Native

vs

Web

vs

Hybrid

How to Select the Right Platform for Your Enterprise's Mobile Apps

kinvey • MOOVWEB

CONTENTS

INTRO

The native vs. web debate has inspired countless articles, daily Twitter discussions, and panels of experts evangelizing why one approach is superior to the other. More recently, hybrid has been thrown into the mix as a viable third option, making these conversations even more complex. There is a compelling use case for native, web and hybrid - but the problem is that nobody is making those use cases clear, particularly when it comes to the enterprise. Knowing how to approach app development is a key first step for enterprises to meet this demand and contribute to the overall apps market.

In this eBook, we've isolated the pros and cons of native, web and hybrid approaches and supplied use cases and expert opinions to help enterprise companies make smarter decisions about their mobile platforms. Ultimately, the decision boils down to: what's your priority?

CHAPTER 1
NATIVE

We're all familiar with native apps. Those downloadable, colorful icons on the home screen of your device are built using tools specific to that device.

When deciding whether your enterprise should develop your app natively, you should consider the inherent benefits and drawbacks to native development, and see if it is the best option given your specific company needs and priorities.

Some key questions to consider:

Will your app make use of native-only features?
Since native apps run on the device operating system - as opposed to the web browser - they are able to make full use of the device's hardware and functionality. That allows native apps to access device functions like GPS, Bluetooth, offline access, SMS messaging, push notifications, background downloading of data, and address book. Let's say your organization wants to build a CRM app to enable your sales force to update their pipeline on the go. Sales people often travel, and offline access is a key feature for someone who may want to add information to the app on a flight. A native implementation of such an app may be the best approach.

Now, as we'll cover later, device APIs are making their way to the browser to allow sites to gain capabilities that were once strictly native. However, this has been a slow transition. For example, it wasn't until iOS6 that the iPhone browser began allowing access to the camera and photo library, a full four years after native apps got that capability.

Will your app push the limits of the hardware?
The number one benefit of a native app is high performance. Games like Angry Birds or Infinity Blade are built as native apps to take maximum advantage of the hardware and operating system features. Running in a mobile browser, games and processor intensive apps could not get the extremely low levels of latency that these highly interactive apps require.

For these reasons, end-users often have quite a different perception of native apps vs web apps. Users tend to think of a native app as being about function, while they look at web apps or mobile sites as content. Even though this perception isn't accurate it's an expectation that exists due historical limitations of browsers in the past. Another contributing factor is the look-and-feel of a native app. Out-of-the-box native apps tend to look and feel more like an integrated part of the device because they automatically inherit the operating system's user interface. While it's possible for web apps to emulate this experience in HTML5, doing so is extra work on the part of the developer.

Will your audience download an app?
The process for a user to obtain a native app is very well defined. Native apps are submitted to app stores, like Google Play or the Apple App Store. This means it must first go through an approval process before it can be made available for download, which adds time and friction. But it also makes for a simple distribution process - users will know exactly where to go to download the app. Some business apps may be distributed through an organization's private app store or through enterprise app store options offered by Google and Apple.

That said, you should be mindful that just because your enterprise app is on the virtual shelf - be it an enterprise app store or a consumer-facing store - it doesn't mean users will flock to download it. There are over a million apps in both Google Play and the Apple App Store, and it's easy to get lost in the mix without an effective marketing campaign. So you're still going to have do the leg work to get your company's app in front of potential users - even if those users are your organization's employees.

If your organization is offering an internal app for employees, keep in mind that downloading this app means that it will be on their personal device even when they are away from the office - assuming the company practices a Bring Your Own Device (BYOD) policy. This could be convenient for users, but it could also be perceived as yet another app that crowds their screens. Knowing the intentions of the audience is key when determining the right platform for your app.

Do you plan on selling your app?
App stores create a standard way to monetize your app, allowing your organization to charge users for downloading it. If you plan on having users pay for the app, this is a simple monetization solution. However, app stores take a percentage of both the sale of the app (e.g. 30% in the Apple App Store) and any digital in-app purchases.

USERS TEND TO THINK OF A NATIVE APP AS BEING ABOUT FUNCTION, WHILE THEY LOOK AT WEB APPS OR MOBILE SITES AS CONTENT.

When not to use native:

If you don’t have the development time and talent
Developing natively means facing the difficult decision of picking and choosing which platforms and devices to support. Just iOS users? What about Android? And can you really afford to ignore that small fraction of users still clinging to their BlackBerry? If you are planning on cross-platform support, your company will need to have both the time and the development talent to build separate apps, in different programming languages, for each platform.

If you don’t have consistent and devoted users
When we all got our very first smartphone, the first thing we did was rush to the app store to find out all the amazing new apps we could download to populate our home screens. But a handful of years later they’ve all started to pile up and app fatigue has started to set in.

If you can keep your users consistently coming back to your app, they won’t mind giving you that real estate on their home screen. But if you don’t stay in the regular rotation, then your app is likely to get deleted when it’s time to do a little spring cleaning.

If you don’t have the back-end infrastructure
Often the most difficult part of building an app isn’t the app itself but integrating it into your company’s backend infrastructure. Native apps are reliant on APIs to power their content and functionality. That means you need to have clean and well-documented APIs that are capable of delivering all the data to your app. If you haven’t done the work to build out an API infrastructure this could add considerable work to delivering your app. However, there are solutions including backend as a service (BaaS) providers, such as Kinvey, that make it easy to integrate any app with those enterprise backend systems plus data from any third-party service.

Native Use case: L'Oréal

The goal of the L'Oréal Beauté Concierge app is to allow members of the L'Oréal professional community to quickly order L'Oréal products at any time. The app is available to professionals on an invite-only basis, providing them with customized browsing and ordering for exclusive L'Oréal products.

When L'Oréal met with Fuzz Productions, the development strategy boiled down to two factors. First, the core functionality of the app was not already on the L'Oréal website. At the same time, they already had the necessary APIs on their backend to deliver data to a native template.

"The level of backend and API support that a client can supply is a huge factor in the decision to build purely native," said Nathanial Trienens, Cofounder & CEO of Fuzz Productions. "In some cases producing an API that can support a fully native enterprise app can be a bigger expenditure than the app itself."

With those considerations, and the need to only build for iOS, L'Oréal and Fuzz made the decision to develop the app natively.

Native Tools:

- XCode
- Android SDK
  - Eclipse
  - Android Studio
- Visual Studio (Visual Studio Express 2012 / Visual Studio Professional 2012)

CHAPTER 2
HTML5

An HTML5 app is an app that behaves like an native app but runs in a web browser. Like any web page, an HTML5 app may need to be downloaded at runtime. Or, if your desired outcome is for the app to have a native appearance, it can be packaged as a self-contained app running in its own web view (see the Hybrid section later in this ebook).

HTML5 has been the primary rival of native ever since it has gained traction in the development world. It seems you can't read an article about HTML5 without a comparison to its opposite - and vice versa. But the differences are important to note when an enterprise is developing its mobile strategy and deciding, "which approach makes most sense for our app concept?" While the answer is not cut-and-dry, there are compelling reasons that an HTML5 approach may be the most appropriate for many organizations' needs. In fact, 75% of Fortune 500 companies are taking steps to deploy HTML5 mobile apps, according to an IBM whitepaper.

Some key questions to consider:

Is multi-platform support at the top of your list, but time and resources are sparse?
Taking a broad device approach is essential for many enterprises, whether they are building consumer or employee facing apps. In the age of BYOD, workers use a myriad of different devices to access information in the workplace. Some organizations have the resources to form teams that specialize in developing for different platforms: one that focuses on iOS apps, one for Android, etc. This requires not only the budget for multiple teams, but also the task of hiring experts in each programming environment - a process that may be more difficult than finding developers with web development skills.

Enterprises that are in the early stages of planning their mobile strategy may not yet have solid internal mobile organizations in place. Building native apps is a repetitive process if multi-platform support is required, especially for those companies without mobile teams and resources at their disposal.

HTML5 allows for a single codebase which developers can deploy on multiple platforms quickly; they can write it once and have that same version run on several operating systems. Platform-specific teams aren't necessary, and the burden of building one app over and over to run on multiple devices is lifted. This is particularly beneficial to organizations looking to build apps for a variety of departments and tasks.

While developing native apps to run on all platforms is certainly not impossible, it does require talent, budget and resources beyond the reach of many companies. The most important questions to answer here are: Is multi-platform support important, and if so, do my teams and resources allow for building the same app over and over?

Is a collaborative community important to your development team?
HTML5 is known for it's open, large collaborative community of developers. There are arguably more developers skilled in web development than in developing for iOS, Android, etc. Enterprise developers building HTML5 apps will have an easier time finding online support, answers to technical difficulties, and a large offering of open-source tools, projects and other free resources to assist in their development efforts. Knowledge-sharing can be hugely beneficial to enterprise developers looking to take advantage of training, sample code and documentation that already exists in the HTML5 community.

Do you plan on making frequent updates to your application?
HTML5 allows developers to bypass the approval process of app stores. Giving updates directly allows developers to bypass the approval process of app stores. Changes are simply pushed to the production servers, and users see the updates immediately. In previous versions of iOS, for example, users had to manually download app updates, giving HTML5 another leg up on that front. However, iOS 7 introduced automatic app updates, making for a more seamless update experience similar to HTML5. The real advantage of web apps here is forgoing any sort of approval process from app stores, therefore getting the improved app into the hands of users faster.

Will you be serving an audience who prefers not to download apps?
While this is a niche use case, there are always those users who would much rather digest content in a mobile web browser than in a traditional mobile app. With millions of apps available in each app store, many experience app fatigue after having filled their device with dozens of applications. It's always helpful to get a sense of who your app will cater to, and if it happens to be those who don't like downloading apps, then HTML5 is a good solution.

KNOWLEDGE-SHARING CAN BE HUGELY BENEFICIAL TO ENTERPRISE DEVELOPERS LOOKING TO TAKE ADVANTAGE OF TRAINING, SAMPLE CODE AND DOCUMEN-TATION THAT ALREADY EXISTS IN THE HTML5 COMMUNITY.

Smartphones are one of the most personal items we carry and giving up control of their personal devices may alienate employees. In a BYOD environment enforcing data security policies for native apps may require installing mobile device management (MDM) software onto an employee's phone. Deploying an enterprise app in the browser bypasses the need for an MDM solution, and lets users access their work data securely while also keeping employees' personal devices feeling "personal".

Do you already have a web app and want to make the transition to mobile web?

If your company has an existing web app but plan on making the transition to mobile, you can leverage your existing code and build a mobile web version faster. Or vice versa - if you are starting with mobile, but planning to transition to web shortly, some of that HTML5 code could be reused. For example, developers can use a responsive delivery process to migrate a traditional desktop site to mobile devices without the need for APIs or backend integration. Because it inherits the business logic of the original website, the overall project has a faster time to market and lower total cost of ownership than other approaches. Enterprises such as 1800-Flowers, Cox Communications, and Macy's have successfully used the Moovweb platform to deliver mobile and tablet websites using this technique.

On the other hand, a native app on iOS, for example, is built in a completely separate environment using a different language (Objective-C), and so it is not transferrable to a web platform. Not only would you have to write this code from scratch on the client, but there's also code on the server that you must write, both of which can be a headache. In addition, you may need to develop an entirely new set of APIs to support a native app.

Do you offer a subscription service?

Many cross-device services (e.g. Dropbox) don't monetize their mobile apps, but the service itself, typically through their desktop web interface. If your service is offered in a subscription model, web apps make this process much smoother. Subscriptions can be a pain when done through mobile app stores, and often more costly. For subscriptions purchased within the Apple App Store, for example, Apple takes 30% of the transaction. The Financial Times famously migrated its subscribers from an iOS app to an HTML5 app to avoid this 30% fee.

Will your app be heavily content-based?

HTML5 can be ideal for apps that require minimal function and focus primarily on content. For example, a newsfeed reader app whose main function consists

of scrolling through and reading articles is a prime candidate for HTML5. When mapping out the functionality of your apps, be sure to boil it down to only the essentials. Oftentimes, less is more.

When not to use HTML5

If advanced performance is a top priority
As we described in the native section, for some use cases HTML5 apps may be slower than native. If you are building an app that requires intense graphics, low-latency, or computational horsepower, HTML5 is not the best option.

If your app must access native device functions or services
In need of push notifications, offline data access, etc. for your app? There are currently only a small number of API's that allow HTML5 apps to connect to these. That said, the trend is slowly moving towards more device access. Emerging smartphone platforms like Tizen and FirefoxOS are pushing this boundary even further by making web apps "first-class native apps" and providing a rich set of device APIs available to HTML5. See the sidebar for examples of APIs.

If you're looking to monetize your app
Unless you're offering a subscription service, as described earlier, monetizing an HTML5 app is not an easy task. Aside from the Amazon App Store for Kindle devices and Google Chrome Web Store, developers must distribute their web apps directly to the public. App Stores give native apps a strong advantage on the purchasing front.

Pitfalls to avoid when building HTML5 apps

Developing HTML5 mobile apps is rarely accomplished without a few challenges. As long as your development team knows what to expect and how to tackle these roadblocks, they can be cleared in no time.

Browser/platform combinations
While one of the great benefits of HTML5 is the ease of cross-platform development, there is a slight catch. Different browser/platform combinations support different HTML5 features. Test ahead of time to become familiar with which features are supported on which platforms, and use tools like Modernizr to gracefully degrade the experience on older browsers. In other words, build for the latest and greatest features, and make sure that older platforms still have at least a usable experience.

Avoid jank
Another common pitfall especially important in mobile environments is “jank,” seen in stuttering UI and animations. This occurs when the rendering engine of the browser can’t keep up with the complexity of the interface or the intensity of the computation happening in the background. This causes lag when scrolling or stutters in animations within the app, which impact the experience.

Device APIs for HTML5 Apps:

- Apache Cordova
- Qualcomm HTML5 Device APIs
- APIs from HTML5 Rocks
- Motion data (accelerometer & gyroscope) from Mozilla
- Camera API from Mozilla
- Device Orientation API
- getUserMedia API
- Airplay API
- Speech Synthesis API
- New APIs in Mobile Safari

HTML5 Tools:

Take advantage of existing tools to help simplify your development process. Here we've collected a handful of useful tools for you.

Application Frameworks:

- Backbone gives you a minimalist, un-opinionated structure to web applications by providing models, collections and views and connects it all to your existing REST API.
- Ember.js takes a different approach, offering a highly structured, convention over-configuration approach to web development that borrows heavily from the philosophy behind projects like Ruby on Rails.
- Angular offers a unique approach to application structure, attempting tomimic the future of current web standards, but often at the expense of a steep learning curve.
- Flight from Twitter is a lightweight, component-based JavaScript frameworkfor assigning behavior to DOM nodes.
- React from Facebook is a JavaScript library for building user interfaces.

DOM Frameworks:

- jQuery Mobile is a unified, HTML5-based user interface system for all popular mobile device platforms.
- Enyo is a JavaScript app framework, "enabling developers to build native quality HTML5 apps that run everywhere."
- Sencha Touch is a mobile app framework based on HTML5 and JavaScript.
- Bootstrap and Foundation are powerful responsive, mobile-first front-end frameworks for faster and easier web development.
- Uranium is a fast, lean interaction widget library for mobile and desktop websites.
- Polymer is a new type of library for the web, built on top of Web Components, and designed to leverage the evolving web platform on modern browsers.
- Modernizr is a Javascript library that makes it easy to detect the capabilities of the user's browser.
- HTML5 Boilerplate is a collection of best practices and experience from hundreds of developers to provide a solid starter template for modern web apps.
- Grunt is the defacto standard build tool for JavaScript projects.
- Node-build-script includes tasks for producing customizable HTML5 projects based on HTML5 Boilerplate.
- HTML5 Project Builder is a fork available on Github allowing you to create a backend for any client side website.

HTML5 Development Tips from Experts

Boaz Sender, CEO of Bocoup:

"Engineers who have been writing non-web apps (such as desktop software) typically don't take the time to actually use JavaScript. It's a good idea to invest time to learn JavaScript, either through training or time prototyping. Now that the language is being taken seriously and is building first class software, developers should absolutely invest the time to learn it."

Dion Almaer, Vice President, Mobile Engineering at Walmart.com:

"Be clear on your goals and non-goals. Build to the strengths of the platform to deliver the best experience for the user. Don't fall into the trap of only focusing on the tech."

Max Lynch, Founder of Drifty:

"Don't tie yourself to the URL. Native UI paradigms are quite a bit different from their browser counterparts and trying to map each view of the app to a URL will be complicated and limiting."

Hampton Catlin, inventor of Sass and CTO of Moovweb:

"The web is the only universally accessible platform on the planet. If you built for a browser and use good practices, it will work anywhere and everywhere."

Nathanial Trienens, Cofounder & CEO of Fuzz Productions:

"*When we develop natively the cost of iOS and Android builds are, to a large degree, separate. But with mobile web development, it's one build across all devices and platforms."*

CHAPTER 3
HYBRID

It's only natural that from the native app and web app, each with their own set of advantages and disadvantages, would come a third classification of apps – the hybrid app. Hybrid apps are native apps that leverage "web views" to display HTML content. Web views are native widgets that display web pages but lack the address bar and navigation controls that appear in a normal browser. This lets developers embed web content inside an app and seamlessly blend both native elements and web content on the same screen. These hybrid apps allow enterprises to get the best of both web apps and native apps. With hybrid apps the question isn't web or native, but instead when and how to use each.

While web and native app classifications are binary, the hybrid app is much more of a spectrum. It's entirely possible - and quite common - to develop an app that is almost entirely built from HTML, CSS, and Javascript, only to add later native elements like the static bottom menu for navigation, GPS calls for location lookup, or camera access for barcode scanning. On the flipside, it's also possible to develop a mostly native app with only a few screens powered by web views.

With all the flexibility and benefits that hybrid allows, enterprises more and more are choosing to develop with some combination of native and web features. And Gartner has predicted that by 2016 most apps will be hybrid.

When should my organization choose hybrid?

Enterprises often choose hybrid apps when looking to develop across a range of devices, but also aiming to take advantage of features that a purely web app cannot give them. Web views enable enterprises to reuse code and content that run on the browser for iOS, Android, or Windows apps. Only the native elements of a hybrid app need to be rewritten for each new app's operating system, which can create considerable cost and time to market savings. Hybrid is an ideal choice when multiple platforms need to be supported.

The choice to develop a hybrid app might also be dictated by content or features that require constant iteration. Web views inside a hybrid app allow enterprises to remain agile - tweaking, changing, and split testing for constant improvement. Whereas making those changing in native code could entail a long arduous process of submitting app updates for even the smallest of changes.

Employee skill set can also be a deciding factor in choosing to develop a hybrid app. Compared to a purely native app, hybrid apps require less Objective-C and Java programming. Instead, much of the app can be developed using the front-end development talent that many enterprises already have on staff.

Finally, hybrid apps can relieve the backend integration issues that exist with native apps and avoid the need to build out APIs. Building and supporting APIs for app development can be an unanticipated cost of developing an app. According to Trienens of application development company Fuzz Productions, “In many cases producing an API that can support a fully native enterprise app is a much bigger expenditure than the app itself.” Using web views, an enterprise can pass through functionality that already exists on the web and only needs to build out APIs for new functionality that doesn’t exist yet. The Moovweb platform is one way to easily leverage existing business logic and content from a desktop website into a hybrid app without the need for APIs.

When should my organization not choose hybrid?

Hybrid apps gain a lot of benefits of native apps, but as discussed earlier native remains superior for high-performance or low-latency interactive applications on mobile devices. For now, augmented reality, 3D rendering, and multimedia are often prime candidates for native. On desktop machines, we’re already seeing browsers close the gap with native applications and we can expect the same to eventually happen with mobile over time. In addition, a hybrid app can “split the difference” by using web content for some areas and still reserve native components for the high-performance functions. For example, the Macy’s iPhone app uses native components for its in-store mapping features powered by Meridian, yet the wedding registry functionality is inherited from the desktop website using web views and the Moovweb platform.

In some cases the lower latency of native can yield benefits for interactive applications. For social apps on the level of Facebook or LinkedIn - two companies who have famously pivoted from using HTML5 in their apps - there may be a positive return on investment in building and managing separate native apps for different operating systems. However, there are complex

IN MANY CASES PRODUCING AN API THAT CAN SUPPORT A FULLY NATIVE ENTERPRISE APP IS A MUCH BIGGER EXPENDITURE THAN THE APP ITSELF.

and controversial reasons these companies switched away from HTML5 with hybrid (see sidebar). Most importantly not all businesses have the engineering resources to devote to pure native development and not all use cases have the same constraints as the winner-take-all social networking industry. In scenarios where more than one platform is supported, the cost of going native can go up at a faster rate than the UX improvements gained. This is an important consideration for enterprise apps where competitive conditions are different and ROI considerations are paramount.

Facebook ditches hybrid app:

When a company as influential as Facebook makes the move to a native app it's bound to make some waves. But when you look at the factors that went into decision, it isn't an open-and-shut case against hybrid.

For starters, there are conflicting accounts on the reason for the transition. On the technical side, Facebook's engineering staff have pointed the finger at their app's main user interface, the newsfeed. Despite its simple appearance, the newsfeed is an infinite list of mixed media containing text, links, image galleries and videos all on the same screen. Managing memory and scrolling performance with a dynamically refreshing infinite list was difficult inside a web view, especially without proper tooling support. However HTML5 company Sencha has challenged these claims and built a performant version of the newsfeed dubbed "FastBook" using Facebook's APIs.

Meanwhile it's been reported that instead of being a purely technical decision, the Facebook's transition was the result of a lack of mobile resources and a political breakdown between engineering teams. And as Facebook's engineers and others have pointed out, the current so-called "native" Facebook app still makes use of web views as a fallback for new and rapidly changing content.

Finally, it's worth noting that applying Facebook's experience to other businesses is misdirected. Facebook has significant engineering resources that they can devote to native development and operates in the unique dynamics of the social networking industry. This makes them a fascinating yet unrepresentative case study. There are definite advantages to both hybrid and native approaches but make sure you know which are the right ones for your business.

Best Practices

When building a hybrid app there are a few best practices that can help an enterprise get the most out of their app.

Don’t mimic the OS
Attempting to duplicate design patterns of the native environment using web views is precarious. The app risks sliding into the Uncanny Valley – or that feeling that something is just not quite right. Users know how it should behave, and for some reason it’s just a little bit off, which is rather unsettling.

Instead develop the app experience in the web views, leave elements that are unique to the operating system, like transitions, for the native layer.

This will also help you avoid the awkward situation where a design pattern from one operating system is mistakenly delivered to another. There may be no faster way of unnerving an Android user than sending them iOS-looking elements.

Make it seamless
On the best hybrid apps it’s difficult to tell which elements or features are native, and which are web views. When a hybrid app is on its game it feels like one seamless experience. This often requires one layer informing the other, but the result is an app that never breaks character.

For example, in the 1-800-Flowers iPhone app developed by Moovweb, the products list screen is a web view, but tapping the gallery button takes the user to a purely native screen. Even though they are built differently, the native and web view screens communicate so the same product is displayed in either interface. It’s touches like these that can make the boundaries between native and web views seamless and maintain a consistent experience.

Making sure HTML content is designed for an app environment is also critical for a quality hybrid app. App screens that load without sliding transitions or have the same header menu as your website will break the user’s expectations for an app experience. This isn’t an academic issue since Apple’s App Store approval process will reject hybrid apps that fail in this manner. The accompanying sidebar describes best practices that will help your hybrid app feel like an app instead of a website.

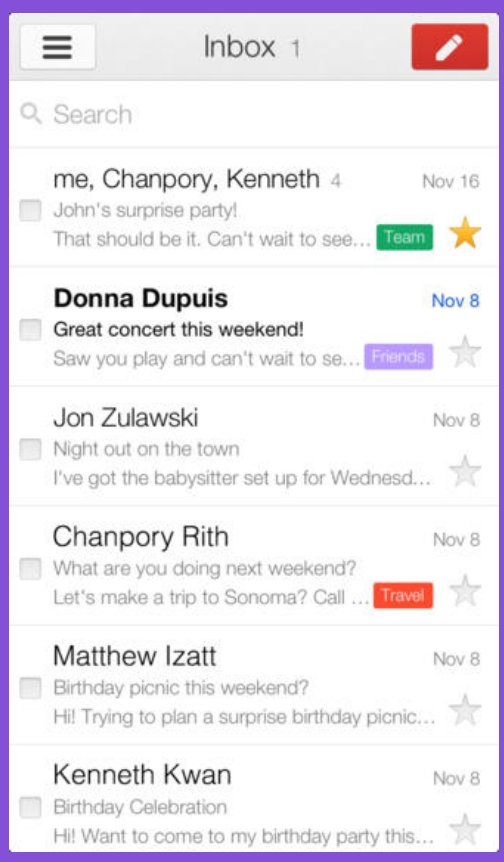
ATTEMPTING TO DUPLICATE DESIGN PATTERNS OF THE NATIVE ENVIRONMENT USING WEB VIEWS IS PRECARIOUS. THE APP RISKS SLIDING INTO THE UNCANNY VALLEY – OR THAT FEELING THAT SOMETHING IS JUST NOT QUITE RIGHT.

Tips for seamless Web Views:

- **Remove headers and footers:** Nothing screams “web page” like seeing the header or footer of a website inside the screen of an app. Make sure you remove these areas and replace them with native components or HTML5 that emulates native components of the target operating system.
- **Remove tap highlights and callouts:** The default tap highlights and callouts generated by a web view are another browser behavior that can break the native illusion of a hybrid app. Fortunately you can easily turn these off by using CSS. For WebKit based web views (i.e. iOS and Android) the relevant CSS are properties -webkit-tap-highlight-color and -webkit-touch-callout.
- **Handle external links:** Remove links for third party websites from your HTML content, or make sure your app opens them in the browser or a browser-like interface within your app.
- **Fix tap delay:** Mobile web views introduce a 300 millisecond delay responding to user taps which can make hybrid apps feel slower than native ones. FastClick is one of many open source libraries that fixes this issue.
- **Use CSS3 Effects:** Instead of JavaScript, use CSS3 transforms, animations, and transitions for smoother animations in web views. When applied properly some CSS3 properties will result in hardware accelerated animations for even better performance.
- **Fix scrolling:** By default on iOS devices, web views have a different scrolling momentum than native screens. Changing the scrolling properties will remove a subconscious but persistent reason hybrid apps can “feel” different than native ones.

You didn’t even know these were Hybrid, did you?

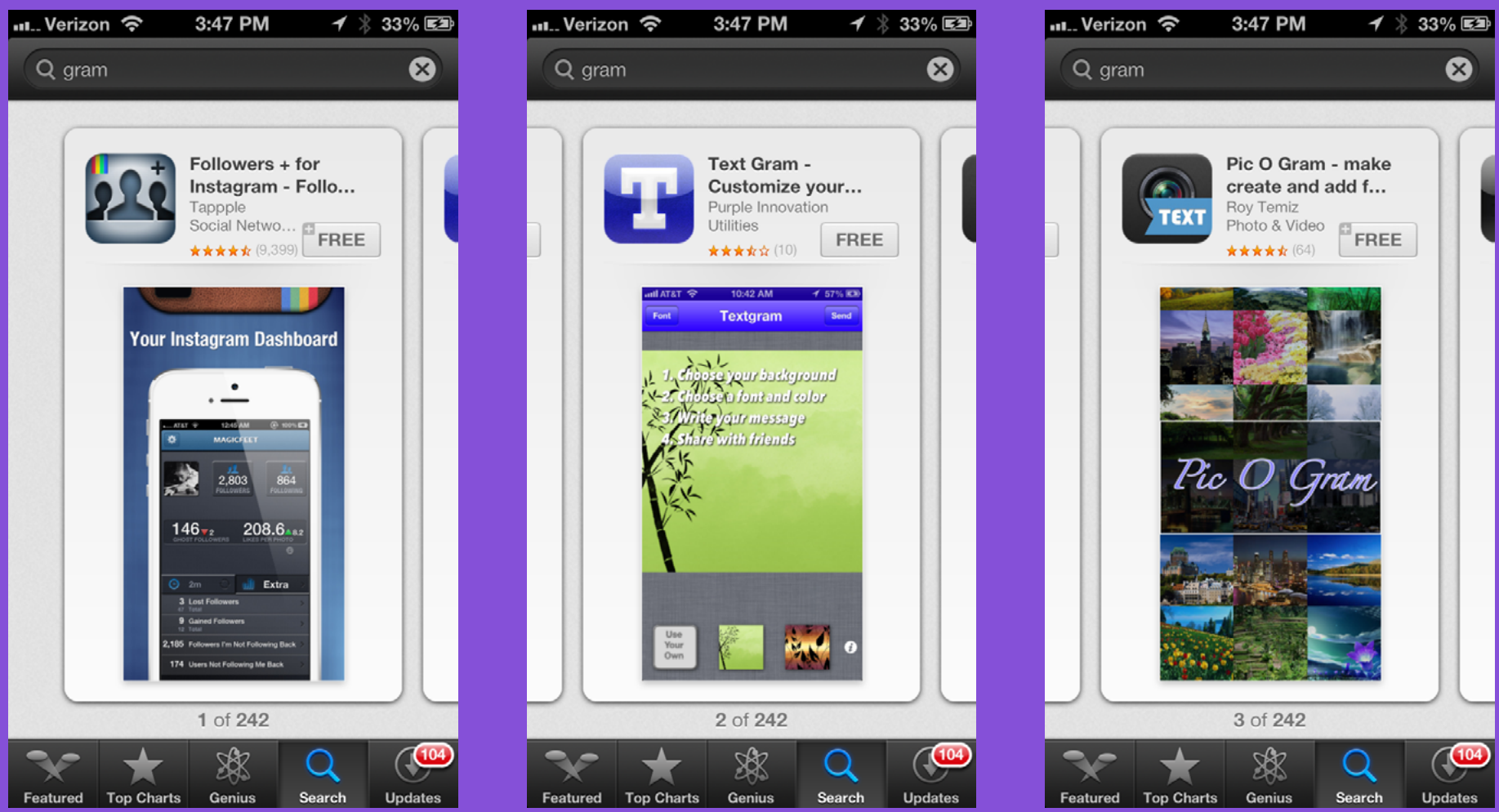
iBooks uses web views and native elements



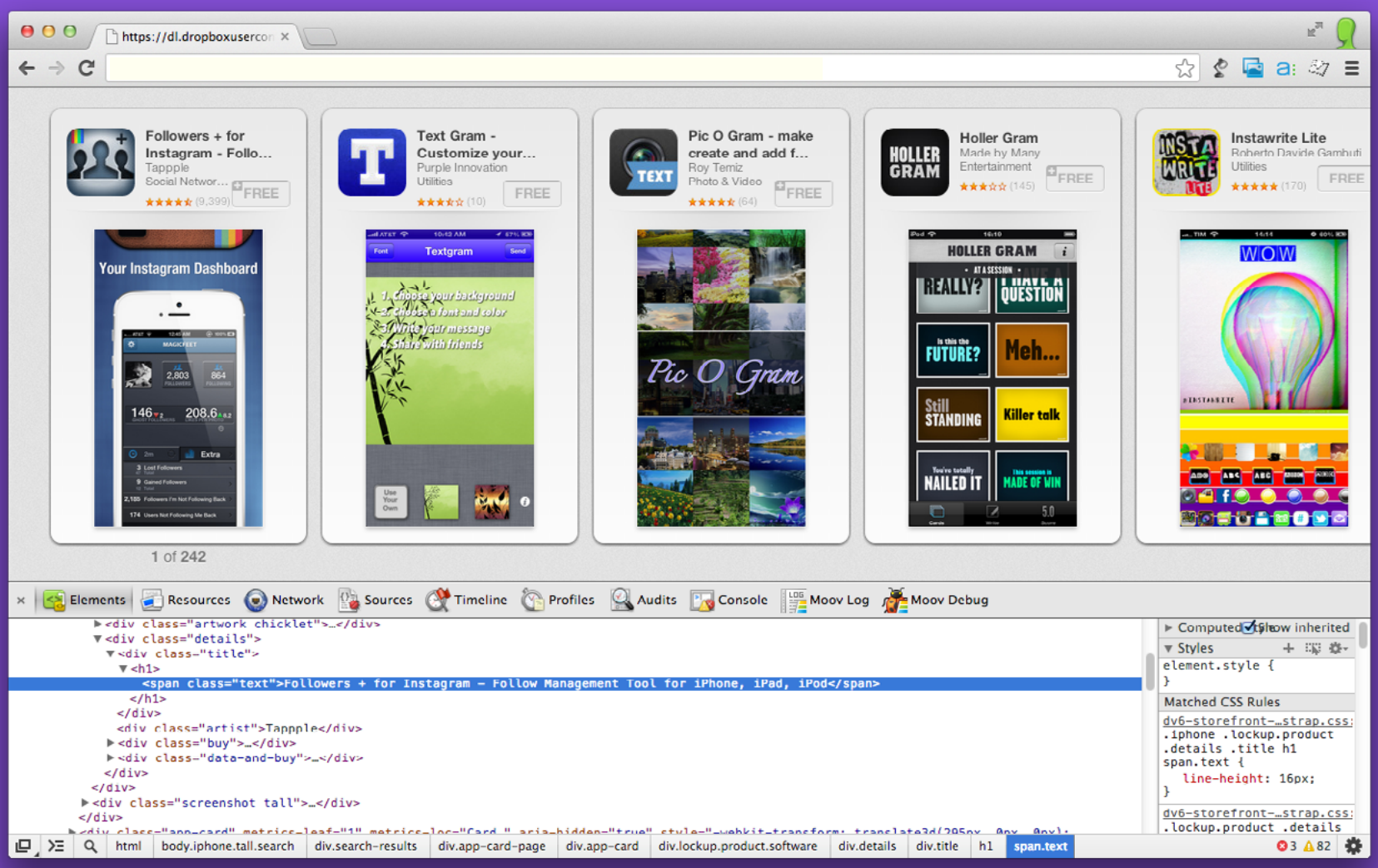
GMail app uses web views and native elements

How the iOS App Store is hybrid

Top search bar and bottom navigation toolbar are native



App descriptions are web view



Like all technologies, the beauty of a hybrid app lies in its implementation. Millions and millions of iPhone and iPad users fire up the Apple App Store to download billions of apps without ever realizing they are looking at a web view. That's right, Apple themselves use web views in the App Store app to display search cards. It just doesn't look like what you might think of when you think of HTML. And that is an important distinction to make with hybrid apps. You aren't simply wrapping a mobile site in native elements. The best hybrid implementations use HTML to deliver content that looks and feels like it belongs in an app, but with all the benefits of a language that web developers know and love.

Hybrid App Patterns

While native and web are pure classifications, hybrid apps encompass a wide spectrum of different architectures. Over the past few years, a few common patterns have emerged across this spectrum for balancing the roles between the native and web layers. While not every app will fit cleanly into a single pattern, they serve as models for understanding the wide range of what's possible.

- **Pure Pattern:** So-called "pure" hybrid apps consist of a single webview that occupies the entire screen. All the content and navigation controls are implemented in HTML5, and a thin native wrapper is used to expose native APIs (like Address Book access) to the HTML5 code. PhoneGap (Cordova) apps use this model. Because of the heavy reliance on HTML5 these apps typically have the most code reuse between mobile platforms and require the least level of native programming experience.

- **Blended Pattern:** Blended apps use native components for the main navigation UI (such as the tab bar) but web views for the formatted content. These apps often use multiple web views with native powered animations between screens. Apple's App Store app and Google's GMail for iOS use this pattern on some screens.

- **Mullet Pattern:** These apps are fully native for early parts of a user flow, such as product browsing, and web based for later areas like checkout. This is a very common pattern for mobile commerce apps where APIs supporting native screens typically exist for product data but not for the hard to integrate features like checkout and payments. The Belk and Walmart apps for iOS both use this pattern.

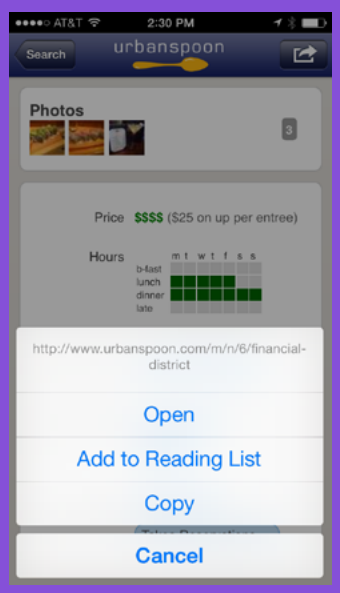
- **Fallback Pattern:** In this pattern, the app is mostly native but uses hybrid web views as a fallback for little used or frequently changing content. Facebook's current app falls under this pattern.

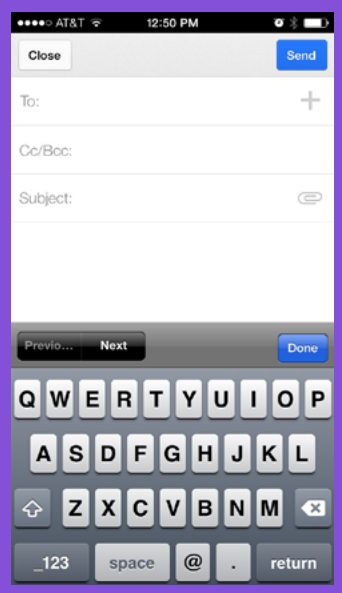
- **API Pattern:** Not all hybrid apps simply render HTML from a server. API driven hybrid apps get their data from the server in JSON or XML and instantiate that data into HTML within the app.

How to spot a hybrid app on iOS

How do you know if you're using a hybrid app? If it's a good hybrid app you shouldn't! But even in the best apps there are often subtle differences that can hint at the use of a web view inside an app.

- Tap highlights and touch callouts are good hints that the content is in a web view. You can trigger a tap callout like the one shown in the Urbanspoon app by using a long press on a link. Developers can avoid and even re-program callouts so while their presence or absence isn't always a 100% indicator of a hybrid app, it's usually a good sign.





- On iOS devices, the virtual keyboard in a hybrid app will have Next and Previous buttons just like Mobile Safari (see screenshot on right).

- The scroll momentum of web views feels different than native views, although developers can change this setting.

- If all else fails, using a network analysis tool like WireShark, Charles Proxy, or Fiddler can help you detect HTML used to power a hybrid app's web view.

Hybrid App Tools

- **PhoneGap** (Apache Cordova) is a free and open source framework that allows you to create mobile apps using standardized web APIs.

- **Trigger.io** is a mobile development platform that bundles a hybrid framework with a JavaScript API for device functionality and UI components with a cloud based build environment.

- **IBM Worklight** is an application development platform that extends PhoneGap (Apache Cordova) with additional development tools and server support for backend integration, authentication, push notifications, and life cycle management. Stackoverflow has a useful guide to understanding the differences between PhoneGap and Worklight.

- **Sencha Space** is a platform for securely distributing HTML5 and hybrid business applications in a managed enterprise environment. Space is an interesting alternative to the traditional mobile-app-management model and offers business control of their data without commandeering an employee's phone.

Hybrid Use Case: Walmart, Belk, and GNC

Walmart, Belk, and GNC are some of the largest retailers in the U.S. and all three companies have taken a hybrid approach to their apps. Hybrid development has enabled them to launch apps across multiple platforms (mobile web, iOS, and Android) without compromising their ability to use native features such as barcode scanning for in-store functionality. By letting them reuse their existing infrastructure investments, the approach also got their apps to market faster and with lower cost.

For both GNC and Belk, the hybrid apps were developed using the Moovweb platform to inherit the existing business logic via responsive delivery and without requiring new APIs or backend integration work. The GNC app uses the blended pattern (see Hybrid App Patterns above) which seamlessly mixes native navigational components with web views for the main content. The HTML content is powered by a Moovweb transform engine (often called a t.engine) that transforms the desktop site into mobile HTML suitable for web views.

By contrast, the Belk app's product browsing screens are entirely native while web views were used for the checkout process. As with the GNC app, a t.engine powers the content for the hybrid web views. But the native screens are supported by a "synthetic API" created by using a t.engine to translate the existing site into JSON.

In both cases, a transform engine also powers the company's mobile web sites unifying the mobile initiatives across sites and apps while leveraging the investment in the existing eCommerce system.

In the case of Walmart, their engineering team also chose hybrid to leverage their existing mobile web development efforts. The checkout experience was something the team found they would be updating frequently and this made it a perfect area to try hybrid. "There was a pragmatic piece to this where the mobile web team was done with the functionality, so we got some good leverage," said Dion Almaer, VP of Mobile Engineering at Walmart.com.

One interesting feature of Walmart's approach is the ability for the app to jump in and render native UI where appropriate. For example, when selecting a delivery option within the app (shipping to a store or to your house), the app shows a native widget rather than the JavaScript version of Google Maps (often thought to be weak in comparison to MapKit). "We use events throughout

IN THE CASE OF WALMART, THEIR ENGINEERING TEAM ALSO CHOSE HYBRID TO LEVERAGE THEIR EXISTING MOBILE WEB DEVELOPMENT EFFORTS.

to make this seamless," Dion explains. "The pick store button can send an event 'getStoreID' and it will be waiting for a 'useStoreID' with the given store number. On the website this is all done in JS, but in the app, native uses the bridge."

CONCLUSION

Selecting the right mobile platform is key for enterprises looking to deliver rich mobile experiences to employees and customers. Native may be the way to go for organizations with plans for high performing apps that require low-latency access to hardware. HTML5 is a great option for companies with limited talent and resources looking to deliver multi-platform apps. Hybrid is the way to go for enterprises that wish to deliver an app across many devices but also take advantage of features that a purely web app cannot give them. The first step is for organizations to evaluate the benefits and drawbacks of each platform to help make the most informed decision possible.

Written By

Ishan Anand
Ishan is director of new products at Moovweb and has been launching mobile products for the iPhone since the day it was released. His apps have been featured on TechCrunch, ReadWriteWeb and LifeHacker. Prior to Moovweb, Ishan worked at Digidesign and his expertise was in multi-threaded real-time systems programming for the computer music industry. Ishan holds dual-degrees in electrical engineering and mathematics from MIT.

Dave Wasmer
Dave is a frontend developer for Kinvey, where he spends his days building complex Backbone single page apps and crafting intuitive user experiences. His nights are spent hacking on web apps, tooling, and more. He also enjoys helping others learn all about JavaScript and Backbone; he has taught at General Assembly and the Startup Institute, as well as mentored Lean Startup Machine workshops, Startup Weekend competitions, and DoD hackathons.

Project Managed By

Kelly Rice
Justin Megahan

Designed By

Jake McKibben

Brought to you by

Kinvey
What is Kinvey? Kinvey makes a fully-featured Backend as a Service solution, offering 3rd party data integrations, multi-platform support, push notifications, and custom business logic on a platform where it's free to get started and you only pay when your app is successful.

Moovweb
Moovweb is a next generation mobile platform for Responsive Delivery, transforming desktop sites in real time for phones, tablets, kiosks, and future endpoints. By leveraging existing web investments, Moovweb unifies web and mobile strategies, dramatically cutting cost and creating business agility.