn more about Apple’s SDK frameworks, you’ll gradually add new tools that you can program with. Nonetheless, we’ve outlined two broadly different sorts of SDK programming.

SDK native apps are those SDK applications that make use only of the iPhone itself, not the internet and the outside world. Even given these restrictions, you can still write numerous complex native programs. We’ll start things off with a look at the basic building blocks of the SDK: Objective-C and the iPhone OS (chapter 10), Xcode (chapter 11), Interface Builder (chapter 12), view controllers (chapters 13 and 15), and actions and events (chapter 14). Then we’ll delve into the SDK toolkit, talking about data (chapter 16), positioning (chapter 17), media (chapter 18), and graphics (chapter 19).

SDK web apps are those SDK applications that also use the iPhone’s always-on internet. In many ways they bring us full circle as we look at the web from a different direction. Although chapter 20 mainly covers how to access the internet using the iPhone, it’s also what opens the door to the unique ways in which you can integrate your web and SDK work. That integration can appear in several forms.

2.5 Integrated iPhone development

The purpose of this chapter has been to delineate the two sorts of iPhone development—using the web and the SDK. Thus far we’ve talked quite a bit about what each style of programming does best, and we’ve even outlined stand-alone development methodologies. We’ve also touched on the fact that quite often you might want to use the two styles of programming *together*.

This is the only time that we’re going to look at these ideas in any depth, but we invite you to think about them as you move through this book, to see how you can use the strengths of both the web *and* the SDK to your advantage. Table 2.4 summarizes our three integrated development methods, highlighting the strengths that each takes advantage of.

We’re going to finish this chapter by exploring the three types of integrated development in more depth.

Table 2.4 Writing web programs using both the web and the SDK can let you take advantage of the strengths of both mediums (and all the contents of this book).

Method	Web strengths	SDK strengths
Mirrored development	Ease of first-time user access	Built-in economic model
Mixed development	Any strengths, especially: Rapid deployment Access to dynamic data	Any strengths, especially: Ease of continued access Native speed
Client-server development	Access to dynamic data Offline server access	Improved language depth Integration with libraries Native speed

2.5.1 *Mirrored development*

It's obviously easier to get users to try out a free but limited version of your software than it is to get them to purchase a more complete version. The business model of upgrading users from free to premium versions of software has been used extensively, with "freemium" being the latest buzzword. There are two ways you could create a freemium model for your software.

First, you could do what a lot of developers are already doing and offer a free trial version of your software on the iPhone App Store. This has the advantage of putting the software in the place that people are looking for software, but has the disadvantage that your application could get lost amid the hurly-burly of the store.

Second, you could create a version of your software for the web, using web app technologies. We think this model is particularly useful for those of you who have existing web pages that might already be drawing users to them in more highly targeted ways than the iPhone App Store could. Then, after releasing a limited version of your application over the web using techniques like the WebKit, iUI, and Canvas, you also release a feature-complete version of your application through the App Store using the SDK.

Although we've highlighted the economic reasons for this sort of mirrored development, it's possible that web sites might decide to extend existing web apps to include features not available in their web-based application. If so, then you'll have a clear delineation between what the programs include: the SDK will uniquely include those features that weren't available through the web, like location-aware and orientation-aware data.

2.5.2 *Mixed development*

In a mixed development environment, instead of making the web a subset of your SDK work, you're looking at an overall project and deciding to program some of it using the web and some of it using the SDK. This can be a chaotic methodology if not managed carefully, but it gives you the best opportunity to use the strengths of each sort of development. We find it most likely to work when you have two classes of users or two classes of programmers.

On the user side, a good example might be a situation where you have administrative users and end users. Assume you're managing some data project. The data input methods used by your administrators don't necessarily have to look great. You can develop them quickly using the web and then your administrators can choose whether to input data from their iPhones or from their desktops. Conversely, you want to provide a great user experience for your end users, so you take advantage of the iPhone's native graphical and animation features to output your data in interesting ways.

On the programmer side, you might simply have developers who are more comfortable in either the web or Objective-C arena. A mixed development project allows you to use not only the strengths of the individual programming methods but the strengths of your individual programmers as well.

The exact way in which you do mixed development will depend on the specifics of your project, but there should be real opportunities to take advantage of each programming style's strengths without much redundancy.

2.5.3 Client-server development

The final type of integrated iPhone development is the most powerful—and also one that's already in use on your iPhone, whether or not you know it. Client-server development combines back-end web development and front-end SDK development in a fairly typical client-server model. This is effectively what is done by existing iPhone programs such as Maps, Stocks, and YouTube, all of which pull data from the internet and display it in attractive ways while also taking advantage of the iPhone's unique capabilities.

On the one hand, you don't need a lot of the details of web development as presented in this book to create a data back end. Instead you can depend on your existing Perl, PHP, or Ruby on Rails code and use it to kick out RSS or some other data feed that your SDK software can easily pick up. On the other hand, if you're already doing that much development on the web side of things, creating a second web-based interface for iPhone users should be trivial.

Thus, a client-server development environment can give you the excuse to use either of the other integrated development strategies that we suggested.

2.5.4 Last thoughts on integration

We know some of you will be gung-ho to create a program that integrates SDK and web work all on your own, but we also recognize that in many larger companies it's likely that different people will undertake different tasks, with some developers doing the web side of things and some instead doing SDK programming.

If this is the case—and particularly if you're creating a suite of programs that integrate SDK and web development alike—don't be afraid to share the knowledge in this book as well as what you learn on your own. The iPhone is a new and innovative development platform. Its unique features create lots of opportunities for interesting work and also some potential gotchas as well. Although we're offering as many lessons as we can, we're sure there's a lot more you'll learn while you're out programming in the real world, and as with all knowledge, the best use is to share it.

2.6 Summary

The iPhone encourages two dramatically different methodologies for programming. You can either use traditional web programming techniques—with quite a few special features to account for the iPhone's unique elements—or you can dive into the intricacies of SDK development.

We've divided this book in two: half on web development and half on SDK development. There are two main reasons for this.

First, we wanted it to be an introduction to the entire world of iPhone development. No matter which path you want to take, this is the right book to get you started

and to bootstrap you up to the point where you can program on your own. If you're a web developer without much C programming experience, you can even follow the entire path of the book, which will move you straight from the world of the web to the world of the SDK.

Second, we believe that good reasons exist to program in both environments. It's not merely a matter of expertise on the part of the programmer, but instead of capability on the part of the programming languages. There's a lot of simple stuff that's ten times easier to do in HTML, but similarly some complex stuff that's only possible to do using the SDK.

With the understanding of this book's bifurcation clearly in hand, we're now ready to jump into the first compartment of our toolbox; web development, where we'll start to explore what's necessary to make web pages look great on the iPhone device.

iPhone IN ACTION

Christopher Allen and Shannon Appelcline

Free ebook
SEE INSERT

The iPhone explodes old ideas of a cell phone. Its native SDK offers a remarkable range of features including easy-to-build graphical objects, a unique navigation system, and a built-in database, all on a location-knowledgeable device. Websites and web apps can now behave like native iPhone apps, with great network integration.

iPhone in Action is an in-depth introduction to both native and web programming for the iPhone. You'll learn how to turn your web pages into compelling iPhone web apps using WebKit, iUI, and Canvas. The authors also take you step by step into more complex Objective-C programming. They help you master the iPhone SDK including its UI and features like accelerometers, GPS, the Address Book, SQLite, and many more. Using Apple's standard tools like Dashcode, Xcode, and Interface Builder, you'll learn how to best use both approaches: iPhone web and SDK programming.

This book is intended as an introduction to its topics. Proficiency with C, Cocoa, or Objective-C is helpful but not required.

What's Inside

- A comprehensive tutorial for iPhone programming
- Web development, the SDK, and hybrid coding
- Over 60 web, Dashcode, and SDK examples

About the Authors

A technologist and entrepreneur, **Christopher Allen** is a popular speaker, co-founder of iPhoneDevCamp, and host of its Hack-a-thon. iPhone consultant **Shannon Appelcline** is a writer and programmer who develops social web software and has written numerous books.

For online access to the authors, code samples, and a free ebook for owners of this book, go to www.manning.com/iPhoneinAction

"The entry to the world of iPhone."

—Aiden Montgomery, Wile Ltd.

"If you're new to iPhone development, this is *your* book!"

—Larry C. Whipple
Mobile Productivity, Inc.

"Get this book. It's pure gold."

—Martijn Dashorst
Author of *Wicket in Action*

"The quick & easy guide."

—Premkumar Rajendran
HCL Technologies

"The only book on iPhone development I will ever need."

—Rama Krishna Vavilala, Author
of *ASP.NET AJAX in Action*

ISBN-13: 978-1933988863
ISBN-10: 193398886X



9 781933 988863



MANNING

US/CAN \$39.99 [INCLUDING EBOOK]