Understanding Hybrid Architecture

Version: 2013.11.21

Contents

Contents
CIO/Project Manager
    Why is this important to me?
Front-end Developer
    Why is this important to me?
Hybrid Architecture Concepts & Fundamentals
Android Platform
    Fragmentation
    Here Comes Chromium
    Unfortunate side effects
iOS Platform
Development Options when Encountering Performance Issues or Limitations with Hybrid
    Implement your entire app using Native SDK
    Implement screens with advanced capabilities or degraded performance on older versions using native, and other screens using web technologies (a mixed hybrid application)
    Implement your entire app with web technologies, turning off advanced visual effects and possibly using alternative u/x design that avoids slow performance (for example scrolling of large lists can be slow)

Copyright © 2013 International Business Machines. All rights reserved.

CIO/Project Manager

Why is this important to me?

Understanding how Hybrid application architectures differ from other environments such as web or pure native applications is crucial to understanding how this type of architecture can be used to reduce overall project costs over time. Not understanding the limitations of hybrid environments can also unexpectedly introduce user experience or performance problems late in a development cycle.

If you don't have a strategy in your project or across your organization for how to deal with known hybrid issues on different platforms or new issues that might come up, chances are you will spend a lot of extra money in recovery that could have been saved.

If you're a manager, please take the time to understand the following Concepts & Fundamentals section and as a follow-on activity, read the related Updating Applications To Support New Operating System Versions guidelines and make sure to adjust your projects' development process accordingly.

The wide variations in the browser environments described across platforms and versions in the next section is *situation normal* for hybrid and web application developers; however, it should be clear that in this world, different rigorous development processes, different development techniques and different planning will be required to ensure project success that you may not be aware of.

The mobile development model, whether native, hybrid or web architectures is not:

- Write once, run everywhere
- Write once, run forever without having to make changes
- Design once, pixel perfect, run on any screen size.

One of the biggest mistakes is expecting that it is.

Mobile development with hybrid and web architecture is:

- Writing to standards (not proprietary api's)
- Reusing Web skills
- Reusing Tools
- Reusing HTML/CSS/JavaScript assets over time and across projects and platforms
- Making necessary adjustments to applications as runtime environments change using well established design techniques, patterns and best practices to keep changes minimal.
- Design mobile first, using responsive design techniques and environment awareness--not pixel perfect, adaptable to different screen sizes and orientations.

Copyright © 2013 International Business Machines. All rights reserved.

Adapting to rapidly evolving mobile platform releases requires a strategic plan for dealing with these platform changes over time, including updating and migrating applications with support for new emerging platforms on a near-continuous basis. And it requires the use of development techniques that allow your application design and architecture to accommodate changes in the underlying platform.

Copyright © 2013 International Business Machines. All rights reserved.

Front-end Developer

Why is this important to me?

There are always different features available that need to be checked for and code needs to be written in a flexible manner to be resilient to these differences. Various techniques are now well-established for helping developers deal with these kinds of differences in JavaScript and HTML, such as feature detection, feature testing, mobile first responsive design and open source tools and performance optimization techniques that you should be aware of and using in your applications. It is important that you have a good understanding of the Front-end Development technology-specific Best Practices and Checklist items that have been captured on this site.

Most front-end developers view webkit as a black box component, but it may be helpful to understand how Webkit is designed to understand what are the causes of variation. For more technical details on the how webkit browser engines is designed and how it varies across platforms, see Paul Irish, WebKit for Developers.

As you develop your own tips and tricks, consider sharing the techniques with us via a blog post or comment on this site, so that others can benefit from your lessons learned.

Copyright © 2013 International Business Machines. All rights reserved.

Hybrid Architecture Concepts & Fundamentals

Hybrid Applications consist of native code that creates an embedded web browser used to render parts of the application user interface which are written in standard HTML, JavaScript and CSS formats.

The actual browser engine used within the embedded web browser and it's configuration options are provided by the device operating system vendor. The browser engine's implementation changes between versions of the operating system, usually the same browser engine technology is kept and just a newer version of the browser engine is used when an O/S is updated, but sometimes completely different browser engines are used in newer versions of the O/S. Across different operating systems, completely different browser engine implementations are used (eg. WinPhone8 - Internet Explorer/Trident, Apple iOS - Safari/ Webkit), even though they may be based off of the same underlying browser engine codebase.

Even if an operating system provides a specific browser engine, parts of the browser engine can be replaced by carriers (that have forked the primary O/S for their devices or markets). A good example of this is Webkit, which has replaceable subsystems that are implemented differently on different operating systems (eg. iOS, Android, Blackberry).

It's also important to realize that the embedded web browser engine used inside the webview native code of the hybrid app often has different capabilities enabled from the default standalone browser application provided by the mobile operating system vendor. There is no guarantee that the browser engine used by the default browser will have exactly the same features and runtime characteristics as the browser used by the embedded webview API provided by the same O/S vendor and version. As an example, see the next section about Android's browser engines changing in different versions of Android O/S.

The above facts are not obvious, but can have large ramifications on application functionality and performance.

Copyright © 2013 International Business Machines. All rights reserved.

Android Platform

Currently, there are the following versions of Android in the marketplace:

Version	WebView Browser Engine	Default Browser Application Browser Engine
2.x + carrier-specific forks	Android Default (webkit)	Android Default (webkit) + carrier-specific replacements
3.x + carrier-specific forks	Android Default (webkit)	Android Default (webkit) + carrier-specific replacements
4.1-4.3 + carrier-specific forks	Android Default (webkit)	Chromium (Blink) + carrier-specific replacements
4.4 (Kit Kat)	*Chromium (Blink)*	*Chomium (Blink)*

For user-installed web browser applications on Android, there are multiple alternative browsers available through Google Play.

Fragmentation

Note that there are different forks of the Android codebase versions for different worldwide carriers. Some of these carriers tweak or modify the webkit subsystems, others replace the browser with their own, for example the UCBrowser is common in India and Chinese carriers.

Here Comes Chromium

As you can see from the table above, in Android 4.0, the platform's default browser application changed from an Android webkit fork to use the new Chromium engine based on Blink. Chromium (the same browser engine that Google Chrome's desktop browsers are based on) supports more HTML5 features and has better support of many of the same features that were in the Android Default browser engine.

Unfortunate side effects

Unfortunately, it appears that around the time Android began switching Chromium in for the default web browser, the Android default browser engine used by the WebView API's provided by Android was left in the dust. This means that currently on Android, web apps running in the Chrome browser have a different (more advanced) set of features and much better runtime performance than hybrid apps which are using the less capable engine. Fancy effects such as 3d transitions are possible only on the default 4.x browser application.

Because of the additional carrier forks (many carriers fork once and never provide OS updates to customers), it is impossible to back port and update the default web view and browsers on older versions of Android. It is technically possible to embed Chromium browser engine inside of a hybrid app, but experimental open source projects have shown that this only works on

Copyright © 2013 International Business Machines. All rights reserved.

Android 4 (Chromium would need to be backported to previous Android API's to be embeddable on those versions), and embedding Chromium within an app increases the application footprint by at least 16MB.

Because of these differences, Android platforms require more effort to test on a wider variety of devices than iOS. You may find that on a specific device model performance goals or user experience is not achievable. In these cases, see the Development Options when Encountering Performance Issues or Limitations on Hybrid and Web Architectures section for common solutions.

This wide variation is situation normal for native, web and hybrid architectures on Android and you will need to plan for changes over time, even for apps already in production, and is very similar to the browser history and problems/solutions needed for Internet Explorer on desktop platforms. See the Upgrading Applications to Support New Operating System Versions document for further guidance on how to plan accordingly.

It is great news for the Android 4.4 platform that the browser engine will be Chromium/"Blink"-based for *both* the webview used by hybrid applications and web apps running on the default browser application. Performance improvements and advanced HTML5 capabilities are now possible for hybrid apps running on Android 4.4. However, it's impossible to have the Chromium webview backported onto older Android versions, so customers running 4.0-4.3 will need to upgrade their OS, a non-trivial procedure in order to have decent hybrid capabilities.

Also, applications which were using various performance workarounds on Android 4.0-4.3 need to now check the API level and either apply the previous workarounds for 4.0-4.3 or remove them for the new API levels, and the app must be built to target Android 4.4, otherwise the app will resort to a "quirks" mode that will degrade behavior to be similar to 4.0, even though the app would be running on 4.4.

Copyright © 2013 International Business Machines. All rights reserved.

iOS Platform

Currently, on all versions of iOS in the marketplace, the browser engine used in embedded web views is the same as the browser engine used within the default browser application, Safari/Webkit.

The main differences between the browser engines are in HTML5 capabilities, bug fixes, and normal browser improvements. However, even though web applications running in hybrid web views, and default Safari browser are running on the same codebase, the options enabled in the browser can vary. Also, between versions of iOS, webkit browser features that are controlled by the O/S may change, resulting in variations.

Here are some examples of variation that developers have experienced that application developers may need to make changes to their applications when new versions appear:

- JavaScript JIT optimization may not be enabled when launching web apps that have been bookmarked to the home screen, resulting in lower performance.
- The browser dimensions may no longer be the same as the screen dimensions, affecting viewport dimensions.
  Examples:
    - iOS7's changing the previously reserved status bar area to be an overlay results in a 20px change in viewport height.
    - When a soft keyboard is shown on iOS7, the default behavior of the browser viewport has changed. Previously, the viewport's height when keyboard was displayed was the device-height, in iOS7 the viewport's height becomes the remaining height in the visible area above the soft keyboard.

  The first change due to status bar can be solved easily with a media query.
  The change because of the soft keyboard can be solved with a <meta> viewport height change. Unfortunately, running the new html on an older version of iOS will not give the application proper behavior, because this viewport height setting was broken in previous releases.

The goal of this section is not to list all of these known differences and solutions here, but rather to bring awareness as to the kinds of things that can affect an application that was working on a previous version of the same iOS.

You may find that on a specific device model or older O/S version that is still supported in your market, performance goals or user experience is not achievable. In these cases, see the Development Options when Encountering Performance Issues or Limitations on Hybrid and Web Architectures section for common solutions. Again, this is normal for native, web and hybrid architectures. See the Upgrading Applications to Support New Operating System Versions document for further guidance on how to plan accordingly.

Copyright © 2013 International Business Machines. All rights reserved.

Development Options when Encountering Performance Issues or Limitations with Hybrid

What all of this means is that if you are targeting Android you have the following options if you find performance problems due to the default Android browser based on the Android devices used in your market:

1. Implement your entire app using Native SDK

This is a perfectly acceptable technical approach, but has a higher business cost (both initial and ongoing) due to maintenance, training/skills over time.

You'll need to hire a larger team of native developers on each project with skills specific to each native vendor O/S that you need to support the applications on. This option also results in having completely separate codebases across vendor operating system platforms, with no sharing of common code assets between platforms.

2. Implement screens with advanced capabilities or degraded performance on older versions using native, and other screens using web technologies (a mixed hybrid application)

An example of this approach is that some screens such as login, top-level navigation, and other frequently visited screens are done in native, while less-frequently visited screens are implemented with web technologies. Over time, the native views can be dropped and replaced with web-based views (that may run just fine in the iOS environment branch of your project) so that you remove native parts as the browser capabilities improve.

You'll need to hire a small team of native developers with skills for each native platform supported on each project that uses this option. Care needs to be taken to align the visual design and behavior of the native implemented views with the web views so that the integration is as seamless as possible from an end-user perspective. Costs concerns for the native portions of the app are same as option 1.

3. Implement your entire app with web technologies, turning off advanced visual effects and possibly using alternative u/x design that avoids slow performance (for example scrolling of large lists can be slow)

Copyright © 2013 International Business Machines. All rights reserved.

As an example, one of the known performance problems on certain device models such as Samsung Galaxy Tab 2 have poor native scrolling performance in the embedded web browser. If you can change the application design such that the user does not touch scroll through a list, but instead is designed to provide paging buttons, and this user experience tests acceptable with your users, you have a workaround that does not require native.

Copyright © 2013 International Business Machines. All rights reserved.