MOBILE DEVELOPER GUIDANCE

powered by Kendo UI THE WAY OF HTML5

SUMMARY:

If you have unlimited time and money, build native apps for all mobile platforms. If you have anything less, important decisions must be made that can have a significant impact on the cost, capability, and reach of your mobile apps.

This document presents key considerations and provides focused guidance to help ensure any mobile strategy is well with the goals of your business.

PREPARED BY:

- Todd Anglin, VP HTML5 Web & Mobile Tools, Telerik
- Chris Sells, VP Developer Tools, Telerik
- Doug Seven, EVP Cloud Tools, Telerik
- Stephen Forte, Chief Strategy Officer, Telerik

CONTENTS

DEVELOPMENT DECISIONS

There are two core decisions that must be made when beginning any mobile development project:

- Which platforms will be supported?
- How will the app be developed to support those platforms?

The answers to these fundamental questions will have the biggest impact on the cost and skills required to complete a mobile development project.

Cross-Platform or Single-Platform

Unlike desktop software, where for years a single platform (Windows) has dominated the landscape, mobile platforms are far more fragmented. This requires that developers consider the platforms they aim to support before beginning any mobile development project. The decision to support multiple mobile platforms or only a single mobile platform can have a significant impact on development decisions.

Single-Platform

Single-platform development is most appropriate for internal-facing app development, where devices are controlled, or for businesses that have made a strategic decision to focus on a single mobile audience. If an organization has standardized on a single mobile platform or strategically decide to focus on one customer platform, such as iOS, app development can be streamlined thanks to reduced complexity designing, building, and testing apps.

While single platform development eliminates the complexity of cross-platform development decisions, it does limit the reach of a mobile app and creates platform lock-in, increasing risk if mobile platform preferences change in the future. The reduced flexibility can also make it challenging for organizations aiming to adopt Bring Your Own Device (BYOD) policies.

GUIDANCE: *Single-platform development should only be pursued by organizations with highly controlled users on a standardized mobile platform.*

Cross-Platform

Cross-platform development extends the reach of mobile apps by deploying to two or more mobile platforms, such as iOS and Android. There are many different approaches to cross-platform development, each with different advantages that will be discussed later, but the common trait amongst all approaches is the desire to develop the same (or very similar) apps for multiple mobile platforms.

Supporting multiple platforms does add complexity to all stages of the development process, making it a longer and typically more expensive route than single-platform development. The gain, of course, is the ability to reach a substantially larger pool of users while maintaining the flexibility to change course as mobile platforms continue to evolve.

In today's mobile market, no single mobile platform enables developers to reach a majority of mobile users. At a minimum, according to an August 2012 IDC report , supporting iOS and Android unlocks approximately 85% of the market, while additionally supporting BlackBerry and Windows Phone delivers an additional 8%. Intelligent cross-platform development balances the cost of supporting multiple platforms with the added benefit of reaching more users.

GUIDANCE: *Cross-platform development should be the preferred approach for any organization supporting a BYOD policy or that is developing apps for the general mobile market.*

BROWSER, HYBRID, OR NATIVE

Whether building single or cross-platform mobile apps, there are three primary development techniques that can be used to build mobile apps:

1. Browser
2. Hybrid, and
3. Native

Each technique delivers different benefits, and picking the right technique requires careful analysis of an app's requirements and goals.

Browser

Browser app development leverages the power of HTML, CSS, and JavaScript to build and deliver apps via modern mobile browsers. Modern mobile browsers include Apple's Mobile Safari (found on iOS devices), Google's Chrome for Android (available on Android 4+), and Microsoft's Internet Explorer 10 (found on Windows Phone 8).

While the technique is commonly referred to as "Browser" (or sometimes "Mobile Web Apps" or "HTML5 Apps"), apps built in this way do not necessarily require an active Internet connection to function (after the initial download). With a capable HTML and JavaScript framework, Browser apps can be made to look and behave very similarly to Native apps.

Mobile Browser apps are the fastest way to build apps that can reach multiple mobile platforms. With no installs or special packaging required, any mobile device with a capable mobile browser can load and use an app built and deployed in this way. Developers can choose to optimize their app for specific platforms, presenting different views depending on the mobile OS and device, or they can build a "one size fits all" UI that is common for all devices. In either case, the majority of the app's JavaScript and HTML resources will be shared for all platforms.

The biggest drawback of Browser apps is that they cannot access the full range of device sensors and APIs, and they cannot be deployed via the popular mobile app stores.

PROS

- Familiar development languages (JavaScript, HTML, CSS) and relative abundance of software developers with these skills
- Simple, browser-based deployment
- Easiest way to deploy an app to multiple platforms, reach most users ("automatically" available on any device with a capable browser)
- Built on industry standards, no platform lock-in

CONS

- Limited access to device APIs and sensors (bound by browser security)
- Cannot be distributed via mobile app stores
- Few mobile platforms support direct installation of mobile Browser apps (such as the iOS "Add to Home Screen" option in Safari)
- App developer must monetize app independently (no app store revenue)

BEST FOR:

- Organizations seeking the least expensive, fastest, and most flexible path for building and deploying cross-platform mobile apps
- Apps that do not want to be distributed via App Stores (commonly to avoid the “app store tax” or app store rules)
- Manage, Inform, Shop, Search, and some Connect mobile app modes (as defined by Yahoo’s Seven Mobile Modes whitepaper)

Hybrid

Hybrid apps aim to combine the convenience of developing with HTML, JavaScript, and CSS with the power of native apps. With this technique, mobile apps are developed using the familiar technologies of the web, identical to Browser apps, but the resulting application is then packaged in a native app “shell” that extends the power of the Browser app. The native shell acts as a proxy that allows JavaScript to access a wide range of device APIs and sensors not normally available in a browser, and the shell also allows apps to be distributed through mobile app stores.

The Apache Foundation’s Cordova – also know by Adobe brand name, PhoneGap – are the most popular open source hybrid app containers. When working with these containers, developers can realize the productivity of Browser app development while maintaining access to device APIs and sensors, like the camera, compass, accelerometer, and more.

Unlike Browser apps, though, which are easily deployed through the browser, Hybrid apps must be built for each target platform and installed on a device. The app’s code can be the same for each platform, but a separate configuration and build of the app is required for each supported platform. New cloud-based tools, like Icenium and PhoneGap Build, help offset this challenge by managing the build and deployment of hybrid apps to different platforms.

Done well and with the right tools, a Hybrid app can be indistinguishable from a native app to end-users.

PROS

- Familiar development languages (HTML, JavaScript, CSS) and availability of software developers with these skills.
- Access to many device APIs and sensors (beyond normal limits of JavaScript)

- Installable apps that can be app store deployed, monetized
- Common code base for cross-platform apps

CONS

- Separate configuration and build required for each target platform (requiring native tools or the assistance of new cloud build tools)
- More difficult to package, debug, and deploy than Browser (install required)
- Bound by app store rules, less freedom than Browser

BEST FOR:

- Organizations building cross-platform apps seeking the benefit of Browser app development without the associated deployment and device limits
- Apps that want to be deployed via mobile app stores
- All seven Mobile Mode types defined by Yahoo!, except for graphically intense games or entertainment experiences

Native

Native app development is perhaps the most familiar mobile app development technique. Native app development refers to the technique that leverages platform-specific SDKs and languages to build an app. For example, Native apps on iOS use Apple's APIs, Objective C, and UIKit. On Android, apps use Google's APIs, Java, and Android's specific XML UI markup. On Windows Phone, it's Microsoft's APIs, .NET, and XAML.

NOTE: Browser and Hybrid apps also run "natively" on devices. While the term "native app" refers to native SDK apps, it's important to note that Browser and Hybrid apps also run on native mobile runtimes, not on plug-ins.

Native app development clearly has the most access to underlying device performance and capabilities, but it is also the least flexible and most expensive approach to cross-platform mobile development. Building for multiple platforms with this approach requires developer teams to build the same app multiple times, in multiple languages, in multiple development environments.

This problem is exacerbated as more platforms are targeted. For each platform that must be supported, organizations must spend similar time, effort, and money to produce and test an app. This applies to app maintenance, too.

Cross-compiling tools offer an intermediate solution by allowing apps to be written in one language, such as .NET, and then compiled in to a native app. Cross-compiling solutions help reduce the number of languages developers must master to build cross-platform native apps, but they often don't reduce the need to build separate apps for separate platforms. Cross-compiling also introduces a layer of abstraction between source code and runtime that must be considered when debugging and optimizing.

PROS

- Unrestricted access to underlying platform APIs and device capabilities
- Maximum application performance (due to vendor optimized device APIs)
- Fewest limits on app design and behavior (only limited by platform, device)

CONS

- Most expensive, least flexible technique for cross-platform development
- Many languages and tools required for each platform
- Smaller pool of available developers with necessary skills for each platform
- No-to-low development reuse between platforms

BEST FOR

- Organizations committed to a single-platform mobile app strategy or cases where an app's requirements exceed the capabilities of Hybrid or Browser app development
- Apps with unique, richly animated interfaces (pushing platform boundaries)
- Entertainment apps, especially graphically rich games
- Millisecond or frame rate optimized apps

BOTTOM LINE GUIDANCE

Every app and every organization will have different priorities that guide the decisions made for app development, but there are several key drivers for most mobile development projects:

- FLEXIBILITY: The ability to easily adapt an app to new and changing mobile platforms
- TIME TO MARKET: The time it takes to build and deliver an app to mobile users
- COST: The cost of building a mobile app, in terms of development hours, training, and maintenance
- PERFORMANCE: The perceived speed of a mobile app

Assuming the goals of an organization are to deliver an app that minimizes cost and time to market while maximizing flexibility and reach, the recommended development technique is Browser or Hybrid (depending the preferred deployment method and need for access to device APIs).

Native development is only advisable for apps that demand maximum device performance and for organizations that can afford the investment in development and maintenance teams skilled in each mobile platform.

Mobile Form Factors

In addition to platform decisions, mobile app developers must also consider the range of mobile form factors that will be supported. The most common mobile form factors today are tablets and smart phones, and apps created for these form factors are usually designed to scale to different physical device dimensions. In other words, separate apps are not created for 10-inch and 7-inch tablets, or for phones ranging in size from 3- to 5-inches.

There are two basic development approaches for supporting tablet and smart phone form factors:

1. ONE SIZE FITS ALL

Many tablets today are capable of running smart phone apps built for the same platform, so developers can choose to build one app for both phones and tablets. With this approach, developers typically design for the phone and allow tablet users to simply experience an "enlarged" version of the same app.

Alternatively, developers can use responsive design techniques to build an app that will rescale for different form factors, but generally provide the same experience regardless of device.

BEST FOR: Organizations that want to minimize cost or that are only interested in emphasizing a single form factor

2. TAILORED EXPERIENCE

The other option, of course, is to design separate apps for phone and tablet form factors, optimizing the experience for the UX conventions of each device. This typically provides a much better experience for tablet users, but it does add time to all phases of the app development process (design, construction, and testing).

Creating separate designs for different form factors does not necessarily mean creating separate apps. Techniques exist for building and deploying a single app that contains both the tablet and smart phone views.

BEST FOR: Organizations that want to emphasize phone and tablet experiences.
Supporting multiple form factors should also be considered when choosing a development technique. For example, Browser and Hybrid apps constructed with the right frameworks can be built and deployed to support multiple platforms and multiple form factors from a single codebase. This adaptive design further simplifies the work involved in supporting a wide variety of users. Native app development, meanwhile, typically requires more code and configuration to support multiple form factors.

MOBILE WEB SITES VS. MOBILE BROWSER APPS

It is important to understand the difference between developing mobile apps and mobile sites, particularly when considering Browser development. While both leverage HTML, JavaScript, and CSS and both deploy through the mobile browser, the experiences and development techniques are very different.

Web Sites

Mobile web sites are adapted presentations of web content, optimized for the smaller screens typical to phones and tablets. Web sites use responsive design to dynamically adjust the layout and styling of content so that it is comfortable to consume on any form factor. Mobile web sites do not focus on providing an “app navigation” experience, but rather rely on traditional browser navigation concepts (URLs, back button, links) and depend on an active Internet connection to work (“Always Connected”).

WHEN TO BUILD A MOBILE WEB SITE:

- Targeting mobile users with content normally presented on a website
- Informational, read-only experiences
- Always connected, server-side mobile apps

Browser Apps

Mobile Browser apps, on the other hand, are designed to provide a native app experience. They use the same underlying technologies and delivery vehicle as mobile web sites, but the user experience is very different. Browser apps deemphasize traditional web navigation concepts and provide separate, in-app navigation. They are also capable of working without an active Internet connection by leveraging modern application caching techniques. Apps are also more focused on specific tasks than informational mobile web sites.

WHEN TO BUILD A MOBILE BROWSER APP:

- Goal is to build an "app" for end-users instead of a "website"
- Interactive, task-oriented mobile experience (see Yahoo's Mobile Modes)
- Sometimes connected, browser deployed apps

Research also shows that users are far more engaged with apps on mobile devices than websites. According to ComScore's 2012 Mobile Matrix 2.0, 4 out of every 5 mobile media minutes is spent using apps instead of websites, despite the fact that apps and websites reach roughly the same audience. As an example, the report found that 80% of mobile time spent with Facebook was done via the app, compared to just 20% done via the browser. This behavioral trend should also be considered when choosing between Browser App and Mobile Site development.

COMMON SCENARIOS

Prescriptive guidance for specific mobile development scenarios

SCENARIO 1: "I want to build an app for my business that supports iOS and Android."
GUIDANCE: Build a Browser or Hybrid app using HTML5, JavaScript, and CSS. If support for Windows Phone 7 (WP7) is also required, a native WP7 app created with .NET and XAML should also be built since WP7 lacks a capable HTML5 runtime for delivering quality mobile experiences. Windows Phone 8 with IE10, however, can be targeted with a cross-platform HTML5 mobile app.
SCENARIO 2: "I want to build an app for Windows 8 tablets."
GUIDANCE: Build a Native app using .NET and XAML. Windows 8 and Windows Phone offer an experience that is very different from the iOS, Android, and BlackBerry platforms, so targeted, native apps are the best solution. Using .NET and XAML also allows the app to be adapted for Windows Phone 8, a platform that does not support WinJS and HTML Windows Store applications.

SCENRARIO 3: "I want to build one app that targets all mobile platforms."
GUIDANCE: Build a Hybrid app using HTML5, JavaScript, and CSS. The Hybrid container will allow the app to be deployed via app stores and access native device APIs, while the familiar web technologies will maximize code reuse across all major mobile platforms, including Windows 8. Over time, this approach should be increasingly capable as devices and web standards evolve.

SCENARIO 4: "I want to build a rich, graphical mobile game."
GUIDANCE: Build a Native app using the native SDKs for your target platforms. For apps where milliseconds matter, there is no substitute for low-level, platform-specific development to deliver a smooth, responsive experience.

FUTURE DISRUPTORS

In addition to making a mobile development decisions that are correct for today's mobile landscape, it is important to understand the disruptors that hold the ability to reshape the mobile landscape over the next two to three years. Carefully considering these trends today further ensures a successful long-term mobile strategy.

Windows 8 and Windows Phone 8

While Microsoft continues to lag far behind Apple and Google for a share of the mobile landscape today, their next attempt at changing the balance arrived this year in the form of Windows 8. With this release, Microsoft hopes to challenge Apple's iPad dominance and reclaim a seat at the mobile table. Companies focusing on iOS and Android today should be prepared to address Microsoft's new platform if it is successful at earning a piece of the pie, especially for tablets.

Chrome OS, Firefox OS, and Tizen

Three new operating systems are likely to challenge for a share of the mobile pie beginning in 2013 in the form of Google's Chrome OS, Mozilla's new Firefox OS (formerly "Boot-to-Gecko"), and Tizen, a mobile OS collaboration between Intel, Samsung, and the Linux Foundation. All three operating systems put HTML5 at the center of their DNA, requiring all apps to be built with HTML, JavaScript, and CSS. Companies that want to be prepared to support these platforms should consider Browser or Hybrid development today.

7-inch Tablets

The prevailing tablet form factor today is the 10-inch tablet, thanks largely to Apple's dominance with the iPad. New 7-inch tablets introduced by Amazon, Google and Apple, though, are gaining traction. Developers are un-impacted today, as most deploy the same app design to 10-inch and 7-inch tablets, but if new UI guidelines are introduced for 7-inch tablets, developers may be required to start designing for three distinct form factors when targeting native experiences.

Platform Proliferation

According to 2011 research by Gartner, CIOs expect to support an average of 3.5 smartphone and tablet platforms by the end of 2012, with that number growing in subsequent years. Regardless of the exact number, it is clear that organizations- even those capable of controlling devices- are embracing a multi-platform world. The addition of new platforms like Windows 8, Chrome OS, Firefox OS, and Tizen will only extend the problem. Companies skeptical of the need to support multiple mobile platforms should consider this trend before embarking on single platform development.

ABOUT KENDO UI

Kendo UI is a complete solution for JavaScript and HTML5 developers that provides web and mobile developers with all the necessary components for building HTML5 and JavaScript mobile apps and sites. Based on jQuery, Kendo UI delivers a rich UI for the web, HTML5-powered data visualizations, and tools for building native mobile apps with HTML5. This leading edge framework delivers everything in a unified, compact package, backed by industry leading professional support.

ABOUT TELERIK

Telerik is the market-leading provider of end-to-end solutions for application development, automated software testing, agile project management, reporting, and content management across all major Microsoft development platforms. Telerik is trusted by more than 100,000 customers worldwide for its innovation and industry best technical support.

Report Inquiries

Todd Anglin
Vice President, HTML5 Web & Mobile Tools
Telerik
phone: 888-365-2779
Todd.Anglin@kendoui.com

Sasha Krsmanovic
Director of Marketing
Telerik
phone: 888-365-2779
Sasha.Krsmanovic@telerik.com

Media Inquiries

Erica Burns
Manager, Media & Customer Relations
Telerik
phone: 888-365-2779 ext. 154
Erica.Burns@telerik.com

Carmen Hughes
Ignite PR
phone: 650-227-3280 ext. 101
c: 650-576-6444
Carmen@ignitepr.com

Mobile Modes Whitepaper, Yahoo!
http://advertising.yahoo.com/article/mobile-modes.html
Mobile Matrix 2.0, ComScore
http://www.comscore.com/Press_Events/Press_Releases/2012/5/Introducing_Mobile_Metrix_2_Insight_into_Mobile_Behavior