1

Introducing Safari/WebKit Development for iPhone 3.0

The introduction of the iPhone and the subsequent unveiling of the iPod touch have revolutionized the way people interact with handheld devices. No longer do users have to use a keypad for screen navigation or browse the Web through "dumbed down" pages. These mobile devices have brought touch screen input, a revolutionary interface design, and a fully functional Web browser right into the palms of people's hands.

Seeing the platform's potential, all the segments of the developer community jumped on board. Although native applications may receive most of the attention, you can still create apps for iPhone without writing a single line of Objective-C, the programming language used to develop native iPhone apps. In fact, iPhone's WebKit-based browser provides a compelling application development platform for Web developers who want to create custom apps for iPhone using familiar Web technologies.

Each subsequent release of the iPhone OS and Safari on iPhone has put increased power into the hands of Web developers, and as I'll discuss shortly, the iPhone OS 3.0 release is no exception.

Discovering the Safari/WebKit Platform

An iPhone Web application runs inside of the built-in Safari browser that is based on Web standards, including these:

- HTML/XHTML (HTML 4.01 and XHTML 1.9, XHTML mobile profile document types)
- CSS (CSS 2.1 and partial CSS3)

- JavaScript (ECMAScript 3, JavaScript 1.4)
- AJAX (for example, XMLHTTPRequest)
- SVG (Scalable Vector Graphics) 1.1
- HTML 5 media tags
- Ancillary technologies (video and audio media, PDF, and so on)

Safari on iPhone and iPod touch (which I refer to throughout the book as simply *Safari*) becomes the platform upon which you develop applications and becomes the shell in which your apps must operate (see Figure 1-1).

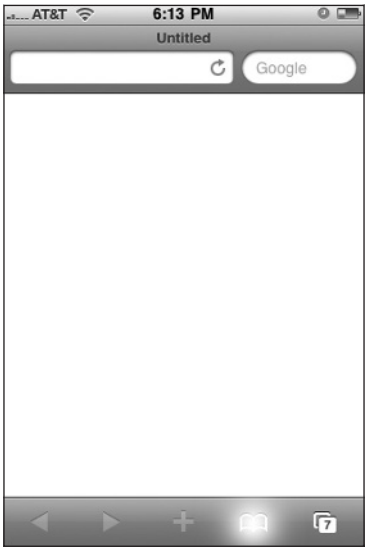

Figure 1-1: Safari user interface

Safari is built with the same open source WebKit browser engine as Safari for OS X and Safari for Windows. However, although the Safari family of browsers is built on a common framework, you'll find it helpful to think of Safari on iPhone as a close sibling to its Mac and Windows counterparts, not an identical twin to either of them. Safari on iPhone, for example, does not provide the full extent of CSS or JavaScript functionality that its desktop counterpart does.

In addition, Safari on iPhone provides only a limited number of settings that users can configure. As Figure 1-2 shows, users can turn off and on support for JavaScript, plug-ins, and a pop-up blocker. Users can also choose whether they want to always accept cookies, accept cookies only from sites they visit, or never accept cookies. A user can also manually clear the history, cookies, and cache from this screen.

Figure 1-2: Safari on iPhone preferences

Quite obviously, native apps and Web apps are not identical — both from developer and end-user standpoints. From a developer standpoint, the major difference is the programming language — utilizing Web technologies rather than Objective-C. However, there are also key end-user implications, including these:

- **Performance:** The performance of a Safari-based Web application is not going to be as responsive as a native compiled application, both because of the interpretive nature of web scripting as well as the fact that the application operates over Wi-Fi and 3G networks. (Remember, iPod touch supports Wi-Fi access only.) However, in spite of the technological constraints, you can perform many optimizations to achieve acceptable performance. (Several of these techniques are covered in Chapter 13, "Bandwidth and Performance Optimizations.")

  Table 1-1 shows the bandwidth performance of Wi-Fi, 3G, and the older EDGE networks.

Table 1-1: Network Performance

Network	Bandwidth
Wi-Fi	54 Mbps
3G	Up to 7.2 Mbps
EDGE	70–135 Kbps, 200 Kbps burst

- **Launching:** All native applications are launched from the main Home screen of the iPhone and iPod touch (see Figure 1-3). In the original iPhone OS release, Apple provided no way for Web apps to be launched from here, requiring Web apps to be accessed from the Safari Bookmarks list. Fortunately, subsequent releases of the iPhone OS have given users the ability to add "Web Clip" icons for their Web apps (such as the Cup-O-Joe Web app in Figure 1-4).

Figure 1-3: Built-in applications launch from the main Home screen

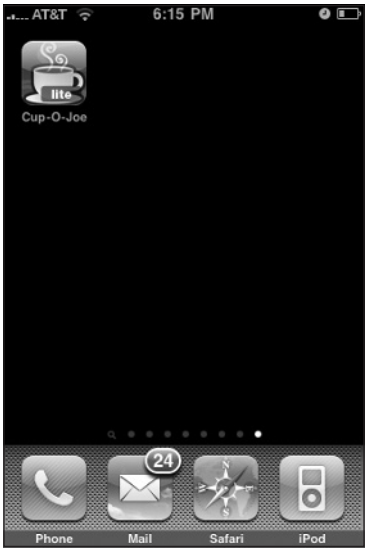

Figure 1-4: Web applications can also be included on the Home screen

- **User interface (UI):** Native iPhone applications often adhere to Apple UI design guidelines. Fortunately, using open source frameworks and standard Web technologies, you can closely emulate native application design using a combination of HTML, CSS, and JavaScript. Figures 1-5 and 1-6 compare the UI design of a native application and a Safari-based Web application.

What's more, iPhone OS enables you to hide all Safari browser UI elements through meta tags, enabling you to essentially emulate the look and feel of a native app. (See Figure 1-7.)

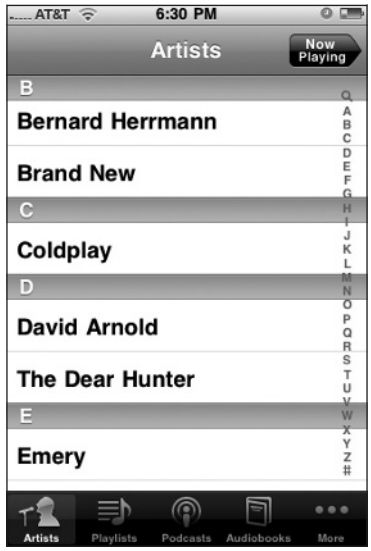

Figure 1-5: Edge-to-edge navigation pane in the native app

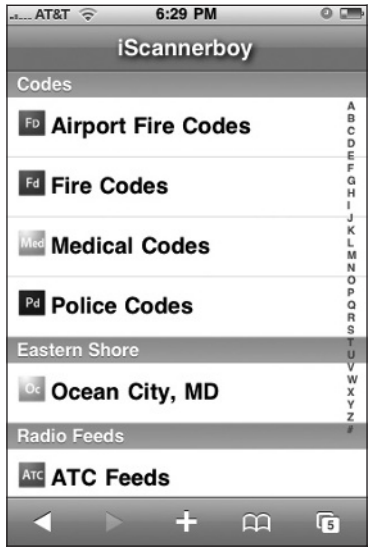

Figure 1-6: Edge-to-edge navigation pane in a custom Web application

Figure 1-7: Web app or native app? It's hard to tell.

What's New in iPhone OS 3.0 for Web App Developers

There are several new capabilities available to Web app developers with the release of iPhone OS 3.0 and Safari 4.0 iPhone. These are highlighted here:

- **Geolocation:** Safari on iPhone now supports HTML 5 geolocation capabilities, which enable JavaScript to interact with iPhone's GPS service to retrieve the current location of the iPhone (see Figures 1-8 and 1-9). As a result, you can create apps that can broadcast the location of a GPS-enabled iPhone.

  Google is using this capability with its Latitude service for sharing your location with your friends.

- **HTML 5 Media Tags:** The newest release of Safari on iPhone supports HTML 5 `video` and `audio` tags for embedding video and audio content in Web pages. These new elements eliminate the need for complicated `embed` and `object` tags for embedding multimedia elements and allow you to utilize a powerful JavaScript API. What's more, because iPhone doesn't support Flash, you can use the `video` tag to embed QuickTime MOV files.

  Safari is the first major browser to provide full support for HTML 5 media tags; therefore, you have to be careful in their usage on standard Web sites because other browsers may not support it yet. However, because you are creating an app specifically for the iPhone, you can make full use of these tags.

- **CSS animation and effects:** The new release of Safari supports *CSS animation*, which enables you to manipulate elements in various ways, such as scaling, rotating, fading, and skewing. Safari on iPhone also supports *CSS effects,* which enable you to create gradients, masks, and reflections entirely through CSS.

- **SVG:** SVG (or Scalable Vector Graphics) is a new XML-based format for creating static and animated vector graphics. With SVG support, Safari on iPhone not only provides a way to work with scalable graphics, but actually provides a technology that could replace the need for Flash to create animated media.

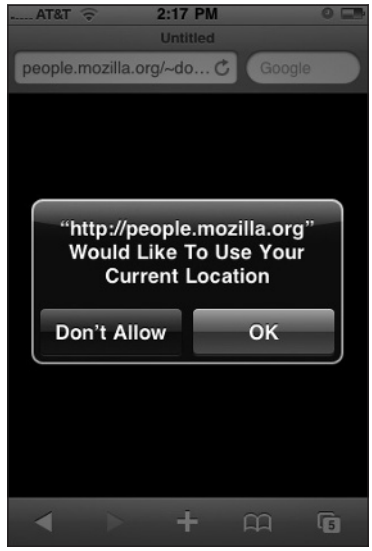

Figure 1-8: Users are asked to confirm GPS location services support

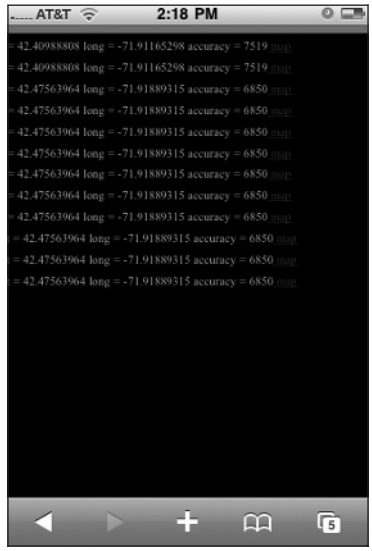

Figure 1-9: Test Web app that displays geolocation info in real time

Four Ways to Develop Web Apps for iPhone

A Web application that you can run in any browser and an iPhone Web application are certainly made using the same common ingredients — HTML, CSS, JavaScript, and AJAX — but they are not identical. In fact, there are four approaches to consider when developing for iPhone:

- **Level 1 — Fully compatible Web site/application:** The ground-level approach is to develop a Web site/app that is "iPhone/iPod touch-friendly" and is fully compatible with the Apple mobile devices (see Figure 1-10). These sites avoid using technologies that the Apple mobile devices do not support, including Flash, Java, and other plug-ins. The basic structure of the presentation layer also maximizes the use of blocks and columns to make it easy for users to navigate and zoom within the site. This basic approach does not do anything specific for iPhone users but makes sure that there are no barriers to a satisfactory browsing experience. (See Chapter 12, "Enabling and Optimizing Web Sites for the iPhone and iPod Touch," for converting a Web site to be friendly for iPhone users.)

Figure 1-10: The site is easy to navigate

- **Level 2 — Web site/application optimized for Safari:** The second level of support for iPhone is to not only provide a basic level of experience for the Safari on iPhone user, but to provide an optimized experience for those who use Safari browsers, such as utilizing some of the enhanced WebKit CSS properties supported by Safari.
- **Level 3 — Dedicated iPhone/iPod touch Web site/application:** A third level of support is to provide a Web site tailored to the viewport dimensions of the iPhone and provide a strong Web browsing experience for Apple device users (see Figures 1-11 and 1-12). However, although these sites are tailored for iPhone viewing, they do not always seek to emulate Apple UI design. And, in many cases, these are often stripped-down versions of a fuller Web site or application.

Figure 1-11: Amazon's mobile site

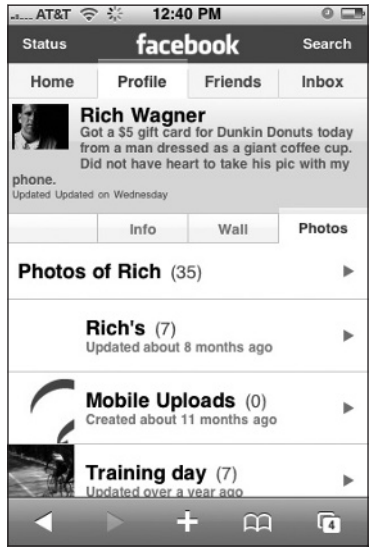

Figure 1-12: Facebook's dedicated site for iPhone

- **Level 4 — Native-looking iPhone Web application:** The final approach is to provide a Web application that is designed exclusively for iPhone and closely emulates the UI design of native applications (see Figure 1-13). One of the design goals is to minimize users' awareness that they are even inside of a browser environment. Moreover, a full-fledged iPhone application will, as is relevant, integrate with iPhone-specific services, including Phone, Mail, and Google Maps.

Therefore, as you consider your application specifications, be sure to identify which level of user experience you want to provide iPhone users, and design your application accordingly. This book focuses primarily on developing native-looking Web applications.

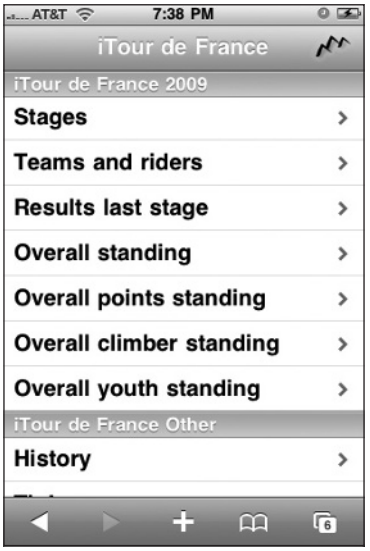

Figure 1-13: iPhone Web app that looks like a native app

The Finger Is Not a Mouse

As you develop applications for iPhone, one key design consideration that you need to drill into your consciousness is that *the finger is not a mouse.* On the desktop, a user can use a variety of input devices — such as an Apple Mighty Mouse, a Logitech trackball, or a laptop touchpad. But, on-screen, the mouse pointer for each of these pieces of hardware is always identical in shape, size, and behavior. However, on iPhone, the pointing device is always going to be unique. Ballerinas, for example, will probably input with tiny, thin fingers, whereas NFL players will use big, fat input devices. Most of the rest of us will fall somewhere in between. Additionally, fingers are not nearly as precise as mouse pointers are, making interface sizing and positioning issues very important, whether you are creating an iPhone-friendly Web site or a full-fledged iPhone Web application.

Also, finger input does not always correspond to mouse input. A mouse has a left-click, right-click, scroll, and mouse move. In contrast, a finger has a tap, flick, drag, and pinch. However, as an application developer, you will want to manage what types of gestures your application supports. Some of the gestures that are used for browsing Web sites (such as the double-tap zoom) are actually not something you want to support inside of an iPhone Web app. Table 1-2 displays the gestures that are supported on iPhone as well as whether this type of gesture is typically supported on a Web site or a full Web application.

Table 1-2: Finger Gestures

Gesture	Result	Web site	App
Tap	Equivalent to a mouse click	Yes	Yes
Drag	Moves around the viewport	Yes	Yes
Flick	Scrolls up and down a page or list	Yes	Yes
Double-tap	Zooms in and centers a block of content	Yes	No
Pinch open	Zooms in on content	Yes	No
Pinch close	Zooms out to display more of a page	Yes	No
Touch and hold	Displays an info bubble	Yes	No
Two-finger scroll	Scrolls up and down an `iframe` or element with the CSS `overflow:auto` property	Yes	Yes

Limitations and Constraints

Because iPhone is a mobile device, it is obviously going to have resource constraints that you need to be fully aware of as you develop applications. Table 1-3 lists the resource limitations and technical constraints. What's more, certain technologies (listed in Table 1-4) are unsupported, and you need to steer away from them when you develop for iPhone and iPod touch.

Table 1-3: Resource Constraints

Resource	Limitation
Downloaded text resource (HTML, CSS, JavaScript files)	10MB
JPEG images	128MB (all JPEG images over 2MB are subsampled — decoding the image to 16x fewer pixels)
PNG, GIF, and TIFF images	8MB (in other words, width*height*4<8MB)
Animated GIFs	Less than 2MB ensures that frame rate is maintained (over 2MB, only first frame is displayed)
Nonstreamed media files	10MB
PDF, Word, Excel documents	30MB and up (very slow)
JavaScript stack and object allocation	10MB
JavaScript execution limit	5 seconds for each top-level entry point (`catch` is called after 5 seconds in a `try/catch` block)
Open pages in Mobile Safari	8 pages

Table 1-4: Technologies Not Supported by iPhone and iPod touch

Area	Technologies not supported
Web technologies	Flash media, Java applets, SOAP, XSLT, and plug-in installation
Mobile technologies	WML
File access	Local file system access
Security	Diffie-Hellman protocol, DSA keys, self-signed certificates, and custom x.509 certificates
JavaScript events	Several mouse-related events (see Chapter 7, "Handling Touch Interactions and Events")
JavaScript commands	`showModalDialog()`, `print()`
Bookmark icons	`ICO` files
HTML	`Input type="file"`, tool tips
CSS	Hover styles, `position:fixed`

Setting Up Your Development Environment on a Local Network

Because iPhone does not allow you to access the local file system, you cannot place your application directly onto the device. As a result, you need to access your Web application through another computer. On a live application, you will obviously want to place your application on a publicly accessible Web server. However, testing is another matter. If you have a Wi-Fi network at your office or home, I recommend running a Web server on your main desktop computer to use as your test server during deployment.

If you are running Mac OS X, you already have Apache Web server installed on your system. To enable iPhone access, go to System Preferences Sharing Services and turn the Personal Web Sharing option on (see Figure 1-14). When this feature is enabled, the URL for the Web site is shown at the bottom of the window. You'll use this base URL to access your Web files from your iPhone or iPod touch.

You can add files either in the computer's Web site directory (`/Library/WebServer/Documents`) or your personal Web site directory (`/Users/`*`YourName`*`/Sites`) and then access them from the URL bar on your iPhone (see Figure 1-15).

If your users experience crashing or instability inside Safari, direct them to clear the cache by clicking the Clear Cache button in the Safari Settings pane.

Figure 1-14: Turn on Personal Web Sharing

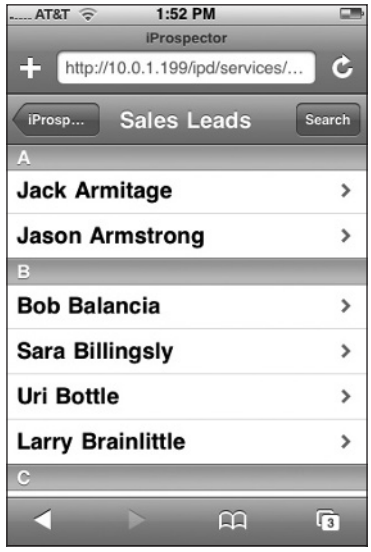

Figure 1-15: Accessing desktop files from an iPhone

Summary

In this chapter, you were introduced to iPhone Web application development and the Safari/WebKit browser. It began with a survey of key new features that are part of Safari on iPhone for iPhone OS 3.0. I then talked about four different ways to approach iPhone Web app development, ranging from an iPhone-compatible Web site to a Web app that emulates a native iPhone application. Then, after a discussion on the constraints and limitations associated with Safari, I closed by showing you how to set up your development environment for testing.