Development of Hybrid Applications with HTML

by Nripin Babu & Arun Bhat

Synopsis

Organizations that are planning mobility initiatives need to address the key needs of manageability, multiplatform support and providing an adaptable and intuitive user engagement while reducing development time and cost. This paper presents a case for the hybrid application approach in addressing these needs.

The main focus of this paper is enumerating key technical considerations that can ease and speed the development process of Hybrid Mobile Applications. These include best practices and tips for the design, development and testing of Hybrid applications.

The goal is to give readers an initial context and practical information for understanding the unique aspects of hybrid mobile application development including architecture, and key design, development & test considerations like usability, performance and security considerations.
# TABLE OF CONTENTS

1. A Case for Hybrid Applications ........................................................................................................ 3  
   1.1 Mobility – Strategic Needs ........................................................................................................ 3  
   1.2 Mobile Development Approaches .............................................................................................. 3  
   1.3 Introduction to Hybrid Applications ............................................................................................ 3  
   1.4 Benefits of Hybrid Applications ................................................................................................. 4  
   1.5 Value of Hybrid Applications ..................................................................................................... 4  

2. Developing a Hybrid Application ...................................................................................................... 6  
   2.1 Logical Architecture of a Typical Hybrid Application ................................................................. 6  
   2.2 Hybrid Application Design, Development & Test Considerations ............................................. 7  
   2.3 UX Design and Custom Usability Guidelines .............................................................................. 7  
   2.4 Application Performance ........................................................................................................... 8  
   2.5 Security Management .................................................................................................................. 8  
      2.5.1 Communication APIs ........................................................................................................... 8  
      2.5.2 Local Storage and Client-side databases .............................................................................. 9  
      2.5.3 Sandboxed Frames ............................................................................................................. 9  
   2.6 Offline Data Management ........................................................................................................... 9  
   2.7 UI/UX Testing ............................................................................................................................ 9  
   2.8 Security Testing .......................................................................................................................... 10  
   2.9 Performance Testing .................................................................................................................. 10  
   2.10 Test Automation ....................................................................................................................... 10  
   2.11 Development Challenges & Recommendations ....................................................................... 10  
   2.12 Hybrid App Platforms ............................................................................................................. 11  

3. A Practical Implementation of a Hybrid Application ........................................................................ 12
A Case for Hybrid Applications

Mobility – Strategic Needs

In order to understand the demand for a hybrid approach, it's useful to first review the strategic needs impacting manageability and cost that Organizations need to address before they invest in building mobile applications.

Manageable Applications: with the emergence of Bring Your Own Device (BYOD) policies organizations must look at reducing the dependency on device specific code by limiting the footprint on the device and look at maintaining a single manageable code base on the web. This approach will offer better control and security.

Multiplatform Support: the proliferation of multiple proprietary platforms, devices and technologies has resulted in a vendor specific platform dependency (ie) code written for one mobile platform cannot be used on another. This critical drawback drives up development costs and to address this Organizations should look at a single code base that can work across multiple platforms and devices

Multiplatform support becomes especially important in an enterprise scenario at a time when enterprises are embracing the BYOD concept. Hence the hybrid model is ideally suited for an enterprise with cross platform deployment needs.

Mobile Development Approaches

Broadly speaking there are three mobile development approaches—Native, Mobile Web, and Hybrid.

Introduction to Hybrid Applications

Hybrid apps make it possible to embed web technologies like HTML5, CSS and JavaScript code into the Apple or Android ecosystems thus combining the best elements of native and mobile web apps.

A hybrid app is essentially a web app, built using HTML5 and JavaScript that is then wrapped inside a thin native container that provides access to native platform features like camera, file system, contact list, accelerometer etc. PhoneGap is an example of a popular container for creating hybrid mobile apps.

| Native: In this approach, an application is developed using the development tools and Operating System language that the respective platform supports (e.g., Xcode and Objective-C with iOS, Eclipse and Java with Android). Native apps make best use of individual platform SDKs and device features and hence result in the best user experience. However, this approach drives up development costs if multiple platforms need to be supported. |
| Mobile Web: This approach is based on standard web technologies like HTML, CSS (for presentation) and JavaScript (for interactivity) and it is merely accessing web pages on mobile devices. It is the simplest to build and does not require much investment. However the drawback with this approach is limited user experience and more importantly security limitations and no support for leveraging powerful native device features like camera, GPS, Bluetooth etc. |
| Hybrid: This is designed to take advantage of both the Native and Mobile Web approaches. It benefits from the versatility of web technologies with the combination of powerful device features and SDK. It is well suited for a broad range of applications and can still provide a good user experience. |
To all but the most tech-savvy end users, hybrid apps and native apps are indistinguishable. In time the need for these wrappers should decline, as HTML5 adds more standard ways to access native capabilities.

**Benefits of Hybrid Applications**

The key benefits of hybrid applications are identified below:

- **Faster Time to Market**
  when compared to pure native app development time cycles

- **Lower Total Cost of Ownership**
  Easier upgrades, lower maintenance costs, development using familiar tools & technologies

- **Cross Platform Portability and Scalability**

- **Good UX Experience**
  with the modern JavaScript toolkits (gestures, swipes, taps & transitions)

- **Access to Native Features**
  like camera, GPS, bluetooth, contacts, accelerometer etc.

**Value of Hybrid Applications**

*What is the value of Hybrid Applications over Mobile Web or Native Applications? Why is it a talking point?*

The hybrid application benefits from the versatility & ease of development of web technologies with the combination of powerful device features and SDK of the native approach. Hence the main advantage of hybrid applications is providing cross platform compatibility, with an emphasis on accessibility to native features which results in a good user experience. The hybrid approach also allows the apps to gain access to the app stores, where the apps enjoy visibility and monetization support.
Key features supported include:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database storage</td>
<td>In mobile web apps data is stored as plain text which lends itself to security threats due to loss of data and if the storage location is fixed other apps can access it. With hybrid applications, data can be stored securely with encryption.</td>
</tr>
<tr>
<td>Network connections</td>
<td>HTML5 supports web sockets. The hybrid approach can handle full socket communications. Native mobile components can open a socket and can communicate with the server/other devices, just like in traditional socket communication. In HTML5, the web server can support the web sockets, though not all the browsers are supporting this feature at this time.</td>
</tr>
<tr>
<td>Push notification</td>
<td>Real-time push notifications are possible with the hybrid approach via the use of native components. HTML5 also supports notifications via server push technology with persistent connections.</td>
</tr>
<tr>
<td>Access to native features</td>
<td>The hybrid approach supports full access to native device features such as the camera, motion sensors, accelerometer, file system access, contact lists and sensors</td>
</tr>
<tr>
<td>Reusability</td>
<td>The backend applications cater to a range of mobile and desktop platforms which makes hybrid applications reusable across both mobile and desktop platforms</td>
</tr>
</tbody>
</table>

**What is the level of adoption of the Hybrid Approach across companies?**

**Gartner, Inc. predicts that more than 50 percent of mobile apps deployed by 2016 will be hybrid. The key reasons are:**

To address the need for mobile applications, enterprises are looking to leverage applications across multiple platforms. The advantages of the hybrid architecture, which combines the portability of HTML5 Web apps with a native container that facilitates access to native device features, will appeal to many enterprises.

The need for context awareness in mobile applications has increased with the capabilities of mobile devices, causing developers to consider both hybrid and native architectures. For applications to leverage location information, notification systems, mapping capabilities and even on-device hardware such as the camera, the applications need to be developed using either hybrid or native architectures. This has caused enterprise developers to consider alternatives to Web application development.

*Additionally Research firm Strategy Analytics forecasts that one billion HTML5-capable mobile devices will be sold in 2013. ABI Research sees more than 2.1 billion mobile devices with HTML5 browsers by 2016. IDC estimates over 80 percent of all mobile apps will be wholly or in part based upon HTML5 by 2015. The firm also predicts 2 million developers working with HTML5 in 2013.*
Examples of popular Hybrid Applications:

<table>
<thead>
<tr>
<th>Application</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESPN Scorecenter</td>
<td>A native app with updated scores within the app, directs users to a mobile website for further articles</td>
</tr>
<tr>
<td>LOTTECARD</td>
<td>100 pages written in HTML and used across platforms, a smaller number of custom developed native pages</td>
</tr>
<tr>
<td>Bank of America</td>
<td>A native app icon pushes users directly to the bank’s mobile site</td>
</tr>
</tbody>
</table>

The paper further explores the design, development & test considerations to deliver a great hybrid application experience.

Developing a Hybrid Application

In this section we will explore key design, development and test considerations to develop a compelling hybrid experience. We will also address development challenges that one encounters in hybrid application implementation and an overview of the various hybrid platforms available.

Logical Architecture of a Typical Hybrid Application

The below diagram depicts the high level hybrid mobile application architecture.

Hybrid apps use a web view control (UIWebView on iOS, WebView on Android and others) to render the HTML and JavaScript files. WebView uses the native browser rendering engine (not the browser itself). The HTML and JavaScript used to construct a hybrid app is rendered/processed by the WebKit rendering engine.

The key ingredient of the hybrid apps is the implementation of an abstraction layer that exposes the device capabilities (native APIs) as JavaScript APIs. This is a feature that is not possible with mobile web implementations because of the security boundary between the browser and the device APIs.
Hybrid Application Design, Development & Test Considerations

Design, Development & Test Considerations to Deliver a Great Hybrid Application Experience

UX Design and Custom Usability Guidelines

Mobile apps need an excellent user experience to make an impact and the limited screen space on mobile devices makes it all the more challenging to provide an effective application.

Developing for the web UI is typically more challenging when compared to a native UI. This is because the user interface is based on a web view and it is very easy to create applications that feel like web pages rather than mobile applications. Therefore it is important to ensure that a hybrid application offers good support for mobile UI features such as:

- Page transitions with slide up, slide down, pop, fade, flip features
- Swipes with tap, tap hold, swipe left, swipe right features
- Platform specific look and feel of controls
- Charting controls support

To create hybrid experiences that feel more like "native apps," it is important to consider the overall user experience, and how people interact with the application’s visual content during both the design and test phases. Additionally it is important not to try and make a hybrid web app work exactly like a native app; instead the focus should be on making the app usable as a web app, building on the familiarity on how the web works on specific devices.

In hybrid apps, the user-interface elements are defined in terms of the hybrid toolkit so the GUI design for hybrid applications should not only involve developing GUI components like toolbars, menus, images, lists, edit fields but also defining how the application handles the touch events, key strokes and gestures validated against the customized UX guidelines (combination of the UX guidelines of multiple platforms) so as to provide familiar behavior and a better usability experience.
Application Performance

It has been observed that 60% of users abandon a mobile application if it does not load within 3 seconds, so performance should be one of the key concerns for mobile application developers.

There are three scenarios where hybrid applications might present limitations on the application performance. For a day to day application there is almost no difference in code execution speed but in some resource intensive tasks the performance lag can be noticed.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Native Application</th>
<th>Hybrid Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Compilation</td>
<td>The code is compiled in development environment and then run on the client</td>
<td>The JavaScript code runs fully on the client, hence there is a slight difference in time taken for execution of code</td>
</tr>
<tr>
<td>CPU/RAM Speed</td>
<td>Native code runs on the devices’ CPU/RAM directly which makes native code faster in the</td>
<td>JavaScript is run by the devices’ browser engine whose speed is dependent on the CPU/RAM of the device</td>
</tr>
<tr>
<td>User Interface rendering</td>
<td>Native applications leverage the raw GPU power of the device directly, so for intense 3D rendering native code almost always can render comparatively more fluidic than hybrid code</td>
<td>Hybrid application would have to rely on HTML canvas based rendering which in turn is based on the browser engine which again is dependent on the CPU/GPU of the device</td>
</tr>
</tbody>
</table>

As JavaScript run time compilers gain efficiency and speed this difference in performance is reducing and they will be the same very soon. However in the interim these limitations can be overcome by adopting a REST based Service Oriented Architecture (SOA) for the backend of a hybrid application.

Security Management

HTML5 is not a single technology stack, but actually a combination of components and technologies. The components include XMLHttpRequest (XHR), Document Object model (DOM), Cross Origin Resource Sharing (CORS), and enhanced HTML/Browser rendering. The technologies include localstorage, webSQL, websocket, webworkers, enhanced XHR, and DOM-based XPATH. These various components have expanded the security breach surface and have made addressing security issues an imperative during the design and testing phases.

Some design techniques that can be adopted to mitigate the security vulnerabilities are:

Communication APIs

Hybrid applications use webviews which are native controls and do not offer the same level of support as a browser does. It is recommended to do a handshake and thoroughly validate messages while adopting web messaging and Cross Origin Resource Sharing (CORS) and websockets for communication across domains and between files of different origin.
Local Storage and Client-side databases

For hybrid applications it's recommended not to store any sensitive information in local storage. This is because the underlying storage mechanism may vary from one user agent to the next and any authentication your application requires can be bypassed by a user with local privileges to the machine on which the data is stored.

Sandboxed Frames

With many sites, including social networking sites, allowing other pages to load in an IFRAME it is a good security practice to use the sandbox attribute of an IFRAME for untrusted content.

Offline Data Management

Native apps generally have the most capable offline functionality, as they have the ability to run background processes making it easier to gracefully handle offline scenarios in a native app. Mobile web apps currently have limited capabilities to run background processes (when the app is closed) or worker threads (while the app is still open) - for example, it would be impossible currently to build a mobile web app that plays music in the background while other apps are running.

Hybrid apps build a bridge between mobile web and native apps, allowing background processes to run within native code while still using web code for the bulk of application functionality. However when the HTML5 code is executing within the mobile device then it would work very well, hence hybrid applications will be capable of working offline as well as a native application. If your app requires store-and-forward type processing for offline updates or background processes, consider building a native or hybrid app.

UI/UX Testing

UI testing for hybrid applications involves testing across different devices and Operating Systems as each device carries its own conventions which its users expect.

The recommended test approach is to create a common subset of custom guidelines which consists of the best of the UI guidelines of the target platforms. This approach will ensure that the UI elements (like toolbar, menus, images, lists, edit fields etc.) in the hybrid application work consistently across the multiple target platforms and provide a better usability experience.

UI testing should also focus on validating how the application handles the touch events, mouse events, keyboard strokes, gestures against the custom UI guidelines.

Example of a custom UI guideline: For a finger-friendly design Apple’s iPhone Human Interface Guideline recommends the ideal touchscreen target size of 44 pixels wide and 44 pixels tall. Microsoft’s Windows Phone UI Design and Interaction Guide suggests a touchscreen target size of 34 pixels with a minimum touch target size of 26 pixels. Nokia’s developer guidelines suggest that the target size should be no smaller than 1cm x 1cm square or 28 x 28 pixels. So in this scenario the best of the guidelines (ie) Apple's guideline is followed while creating & testing the touch target in the hybrid apps.
Security Testing

HTML5 offers greater enhancements for rich web applications, but also presents new opportunities for security breaches. This makes the security testing an important aspect in implementing hybrid applications. Security testing of hybrid applications should focus on testing scenarios for browser vulnerabilities, privacy concerns, sandboxing and permissions issues and encryption of data.

For example scenarios around testing for browser vulnerabilities would involve cross-site request forgery (CSRF) and leveraging CORS to bypass same origin policy (SOP), ClickJacking and phishing by mixing layers and iframe, and using WebSockets for stealth attacks.

Performance Testing

Load testing is another critical step in a hybrid application development as test requirements for performance attributes are much steeper due to the unpredictability in terms of dependency on mobile carriers and data usage both of which can affect the speed and ease of use of the application.

The hybrid application should be validated against performance benchmarks such as time (UI response time, transaction time), battery, network bandwidth, memory and CPU – (Responsiveness, memory leaks) and binary size – (Load Time, RAM usage)

A combined approach of tool based and unit framework approach is used to test the performance of the hybrid app to achieve a higher rate of success in terms of cost and reliability.

Test Automation

Testing a mobile application requires strong object recognition capabilities and the usual approach toward testing the UI is automation. However for hybrid applications the best approach towards testing the UI is a combination of manual (visual analysis) and automation (object analysis) as fluidity of interactions cannot be understood by the object analysis alone. Some guidelines towards test automation are:

- Use of simulators/emulators instead of real devices in most of the functionality related testing
- Use of techniques for object identification like combination visual based and unique identifiers so as to create robust and reusable scripts

Development Challenges & Recommendations

HTML5 at the time of writing this article is still work in progress and evolving; for example webSQL has been deprecated, there could be additions or even deletions in future.

HTML5 is a combination of HTML, JavaScript and CSS which means that an HTML5 developer must be well aware of all three of them as compared to an apple developer only needing to know objectiveC and an android developer only needing to know Java.
HTML5 features differ in implementation between browsers for example webkit in safari, webkit in chrome browser and webkit in opera vary based on the makers of the browser.

Development IDEs available for HTML5 are very premature as compared to that of xCode and Eclipse, hence HTML5 could take more time to build and stabilize.

Many older devices with old browsers do not support HTML5 there must be a lot of tweaking and hacking done to get around these limitations.

HTML5 is unable to render graphics as well as native code.

**Hybrid App Platforms**

| **PhoneGap** | Is a native packaging framework for apps built using any hybrid technology. Typically it just provides the essential native-only device specific APIs exposed through a bridge architecture. The rest is all pure HTML/Javascript/CSS for UI (this has limitations when it comes to platform specific widgets) |
| **Titanium** | Titanium provides a development framework/language abstraction for cross platform development. But native toolchains are required, as the Titanium framework provides the abstraction layer over native specific widgets and functions. This allows for close-to-native end user experience. Development Language - Python / HTML5 / Javascript |
| **Sencha** | Originally a cross platform (cross browser) Javascript library. A lot of the cross browser testing effort is reduced due to the compatibility that Sencha takes care of. Sencha Touch also provides a packaging SDK for the native shell wrapping required for publishing to Native appstores and accessing device specific functions |
| **IBM Worklight** | IBM Worklight is more than just a cross platform app development framework. It provides a middleware, security infrastructure and MAM. IBM Worklight crossplatform app framework can work with PhoneGap (Apache Cordova) for native application packaging as well |
A Practical Implementation of a Hybrid Application

Scenario – One Framework Supporting Multiple Platforms

A safety equipment manufacturer commissioned Mindteck to build a mobile planning editor tool to enable their field engineers to map and mark the position of safety equipment on the floor plan of an industrial / commercial setup.

The same framework along with a common UI was to be used on a different mobile platforms (like Android/Apple/BlackBerry) as well as desktops and result in productivity improvements.

The other key need was to develop reusable components that can be used by developers to further enhance and utilize this framework.

### Hybrid Solution

- **A cross platform application framework based on HTML 5, JavaScript & CSS that can work on multiple devices**

- **The framework was developed as a set of loosely coupled components that can be reused across multiple such similar implementations**

- **An implementation of the framework for the planning editor tool was developed as a working prototype of the solution**

- **GUI was built to test the framework and was based on an HTML5 canvas from where a human can use click or tap events to interact with the framework**

### Why Hybrid Approach?

- **Same code base working exactly with the same User Experience in Apple, Android and PC platforms**

- **A single application framework which exposes Application programming interface (API) for further enhancement**

- **Functionality to be added for individual properties without disrupting the overall architecture**

- **Developers will be able to further reuse the framework and extend the application as per the business needs**

---

**About Mindteck**

Mindteck, a global technology company, has served medical device manufacturers, top-tier semiconductor companies, analytical/scientific powerhouses, equipment OEMs, public sector entities, and service companies in the Global 2000 for over 22 years. The company’s depth of knowledge and niche expertise in product engineering is complemented by dedicated Centers of Excellence in Wireless Design and Storage Testing. Presently, the company employs over 1,000 individuals throughout offices in the US and UK, Singapore, Malaysia, Bahrain and India. It also has four development centers (US, Singapore and India (Kolkata, Bangalore)). The company is listed on the Bombay Stock Exchange (BSE 517344), an ISO 9001:2008, ISO 27001:2013 certified company and ISO 13485:2003 certified to serve the medical electronics industry; assessed at CMMI Development Version 1.3 - Level 5.