Worklight® - Extend Your Business

White Paper

HTML5 as the Core Technology of the Mobile Enterprise

Contents

Intro 4

Strategic Considerations 4

Commitment from Mobile Vendors 4

Active Standardization Efforts 5

Growing Ecosystem and Tools 5

Delivery Options 6

Benefits of Using HTML5 6

Cross-platform Compatibility 6

Form-factor Adjustments 7

Cross-device Compatibility 7

Development Tools 7

Graphic Toolkits 8

Rapid Application Development Tools 8

Debuggers and Emulators 9

Distribution of HTML5 Applications as Mobile Web Apps 10

Delivering HTML5 Applications as Hybrid Apps 10

Core HTML5 Capabilities 11

UI Elements 11

CSS3 12

Non-UI Features 12

Challenges 13

Device-specific HTML5 Implementation 13

Proprietary Android UI Layers 14

Browser Memory Management 14

The Future of HTML5 15

Conclusion 16

About Worklight 17

Intro

The dispute between HTML-based and native mobile development sometimes seems like a theological conflict rather than a practical one. As mobility continues to rapidly expand into the daily operations of businesses around the world, identifying the right technology becomes a major challenge for the IT department of the organization. This whitepaper is aimed at helping decision-makers focus on technical facts to better assess the pros and cons of adopting HTML5 as their core mobile technology.

This paper highlights the relevant aspects of developing mobile applications using HTML5, the strategic considerations for its adoption and the benefits of leveraging HTML5 as part of the organization's mobile strategy. This paper will also provide an overview of HTML5's capabilities and the challenges enterprises can expect to encounter.

Strategic Considerations

Commitment from Mobile Vendors

The market for mobile operating systems, be it smartphones or tablets, is dominated by a handful of players such as Apple, Google, Microsoft and RIM. These vendors are actively increasing the number of HTML5 features that their mobile browsers support, allowing website developers to deliver a richer user experience to mobile visitors.

But beyond their promotion of HTML5 as a web browsing standard, mobile OS vendors are turning HTML5 into a viable development option for downloadable applications that seamlessly integrate and interact with the mobile device and its native features.

Apple, Google and Microsoft allow developers to embed a browser component within their applications and to freely use it to implement the application user interface and logic. RIM and HP take this approach even further and base their entire app framework on web and JavaScript development that allows OS APIs to directly access device features and capabilities using JavaScript.

Active Standardization Efforts

HTML5 is actually an umbrella term for nearly a hundred interrelated features, including textual and graphical presentation, data semantics and manipulation, messaging, storage and user interaction. The web community is actively working on standardizing these features, creating the basis for cross-platform compatibility of web code on both desktop and mobile browsers. Many mobile OS vendors are already implementing many of these features even before the W3C formally recommends them, either strictly according to the proposal or as proprietary extensions.

Despite the wide difference among the mobile operating systems from these leading vendors, all see HTML5 as a strategic technology for developing applications and delivering content on their platforms.

Growing Ecosystem and Tools

Another strategic consideration for developing mobile apps using HTML5 is the growing ecosystem of tools and resources. These tools and resources support the development effort, shorten the time to market and increase the richness and overall quality of applications.

Strategically, this means several things:

- Developers can and will be able to rely on the increasingly growing knowledge-base that is available online for learning new topics and troubleshooting development problems.
- The variety of development tools will continue to expand, enabling developers to adapt a best-of-breed approach to address their needs.
- The competitive development tools market will continue to generate better tools that are more sophisticated, cheaper and sometimes even free.

Delivery Options

The final strategic consideration with regards to HTML5 app development is the delivery of the application to the end users. There are two options for delivering HTML5 apps to mobile devices. The first is as mobile web apps and the second is as downloadable hybrid apps. Two options that differ from each other in various aspects including the download and installation experience, remote application control, application version management and even monetization strategies. The implications of each delivery option will be discussed in greater detail later in the document.

At first, the choice between these two delivery options might seem cardinal to the architecture of the application, but in fact, converting a pure HTML5 mobile web app into a hybrid app, scaling down a hybrid app into a pure web app, or even simultaneously providing both options to extend the reach of the application could be a very easy task.

Now that we've discussed the strategic factors that organizations should consider, let's evaluate the different benefits of using HTML5 in an enterprise context.

Benefits of Using HTML5

Cross-platform Compatibility

iOS, Android, BlackBerry and Windows Phone are significantly different from each other in terms of their native application development technology. Native iOS apps are developed in Objective-C, Android apps are developed in Java, BlackBerry apps are developed in Java or in JavaScript and Windows Phone apps are developed in C#. These unique approaches to mobile app development rule out any cross-platform compatibility of the application, making each source code an independent effort that cannot be reused.

But with the evident support for HTML by all of these platforms, developers can leverage HTML5 to create cross-platform applications that run on all mobile devices. Economically speaking, companies will not need to develop native expertise for each mobile operating system, but rather reuse their existing in-house web development knowledge.

Keep in mind – one of the major benefits of HTML apps versus other cross-platform solutions is that the HTML and JavaScript code is executed by the device's native browser, without the hindering involvement of third-party interpreters. This makes the code faster, more reliable and grants developers immediate access to native browser capabilities without relying on a third-party support.

The conclusion is clear - using HTML5 as the core technology of your mobile apps reduces the amount of effort and resources needed to support the range of mobile operating systems in the market today.

Form-factor Adjustments

But cross-platform compatibility presents only one challenge. Nearly all mobile operating systems support multiple devices that consist of different sizes, resolutions, form factors, input mechanisms and screen densities. As a result, mobile developers are burdened with the responsibility of ensuring their apps are well displayed on a variety of devices from the same operating system family.

To overcome this challenge, HTML5 developers are able to leverage CSS3 features such as media queries, text overflow, word wrap and relative element sizing to automatically adjust the application page layout to the device on which it is displayed.

Cross-device Compatibility

This challenge is intensified even further by the fragmented nature of the Android market. Different device manufacturers using Android as their OS of choice add their proprietary user interface layer on top of the standard operating system. As a result, developers are forced to ensure that their app is well displayed on a wider variety of devices that utilize UIs such as HTC Sense, Samsung Touchwiz, and Sony Ericsson Timescape. While these UI layers generally affect the appearance and behavior of the native user interface, HTML UI rendered by the mobile browser is far less affected by them, effectively reducing the burden of the application developer.

Development Tools

As mentioned earlier, the ecosystem around HTML5 is growing rapidly and includes a variety of code libraries and development tools, among which one can find JavaScript libraries that provide a native-like user interface. Tools like jQuery Mobile, Dojox.mobile and Sencha Touch are a few examples of tools

that allow developers to use HTML and/or JavaScript to create a rich user interface that looks and behaves like that of a native app.

Using these and similar libraries, the UI development experience resembles that of a native application, allowing developers to focus mainly on their choice of displayed controls and interactions between them. Considerations of colors, shapes and themes are addressed automatically by the libraries. The result is a user interface that looks like a native screen, but implemented using pure HTML or JavaScript code that can be quickly developed, naturally compatible with multiple devices and easily maintained.

Graphic Toolkits

Other types of JavaScript libraries aim at simplifying the development of graphics displayed in an HTML page using the new HTML5 canvas element. The canvas element allows for dynamic, scriptable rendering of two-dimensional shapes and a bitmap images. It consists of a drawable region that is defined in HTML code and includes height and width attributes. JavaScript code may access this area through a full set of drawing functions.

Toolkits like cakejs, Highcharts, Zingcharts and Flot all provide a JavaScript API for creating compelling and interactive pictures, diagrams, charts and other graphics, often using the HTML5 canvas element.

Rapid Application Development Tools

Another promising trend is the emergence of rapid app development tools that use web technologies such as HTML, JavaScript and CSS. Such tools provide a WYSIWYG approach for building web apps, allowing developers to drag and drop components on a blank canvas and define their interaction.

One such example is Sencha's Animator tool. This desktop application helps developers create animations and other dynamic effects that are based on CSS3 and using a graphical user interface. Another example is Maqetta. This open source initiative from the Dojo Foundation is a WYSIWYG editor that runs inside the browser and enables the creation of HTML5 and JavaScript apps.

The approach that these two examples follow can be very beneficial for a wide range of fairly simple apps, but if your app is more complex and the developer requires a more hands-on control of the codebase, it may not always suffice.

Nevertheless, the evident support for HTML5 by desktop and mobile browsers will continue to push the demand for rapid app development tools even higher. Sencha Animator and Maqetta are forerunners in this field who probably provide an accurate outlook of what developers can expect in the near future.

Debuggers and Emulators

Debuggers and emulators play a critical role in mobile development, particularly given its fragmented nature. This is also true for HTML5 applications.

One good thing about the iOS, Android, BlackBerry and WebOS mobile browsers is that they all incorporate the same engine, WebKit. This common component greatly increases the chances that code developed for one device will work on all others. Furthermore, it greatly simplifies multi-platform testing. Even better, leading desktop browsers such as Chrome and Safari are also equipped with WebKit. Since these browsers provide great developer tools and plug-ins as well as extensions for debugging, tracing and testing web applications, this means two important things:

- The growing ecosystem around desktop web development directly behooves and promotes mobile web development.
- Much of the debugging phase can be performed on the developer's desktop which is much more convenient than on-device debugging.

In addition to using Chrome, Safari and plug-ins for desktop web app development, there are also tools that emulate mobile devices directly within the browser. An example of such a tool is the Ripple Chrome plug-in. This plug-in allows the developer to select the desired framework, e.g. PhoneGap or mobile web, device, such as iPad, Nexus One or BlackBerry Torch, and orientation, and then emulates the web page according to the user's selection.

And finally, there are always native emulators available in the mobile manufacturers' SDKs which can be used to emulate various types of devices and test the behavior of web applications on them.

So far we mentioned the benefits of HTML5 as a development approach for mobile apps. Organizations planning their mobile strategy must also consider aspects such as post-deployment control and management (applications “in the wild”) – areas where HTML5 is also beneficial.

Distribution of HTML5 Applications as Mobile Web Apps

Delivering HTML5 apps as mobile web applications enables direct control over the app and simplifies its version management. Organizations can easily upload new functionality and bug fixes onto the application server and ensure that these changes are immediately available for the mobile user. Furthermore, organizations can significantly shorten time to market as application updates are not subject to app store approval processes (a lucrative option for developers who do not wish to share their app revenues with the different distribution stores).

A popular misconception around mobile web apps is that they can only be used online and that the user must explicitly browse to the app each time he or she wants to use it. This perception is simply not true in modern mobile operating systems. HTML5 mobile web apps can be entirely cached on the device and available for offline access. Caching is performed in the background, so the application can be used during the caching process.

To promote the installation of HTML5 mobile web apps, developers can provide Quick Response (QR) codes for device scanning, circumventing the need to manually type cumbersome URLs. Once users access the app, they can easily “pin” it to their device's home screen for easy future access, just like any downloadable app, allowing organization to maintain their branded presence on the user’s device.

Another benefit of delivering code as mobile web apps is that even though they are written extensively using Ajax, mobile web apps can be crawlable by search engines using Hijax or Pretty URLs techniques.

Delivering HTML5 Applications as Hybrid Apps

On the other hand, some organizations might consider delivering the code as a mobile web app to be a major drawback given its lack of visibility in major app stores and markets. If app store presence is a critical requirement, the organization can easily turn the web app into a hybrid app by packaging the HTML5 code within a thin native container and deliver it via stores.

In such cases, the download and installation experience is the same as any other native app. From a version management perspective, updates to the HTML5 resources of the hybrid app can still be pushed directly to the user without the app stores’ time-consuming process of submission and approval.

Another obvious benefit of encapsulating HTML5 code in a hybrid app is the access to native device functionalities that are otherwise inaccessible to HTML5 running within a mobile browser. This approach allows developers to write the majority of the app in cross-platform HTML5 code, while implementing native code to access unique functionality and animation requirements such as augmented reality and other device features that are not yet exposed to web technologies such as the device's calendar, camera and accelerometer.

Core HTML5 Capabilities

UI Elements

Having described the benefits of HTML5, let us now review some of its main core capabilities.

Mobile operating systems already support many aspects of HTML5's UI features. Some developers design their applications to blend with the native look and feel of the underlying operating system. When using HTML form components, developers do not need to worry about manually adjusting their style to the mobile OS. This is done automatically by the browser which is a native application in itself.

The HTML4 form controls, such as dropdown menus and check boxes, are fully supported and are natively rendered by the browser. Some implementations of the HTML5 form controls, such as date pickers on BlackBerry 6, or the e-mail and telephone HTML5 input elements on iOS are already supported.

We previously mentioned Canvas as a major HTML5 feature that was embraced by mobile operating systems. The Android and iOS WebKit browsers support not only the basic canvas, but also its Text API for and the unofficial WebKit extension allowing the use of canvas drawing as CSS background.

Many mobile operating systems are also starting to embrace HTML5's vector graphics standards. Basic SVG is supported by iOS 4.2, Android 3.0 and Internet Explorer 9, which would be available with the upcoming release of Windows Phone 7.5. Using SVG as CSS backgrounds to define fonts and to animate elements is also supported by Android 3.0 and soon by other browsers.

HTML5 allows playing sounds and running videos without the use of external plug-ins. Both these elements are supported by the iOS, Android and hopefully by the future Windows Phone browser.

CSS3

CSS3 is significantly more capable than CSS2 making images redundant, the applications smaller in size and faster to render and the code easier to maintain. Mobile browsers support many of the CSS3 recommendations and working drafts, including transparency, 2d and 3d animations and transformations, word wrapping and overflow, opacity, color models, media queries, rounded corners, gradients and shadows. Many developers find CSS3 to be instrumental for the enhancement of the overall user experience as it allows developers to match the design of native apps for the different devices and operating systems.

Non-UI Features

But HTML5 is more than just a UI tool. A variety of non-UI features that have been introduced by HTML5 are particularly useful for mobile apps and are already supported by mobile browsers. These core capabilities make HTML5 a powerful technology that delivers rich apps across devices and operating systems, using a single development methodology that makes the entire process highly efficient.

Geo-location

The mobile browser provides access to the user's geo-location allowing web developers to write location-aware apps using JavaScript. Combined with web maps, this is a particularly strong feature.

Offline Availability

We discussed the local storing of application resources on the device for offline access, but mobile browsers also support the HTML5 web storage feature which allows storing key-value pairs of data on the device and accessing them offline.

Web Sockets

Web Sockets, which provide a bidirectional, full-duplex communications channel between the mobile client and the server, are already supported by iOS 4.2.

Challenges

Device-specific HTML5 Implementation

Despite the aforementioned benefits of HTML5, the picture is not complete without mentioning the main challenges that are associated with this technology. Because many HTML5 capabilities are not yet formal recommendations, and some of which are still at quite an early stage, some features are implemented by only some of the mobile browsers. With time, the scope of features will expand.

Certain CSS Features

Take CSS3 features like multiple column layout, transformations, and rounded corners for example. Neither of these are formal recommendations; they are either working drafts or candidates for recommendation. Mobile browsers support them through their proprietary –webkit- or –ms- prefixes. This is not a major issue, but one that still requires browser-specific code.

Web Sockets

There are other HTML5 features that are not supported by all mobile operating systems. Web sockets, for example, are supported only on iOS and the WebM/VP8 video format is supported only on Android devices. Note how the fact that both the iOS and Android browsers are based on the WebKit engine does not guarantee they are fully compatible in all of their features.

position:fixed

Another interesting example is the CSS *position:fixed* attribute, which indicates to the browser the need to fix a certain element on the screen, making the other parts of the screen scrollable. This attribute is only supported on Android 2.2 devices. On previous versions of Android and on iOS, if developers wish to render only some screen elements scrollable, they are forced to use alternative mechanisms such as the iScroll JavaScript library.

This example highlights an interesting aspect we previously mentioned as it demonstrates how missing features are compensated for by the HTML5 ecosystem, relieving developers from technological dependency to achieve their goals.

Cache Limitations

The last example that demonstrates the inconsistency of HTML5 feature implementation is the size of the cache. While most mobile browsers implement a 5MB limit per origin, some browsers allow the users to extend it while others don't.

Proprietary Android UI Layers

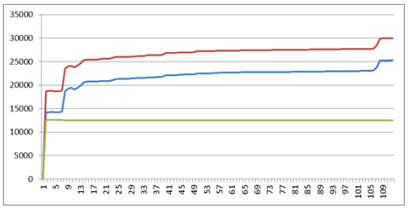
Another type of challenge is specific to Android devices and stems from the additional UI layers that are installed by some device vendors. HTC Sense, for example, is a UI layer that is available on popular devices like the Desire. The problem with HTC Sense is that it changes the layout of some UI controls that are rendered by the browser. The result is that edit boxes styled one way may be automatically rendered in a uniform style which may not necessarily fit the original application design.

Browser Memory Management

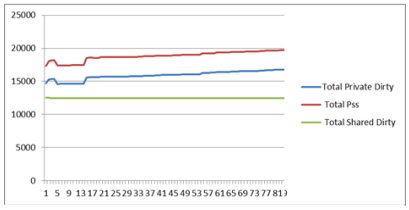
Browser memory management is probably one of the trickiest topics in mobile application development, yet most developers blissfully ignore it. However, if the app makes heavy use of animations, frequently loads and unloads HTML fragments to the DOM, or performs many Ajax requests that result with complex responses, developers may want to monitor the application's memory consumption over time and ensure the availability of JavaScript objects whenever possible.

Finally, perhaps the most basic challenge of all – the device CPU. New and strong devices from the iOS and Android families can run complex HTML-based UIs very smoothly. Some devices also make use of hardware acceleration for animations, drop shadows and other fancy HTML5 and CSS3 features.

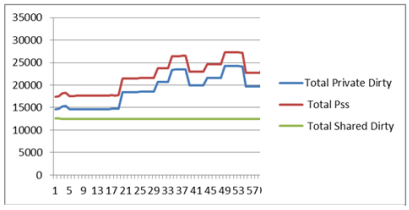
However, applications running on older, or low-end devices (a market that is growing quickly), might experience some degree of performance degradation. This degradation will mainly be noticed when the app renders complex graphics and performs heavy animations.



Heavy and repeated jQuery animation



Frequent loading &releasing DOM fragments



Frequent complex Ajax responses

The Future of HTML5

Desktop browsers are leading the adoption of HTML5 features while mobile browsers, or more accurately WebKit, is following suit. This is not surprising, considering the CPU and memory constraints imposed on mobile browsers as well as the device's battery life. However, with time, we can expect mobile browsers to support some of the most exciting HTML5 features including:

- WebGL, which provides the ability to draw 3D Canvas graphics.
- Indexed DB, which introduces complex on-device storage and will replace the SQLite specification
- Web Workers, which allows developers to run JavaScript code in the background without affecting the responsiveness of the user interface.
- Push notifications to a mobile application, even when the application is not active, or not resident in the device's memory. This capability is provided for native applications, but is not supported for mobile web apps. It is supported, though, since March this year, by the Chrome desktop browser.

Yet certain mobile functionalities remain unsupported by HTML5. For example, access to device features such as the camera or accelerometer is not possible via plain JavaScript on current versions of iOS and Android. This does not stop RIM and HP from providing proprietary JavaScript APIs in their BlackBerry and WebOS devices to access these device capabilities as well as others. Google also presented a preview of this capability designed for future Android versions.

Conclusion

Developers asking to build a game which must exhibit extreme responsiveness and display rich and interactive graphics should probably opt for native application development. However, for most enterprise apps, whether targeting consumers, business partners, or employees, these limitations should not drive decision-makers away from choosing HTML5 as their core mobile technology.

Vendor commitment, recent technological advancements and its evident adoption in the market have positioned HTML5 as a promising future standard for mobile development. But despite its promise, certain technological gaps between HTML5 and native code development remain.

But for those organizations that require access to certain features not currently accessible using web technologies such as the device's camera, accelerometer, calendar, push notifications, augmented reality, or simply rendering complex graphics, a different solution remains – the hybrid application.

Hybrid applications include a thin native layer that exposes the necessary native features of the device as a large corpus of HTML5 and JavaScript code for the application logic and presentation. Taking this approach, developers can still enjoy the benefits of choosing HTML5 as their core mobile technology, without compromising on application functionality, richness and, most importantly, the user experience. Furthermore, as HTML5 continues to evolve and cover more mobile features and functionality, early adopters of this approach will benefit from a seamless and gradual transition as opposed to an extreme overhaul of their entire mobile infrastructure.

About Worklight

Worklight, an IBM company, provides an open, complete and advanced mobile application platform and tools software for smartphones and tablets. Our products enable organizations of all sizes to efficiently develop and deliver HTML5, hybrid and native applications with a powerful and flexible mobile IDE, next-generation mobile middleware, end-to-end security and integrated management and analytics.

Worklight technology enables rich, cross-platform apps without requiring code translation, proprietary interpreters or unpopular programming languages. Worklight dramatically reduces time to market, cost and complexity while enabling better customer and employee user experiences across more devices.

For further information please visit our website – www.worklight.com

Download a free evaluation version of our platform – www.worklight.com/download