



War of the Mobile Browsers

Edwin A. Hernandez

Mobile and wireless communication networks are taking the desktop Internet experience beyond home and office environments. The differences between wired and wireless Internet are narrowing as telecommunication service providers enable 3G networks that allow mobile devices to access Web multimedia content. Pervasive IEEE 802.11 wireless access in coffee shops, museums, restaurants, and universities serves to enhance the experience. In the US, telecommunication carriers such as T-Mobile, AT&T, and Sprint provide 3G plans with speeds of up to 3 megabits per second (Mbps) covering wide metropolitan areas. For instance, South Florida is covered entirely from Jupiter to the Florida Keys.

As a consequence of wireless bandwidth and multimedia evolution, the mobile Web no longer has to differ from standard desktop environments. In the past, Wireless Applications Protocol (WAP) standards and Wireless Markup Language (WML) content

were the choice; in fact, WAP is still popular with many mobile carriers (www.wapforum.org/what/technical_1_2_1.htm). However, rich multimedia mobile phones are rendering the text-based Internet increasingly obsolete with CPUs, usually based on Advanced RISC Machine (ARM) clocks ranging from 100 to 250 MHz, and memory footprints of 128 MB.

MOBILE WEB PARADIGMS

Mobile content is continuously evolving to a richer experience in which standard HTML pages are delivered and formatted to fit a mobile device screen. Web applications, on the other hand, are increasingly popular as interactive HTML experiences. Currently, several mobile browsers provide an experience similar to that of the desktop. Developers use two paradigms in their design (see Table 1):

- **Server transcoding.** In this approach, a mobile client requesting a URL is

sent to a proxy server on the Internet, usually residing at the carrier premises. The proxy server, on behalf of the client, makes the request to the encoded server on the URL. HTML content retrieved is converted into a special or proprietary format and rendered by the requesting mobile browser. In general, these rendering engines are proprietary and reside on the server. The mobile browser might also include additional compression to support low-bandwidth scenarios.

- **Direct delivery.** The mobile device renders HTML directly from a Web server or a local file. Desktops use this approach and mobile browsers are increasingly using it as well. A rendering engine—for example, WebKit (www.webkit.org)—is embedded as part of the device firmware and in many cases these HTML rendering engines are open source. Pocket Internet Explorer (or Pocket IE) follows a similar approach and relies on rendering Compact HTML content. WebKit on the other hand, provides support for standard JavaScript, Asynchronous JavaScript and XML (Ajax), Cascading Style Sheets (CSS), and HTML—in general, those languages defined thus far as Web 2.0.¹

In the server transcoding paradigm, mobile browsers such as Opera and Skyfire (http://en.wikipedia.org/wiki/List_of_web_browsers) are written for native platforms like Java Micro-Edition (J2ME) and

TABLE 1
Mobile browser paradigms.

Server Transcoding	Direct Delivery
Server in the middle required Rendering done at the server, images or PNGs are delivered to the handset Multimedia is handled at the server Might use nonstandard rendering engines A simple thin-client approach can be implemented in several platforms Upgrades done at the server might not require reprovisioning	Direct access to the Internet JavaScript, Cascading Style Sheet, HTML support Might require plug-ins for multimedia content Standard rendering engines Model used on Android, iPhones; might require higher computation hardware

Windows Mobile. These browsers provide full page rendering and visualization for mobile devices that support these platforms. Their rendering engines use server-based computing power to generate compressed image files displayed by the mobile browser. Because these images are preprocessed at the server, content is irrelevant to the mobile browser. Thus, any handset can render RTSP (Real-time Streaming Protocol) videos and complex HTML content.

In general, server transcoding requires the implementation and management of a server infrastructure, and many Web applications might not work with this approach because of their reliance on Ajax or secure mechanisms unsuitable for mobile clients. Additionally, concerns over a single point of failure become more apparent in scenarios that could potentially limit the amount of requests a server makes, such as the webcast of a sporting event, or secure mobile banking applications. In either case, scalability and browser-level security must be addressed.

Conversely, the direct delivery approach to transcoding renders and processes mobile content at the handset using local resources. Today, several mobile browsers take advantage of this paradigm through the WebKit engine, including mobile Safari (iPhone and iPod touch), and more recently, Google Android (<http://code.google.com/android>). The following section briefly explains HTML rendering engines.

MOBILE WEB RENDERING ENGINES

Mobile browsers use rendering engines with specialized frameworks that parse HTML tags and draw images, text, and lines on a canvas area allocated for Web content. In addition to retrieving content from the Web or local repositories, these engines have become increasingly popular for managing state transitions,

providing APIs to control and access XML and HTML content, and properly displaying Web applications containing embedded JavaScript.

These engines don't depend on specific support from a desktop platform and have been ported to ARM-based architectures. WebKit and Gecko (http://en.wikipedia.org/wiki/List_of_web_browsers) comprise the most popular open source engines. Proprietary engines include Presto, Mercury (OpenWave), Netfront, and Internet Explorer (Trident). Other engines on the Web aren't as popular.

WebKit has emerged as the engine of choice for the mobile Web. It provides extensive support to HTML 2.0, including features such as Ajax, CSS, JavaScript, and a plug-in framework. WebKit has been enhanced with

Concerns over a single point of failure become more apparent in scenarios that could potentially limit the amount of requests a server makes.

touch-based interfaces and zooming capabilities to facilitate viewing and further interface small handsets with full-fledged HTML Web applications. Additionally, the same Ajax-based support, or Web 2.0 applications, can now asynchronously update personal mobile Web content.

WebKit is developed under Lesser General Public License (LGPL), as is Gecko from the Mozilla project, and both are available online. Mobile developers can compile ports of both WebKit and Gecko into ARM-based architectures with some modifications, provide libraries, and create applications using these rendering engines on Linux-based open platforms.

In addition to the mobile browser war, there's also a strong battle on the platform side for leadership, and as the browser and platform converge, the competition is becoming heated.

Webkit, for instance, has already been ported over to OS X (Apple's Tiger), thus providing support for Mac, iPod, and iPhone. WebKit has additional ports for Linux as part of the Google Android project, Qt, and Windows-based platforms, though it's not yet running on Windows CE/Smartphone.

Gecko (Firefox), on the other hand, is falling behind on features and capabilities compared to the iPhone and Google Android. However, Skyfire has been able to use Gecko as part of its server transcoding solution. Unlike WebKit and client-based rendering, Skyfire uses a proprietary protocol in all exchanges from its browser and rendering server. Skyfire's additional functionality includes Flash and Shockwave support at the server level.

MOBILE BROWSERS

Table 2 shows a summary of popular open source and commercially available mobile browsers. WAP-based browsers are still the most popularly deployed among handsets, and OpenWave continues to be the market leader in this area. WAP browsers are, in many cases, tied to proprietary device functionality, including features such as WAP-push messages and e-mail.

Many J2ME-enabled phones use graphical browsers, including the Opera browser. Opera has released several J2ME applications with ports that include Windows Mobile and Linux-based devices. Opera Mini 4 and Opera Mobile use a server-transcoding approach to render original HTML content at public or private servers, depending on what version is in use. Opera browsers rely heavily on server-transcoding. They also provide some scripting support, similar to JavaScript, by using "Opera widgets," but this is nonstandard and doesn't work across all browsers. Similarly, Netfront, Openwave, and Operacall run on BREW (Binary Runtime Environment) to support CDMA devices.

Netfront uses a proprietary, platform-independent rendering engine and includes full HTML support and

TABLE 2
Mobile browser comparison.

Name	Engine	Zoom in/Out	Touch	Multimedia	Widgets	Server/Cost	Platforms	Narrowband/Broadband
Openwave/Openweb	WAP2-Openwave	No	No	Limited to plug-in support	No	Yes/Yes Low	Windows, Linux, BREW, Real-time Operating System (ROS)	Yes/No
Opera Mini 4/Skyfire	Presto/Proprietary	Yes	Depends on Java	Depends on Java	Yes, Opera widgets	Yes/Yes Medium	J2ME MIDP	Yes/Yes
Opera Mobile	Presto/Proprietary	Yes	Planned	Several media supported	Yes, Opera widgets	No/Yes Medium	Windows, Linux	No/Yes
Openwave full browser	Mercury	Yes	Yes	Yes, plug-in framework	Yes, but limited	No /No	Windows, Linux, BREW	No/Yes
Netfront	Proprietary	Yes	TAP	Yes, multiple formats	Yes, at client	No/No	Windows, Linux, Real-time Executable (REX)/ BREW	Yes/Yes
Google Android	WebKit	Yes	Yes, driver-based	Plug-ins, Android and Linux-based	Yes, at client	No/No	Linux/Android	May not/Yes
Safari (iPhone)	WebKit	Yes	Multi	RTSP, No Windows formats	Yes, JavaScript	No/No	Mac OS	GPRS/WiFi Yes/Yes

```

/** Android WebView Example */
import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebView;

public class WebView extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        WebView wb = new WebView(this);
        wb.loadUrl("http://www.motorola.com/ui/index.html");
        super.onCreate(savedInstanceState);
        setContentView(wb);
    }
}

```

Figure 1. Instantiating a WebView object on Android. The Document Object Model (DOM) can only be modified as part of the JavaScript embedded in the HTML page. WebViews let this model change and developers can make programmatic DOM changes without creating additional JavaScript support, hiding logic from the HTML content that could be publicly available on the Internet.

a WAP-based browser. Skyfire requires server-based rendering and has become widely popular on Windows mobile devices. Because it uses Gecko as a

server-transcoding solution, most of Gecko's browser features will also be seen on Skyfire's J2ME and Windows Mobile clients.

WEBVIEWS AND THE FUTURE OF THE MOBILE WEB

As the war intensifies at the platform level, both Apple's iPhone and Google Android expand their Web engine framework so other applications can use them, restricting them as features exclusive to the mobile browser. In other words, these platforms let developers include Web-based interfaces as part of their applications. This is achieved by expanding the "view" concept and redefining it as a "WebView." A WebView is an instrument available to developers which a user can instantiate from any application on the mobile phone, hence any HTML content can become part of any application.

WebViews let mobile developers create any user interface design using standard Web-based technology. This HTML-content could be retrieved from a remote server, or from a local file on the device.

This versatility of the Web engine can be extended to include managing resources and manipulating the Docu-

ment Object Model (DOM) retrieved from an HTML site. Figure 1 depicts an Android application instantiating a WebView element. In this figure, a WebView object is created and initialized with a predefined Uniform Resource Locator (URL), which in this example is stored on the Internet but could be located in a file. On execution of this code, the same HTML content that a browser can view the user interface for the mobile application. Hence, developers are then capable of defining user interfaces using HTML and include them as part of their mobile software, reducing cost and increasing compatibility with the Web.

Even though WebViews aren't a new concept, they haven't been widely used on the desktop. Some instant messaging clients reuse an HTML rendering engine view that programatically retrieves HTML and Web content as messages and widgets. Unfortunately, embedding this content is cumbersome and often hard to manage, considering the lack of Ajax and CSS browser support. As the Web becomes more interactive and Web applications can be executed as part of the mobile browser and as part of a stand-alone native application, this approach becomes more feasible to implement and maintain.

As expected, a server-transcoding approach is hindered heavily by native rendering support that maps to the direct-delivery paradigm. Server transcoding might not be suitable for Web application development, especially as the adoption of WebViews becomes more popular. In addition, server-transcoding might be relegated to mobile browsing only. Thus, as Web applications become more pervasive and popular, this requirement will leave server-transcoding approaches as primarily suitable for low-tier devices and, as a result, with less support than expected. Additionally, Motorola provides a similar approach to deliver mobile Web content and Web application development on mobile handsets by using the WebUI framework (<http://developer.motorola.com/technologies/webui>).

This framework provides tools to create mobile Web content that can be rendered in some of the most popular Linux-based phones made by Motorola.

Mobile Web is the next step up for wireless, and carriers are using various mobile browser approaches to deliver mobile content. However, the notion that the mobile Web is about mobile information portals in which text-based support and images are sufficient might no longer prove valid. As Web applications become more pervasive, the use of more compelling and expandable solutions is more important than ever. Considerations such as Web 2.0 standardization and universal support of interactive HTML content like that now experienced on the desktop computer drives innovation, platforms, and hence mobile browsers. Additionally, efforts to integrate mobile Web rendering engines as part of the framework, together with the proliferation of

Web applications, might push WebKit based architectures, including iPhone, Motorola WebUI, and Google Android, to lead the way. ■

REFERENCE

1. O'Reilly, T., "What is Web 2.0. Design Patterns and Business Models for the Next Generation of Software," Sep. 2005; www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html.

Edwin A. Hernandez is

a principal staff software engineer with Motorola Mobile Devices in Plantation, Fla. He received his PhD in computer engineering from the University of Florida. His research interests include mobile middleware, network emulation, wireless networks, and virtual machines. Contact him at edwin.hernandez@yahoo.com.



COMPUTING THEN

Learn about computing history and the people who shaped it.

<http://computingnow.computer.org/ct>