## Native Apps

**Native apps** live on the device and are accessed through [icons](https://www.nngroup.com/articles/classifying-icons/) on the device home screen. Native apps are installed through an application store (such as Google Play or Apple’s App Store). They are developed specifically for one platform, and can take full advantage of all the device features — they can use the camera, the GPS, the accelerometer, the compass, the list of contacts, and so on. They can also incorporate [gestures](http://www.jnd.org/dn.mss/gestural_interfaces.html) (either standard operating-system gestures or new, app-defined gestures). And native apps can use the device’s [notification](https://www.nngroup.com/articles/indicators-validations-notifications/) system and can work offline.

**A native app** is a smartphone application developed specifically for a mobile operating system (think Objective-C or Swift for iOS vs. Java for Android). Native **applications** are developed for a specific platform and installed on a computing device

Building native applications means using the native language of the platform, Objective-C on iOS, and Java on Android. The main advantage of native applications is their **performance**. Native apps are compiled into machine code (Dalvik byte code under Android), which gives the best performance you can get from the mobile phone. Best performance includes fast and fluid animations as well as full access to phone hardware, multi touch support and the latest APIs.

Since the app is developed within a mature ecosystem following the technical and user experience guidelines of the OS (e.g. swipes, app defined gestures, left aligned header on Android, centrally aligned header on iOS, etcetera), it not only has the advantage of faster performance but also “feels right”. What feeling right means is that the in-app interaction has a look and feel consistent with most of the other native apps on the device. The end user is thus more likely to learn how to navigate and use the app faster. Finally, native applications have the significant advantage of being able to easily access and utilize the built-in capabilities of the user’s device (e.g., GPS, address book, camera, etcetera). When a user sends text messages, takes pictures using the device’s default app, set reminders, or uses the device’s music app (the one that came with the phone), they’re using native apps. In short, native apps are exactly that, native to the user’s OS and hence built per those guidelines.

Native development is far from easy. Despite the great number of resources that can be found, it may not be understandable to everyone. As code must be written specifically for each platform, the same code will have to largely be rewritten with little able to be shared. The logic may be the same, but the language, APIs and the development process is different. This process can be relatively long for complex applications.

**Native applications have the best performance, highest security, and best user experience.**

### Advantages of native apps

* + They offer the fastest, most reliable and most responsive experience to users. This is unlikely to change in favor of web apps.
  + It is easier to tap into the wider functionality of the device including the camera, microphone, compass, accelerometer and swipe gestures. It’s still possible using the alternatives, but it’s easiest on native.
  + Native apps can make use of push-notifications, alerting users when their attention is required in the app. You get the opportunity to continually bring your audience back for more which is key to a successful app.
  + You’re more likely to please your users due to the way you can match each app’s UI/UX to the platform conventions. There are dozens of UI/UX differences that make users feel at home. By building native, you don’t have to compromise with UI/UX that you hope will be user-friendly for all platforms.

### Disadvantages of native mobile apps

* + You’ll have to manage a codebase for each platform you launch on
  + iOS apps will not run on Android and vice versa
  + Most developers specialize in one platform ( Android or iOS), so to create an app on both platforms will require two separate developers (or teams)
  + Native apps generally cost more to make than hybrid apps

### ****Examples of Native Apps****

* + Most of the most popular apps out there like Pokemon Go, Twitter, and Waze, are fully native. It’s become trickier though to distinguish who’s using purely native code on Swift, Objective C and Java and who’s relying on hybrid solutions or cross-platform SDKs.

## Mobile Web Apps

**Web apps** are not real applications; they are really **websites** that, in many ways, look and feel like native applications, but are not implemented as such. They are run by a browser and typically written in HTML5. Users first access them as they would access any web page: they navigate to a special URL and then have the option of “installing” them on their home screen by creating a bookmark to that page.

Web apps became really popular when HTML5 came around and people realized that they can obtain native-like functionality in the browser. Today, as more and more sites use HTML5, the distinction between web apps and regular web pages has become blurry.

According to Wikipedia, a web app “is an application that is accessed via a web browser over a network such as the Internet.” So how is this different than a web site? . For example, Wikipedia is a website; it provides information. Facebook is a web app.

Don’t let the word “app” confuse you though. Web apps don’t need to be downloaded like mobile apps do. Web apps load in browsers like Chrome, Safari, or Firefox and they don’t take up any memory or storage on the user’s device.

How are they built? The vast majority are built in JavaScript, CSS, and HTML5. Unlike a mobile iOS or Android app, there is no software development kit (SDK) for a developer to work with. There are templates and frameworks like Angular, React, and Vue.js you can use to get a quick start. As opposed to mobile apps, developing a web app can be simple and quick, however, their simplicity is also their downside. It’s often a good way to test out an idea before investing in a mobile app.

## Hybrid apps

A hybrid application (hybrid app) is one that combines elements of both native and Web applications. Native applications are developed for a specific platform and installed on a computing device. Web applications are generalized for multiple platforms and not installed locally but made available over the Internet through a browser. Hybrid apps are often mentioned in the context of mobile computing.

Hybrid mobile apps are like any other apps you’ll find on your phone. They install on your device. You can find them in app stores. With them, you can play games, engage your friends through social media, take photos, track your health, and much more.

Like the websites on the internet, hybrid mobile apps are built with a combination of web technologies like HTML, CSS, and JavaScript. The key difference is that hybrid apps are hosted inside a native application that utilizes a mobile platform’s [WebView](http://developer.telerik.com/featured/what-is-a-webview/). (You can think of the WebView as a chromeless browser window that’s typically configured to run fullscreen.) This enables them to access device capabilities such as the accelerometer, camera, contacts, and more. These are capabilities that are often restricted to access from inside mobile browsers. Furthermore, hybrid mobile apps can include native UI elements in situations where necessary, as evidenced by [Basecamp’s approach towards hybrid mobile app development](https://signalvnoise.com/posts/3743-hybrid-sweet-spot-native-navigation-web-content).

**Hybrid apps** are part native apps, part web apps. (Because of that, many people incorrectly call them “web apps”). Like native apps, they live in an app store and can take advantage of the many device features available. Like web apps, they rely on HTML being rendered in a browser, with the caveat that the browser is embedded within the app.

Often, companies build hybrid apps as wrappers for an existing web page; in that way, they hope to get a presence in the app store, without spending significant effort for developing a different app. Hybrid apps are also popular because they allow crossplatform development and thus significantly reduce development costs: that is, the same HTML code components can be reused on different mobile operating systems. Tools such as PhoneGap and Sencha Touch allow people to design and code across platforms, using the power of HTML.

**Hybrid applications** are, at core, websites packaged into a native wrapper. They look and feel like a native app, but ultimately outside of the basic frame of the application (typically restricted to the controls/navigational elements) they are fueled by a company’s website. Basically, a hybrid app is a web app built using HTML5 and JavaScript, wrapped in a native container which loads most of the information on the page as the user navigates through the application (Native apps instead download most of the content when the user first installs the app). Usual suspects here are Facebook, Twitter, Instagram, your mobile banking app, etcetera.

Cross platform development is certainly very popular these days. By creating one source code which fuels the app on the two dominant platforms (iOS and Android) a company saves time and money to get an app out the door. Additionally, depending on the complexity of the app, maintaining one source code is very appealing because new features are also easier to develop and deploy. Conversely, bug fixes are platform agnostic and can be done and released easily to production. Lastly, hybrid apps are distributed through the app store like all other apps which means the user will not see any immediate difference between a hybrid and a native app.

Hybrid applications are web applications (or web pages) in the native browser, such as UIWebView in iOS and WebView in Android (not Safari or Chrome). Hybrid apps are developed using HTML, CSS and Javascript, and then wrapped in a native application using platforms like [Cordova](http://cordova.apache.org/). This allows you to use any web-native framework you want, and there are plenty of these.

The application development is faster, **simpler**, more **rapid** and the application is easier to maintain. You can change platforms anytime you need, Cordova lets you build your application for more than one platform just by one adding line of code. As for the phone hardware such as the camera or Bluetooth, Cordova has a large repository of [plugins](http://plugins.cordova.io/" \l "/) you may use.

The main problem with hybrid apps is that they still depend on the native browser, which means they are not as fast as native apps.



## Native, Web App, or Hybrid: Which Should You Choose?

Each of these types of apps has their advantages and disadvantages,

**Device features.** Although web apps can take advantage of some features, native apps (and the native components of the hybrid apps) have access to the full paraphernalia of device-specific features, including GPS, camera, gestures, and notifications.

**Offline functioning.** A native app is best if your app must work when there is no connectivity. In-browser caching is available in HTML5, but it’s still more limited than what you can get when you go native.

**Discoverability**. Web apps win the prize on discoverability. Content is a lot more discoverable on the web than in an app: When people have a question or an **information need**, they go to a search engine, type in their query, and choose a page from the search results. They do not go to the app store, search for an app, download it, and then try to find their answer within the app. Although there are app aficionados who may fish for apps in app stores, most users don’t like installing and maintaining apps (and also wasting space on their device), and will install an app only if they expect to use it often.

**Speed.** Native apps win the speed competition. In 2012 Mark Zuckerberg declared that Facebook’s biggest mistake had been betting on the mobile web and not going native. Up to that point, the Facebook app had been a hybrid app with an HTML core; in 2012 it was replaced with a truly native app. [Responsiveness is key to usability](http://www.nngroup.com/articles/response-times-3-important-limits/).

**Installation**. Installing a native or hybrid app is a hassle for users: They need to be really motivated to justify the [interaction cost](https://www.nngroup.com/articles/interaction-cost-definition/). “Installing” a web app involves creating a bookmark on the home screen; this process, while arguably simpler than downloading a new app from an app store, is less familiar to users, as people don’t use bookmarks that much on mobile.

**Maintenance.** Maintaining a native app can be complicated not only for users, but also for developers (especially if they have to deal with multiple versions of the same information on different platforms): Changes have to be packaged in a new version and placed in the app store. On the other hand, maintaining a web app or a hybrid app is as simple as maintaining a web page, and it can be done as often or as frequently as needed.

**Platform independence.** While different browsers may support different versions of HTML5, if platform independence is important, you definitely have a better chance of achieving it with web apps and hybrid apps than with native apps. As discussed before, at least parts of the code can be reused when creating hybrid or web apps.

**Content restrictions, approval process, and fees.** Dealing with a third party that imposes rules on your content and design can be taxing both in terms of time and money. Native and hybrid apps must pass approval processes and content restrictions imposed by app stores, whereas the web is free for all. Not surprisingly, the first web apps came from publications such as Playboy, who wanted to escape Apple’s prudish content censure. And buying a subscription within an iOS app means that 30% of that subscription cost goes to Apple, a big dent in the publishers’ budget.

**Development cost.** It’s arguably cheaper to develop hybrid and web apps, as these require skills that build up on previous experience with the web. [NN/g clients](http://www.nngroup.com/about-client-list/) often find that going fully native is a lot more expensive, as it requires more specialized talent. But, on the other hand, HTML5 is fairly new, and good knowledge of it, as well as a good understanding of developing for the mobile web and hybrid apps are also fairly advanced skills.

**User Interface.** Last but not least, if one of your priorities is providing a user experience that is consistent with the operating system and with the majority of the other apps available on that platform, then native apps are the way to go. That doesn’t mean that you cannot provide a good [mobile user experience](https://www.nngroup.com/articles/mobile-usability-update/) with a web app or a hybrid app — it just means that the graphics and the visuals will not be exactly the same as those with which users may be already accustomed, and that it will be harder to [take advantage of the mobile strengths and mitigate the mobile limitations](https://www.nngroup.com/articles/mobile-ux/).

To summarize, native apps, hybrid apps, or web apps are all ways to cater to the needs of the mobile user. There is no unique best solution: each of these has their strengths and weaknesses. The choice of one versus the other depends on each company’s unique needs.

It can be very difficult to tell how a mobile application is built. Hybrid mobile applications are no different. A well-written hybrid app shouldn’t look or behave any differently than its native equivalent. More importantly, users don’t care either way. They simply want an application that works well.

### ****Advantages of Hybrid Apps****

All the advantages of hybrid apps stem from the fact that instead of building two apps, you’re building one app and tweaking it a bit so it works on both platforms. Now you only have one codebase to manage. This will probably require half the number of developers two native apps would have required. Or, with the same number of developers, a hybrid app could be published in half the time.

* Developers for hybrid apps are often less expensive than native developers
* Hybrid apps are easier to scale to another platform. Once you’ve built for one platform, you can launch on another like Windows Mobile.
* You retain the same ability to access device features as with native apps, thanks to solutions like Phonegap that act like a bridge between the native SDK and the webview in which the app runs
* Bottom line: Hybrid apps save you time and money

### ****Disadvantages of Hybrid Apps****

* Performance is probably the biggest disadvantage of hybrid apps. Because hybrid apps load in a browser-like component called webview, they are only as good as the webview. Webview is responsible for displaying the UI and for running Javascript code. In the early days of mobile, Google and Apple didn’t give webview the same engines used by their mobile browsers, Chrome and Safari. Since then, webview has vastly improved but it hasn’t reached native performance yet. We will get deeper into different aspects of performance later in the article.
* Getting your hybrid app to run appropriately on each platform generally takes substantial work. In some situations, the total cost might become comparable to that of fully native apps, rendering the cost benefits negligible. It all depends on how close you want to get to the “native user experience” or how simple your app is.
* The UX of the app will suffer. iOS and Android users tend to be very loyal to their platforms, and since they’ve been using them for years, they’re used to how things work in native apps. The differences are subtle but can be frustrating for your users. By building a hybrid app, you won’t be able to please both camps. Try too hard to customize the app based on the platform and it may end up costing the same as two native apps. There are some ways you can do this which we will discuss shortly.
* **Hybrid application features:**
  + Can function whether or not the device is connected.
  + Integration with a device’s file system.
  + Integration  with Web-based services.
  + An embedded browser to improve access to dynamic online content.
  + Single code base across multiple platforms.
  + Don’t have to update each app in the app store to wait for approvals.
  + You can use your existing web talent and don’t need to bring on additional resources.
  + Don’t need to do any API development since it’s all handled via the web.
* **Examples of hybrid applications:**

Most applications could be considered hybrid apps. Web apps, such as online banking services, typically store some content locally; locally stored native apps, such as Microsoft Word, also interface to the Internet.

## ****Native Vs Hybrid apps – A quick overview****

A native app is faster and more reliable by its very design. As users navigate a native mobile app, the contents, structure, and visual elements are already on their phone, available for instant loading, and thereby providing a seamless experience. This is akin to downloading most of a website’s static content to a user’s phone at once which is then available for instant loading regardless of their phone’s internet speed.

In contrast, a hybrid app has only a wrapper that is downloaded to the user’s phone (which may or may not contain all the navigational elements) with most of the data being loaded from the server. In this case, there are two key issues that may have an impact on the overall performance of the app: the number of server requests (i.e., how many people are making calls to the same server at the same time), and the load balance requests (are the requests coming from mobile devices pinging the same servers as desktop/laptop clients, or do they have designated servers).



Experts agree that, despite all efforts, hybrid applications take a hit in the performance war.

**With a hybrid application**, unless a company adds a completely new feature that dramatically changes the user experience, **the user doesn’t need to update the app in the app store. If the update in question is on a page that is loaded from the server**, as the user navigates through your app they will instantly see the update. It’s that simple.

In contrast, **for native applications the user needs to update the app to see the changes.** For most users who set up auto-updates when their phones are on wi-fi this is acceptable, but it doesn’t work for everyone. Nobody wants to exasperate their user by having him/her update the app every month or so. It attracts unnecessary negative attention to the app which could cause the user to simply uninstall it.

* **How are hybrid mobile apps built?**

Hybrid mobile applications are built in a similar manner as websites. Both use a combination of technologies like HTML, CSS, and JavaScript. However, instead of targeting a mobile browser, hybrid applications target a WebView hosted inside a native container. This enables them to do things like access hardware capabilities of the mobile device.

Today, most hybrid mobile applications leverage [Apache Cordova](http://cordova.apache.org/), a platform that provides a consistent set of JavaScript APIs to access device capabilities through plug-ins, which are built with native code. As a side note, Apache Cordova originally started as a project named [PhoneGap](http://phonegap.com/). These days, PhoneGap exists as a distribution of Apache Cordova that includes additional tools

These plug-ins include APIs for accessing the device’s accelerometer, contacts, camera, and more. There is also a number of plug-ins that are built and maintained by the developer community at-large. These can be found in the [Apache Cordova Plugins Registry](http://plugins.cordova.io/). A curated subset of these plug-ins that have been throughly tested, documented, and extended can also be found at the [Telerik Verified Plugins Marketplace](http://plugins.telerik.com/).

Application assets like HTML, CSS, JavaScript are packaged through the tooling made available through Apache Cordova to target platform SDKs. Once built, you have an application that can run like any other kind of application on the device. The tooling provided by Apache Cordova is largely driven through a command line interface. That stated, developers can still leverage IDEs like Visual Studio and solutions like the [Telerik Platform](http://www.telerik.com/platform) to further enhance productivity.

* **What are the motivations to go hybrid?**

Hybrid mobile applications provide a way for developers to re-use their existing skills in web development. Developers don’t like the prospect of getting locked into proprietary platforms. This includes the programming languages and SDKs provided by platform vendors (more on this later).

Hybrid mobile application development looks appealing to an organization’s bottom line. Why hire a developer for each platform when you can hire one developer and target all of them through HTML, CSS, and JavaScript? Well, the reality is a bit more complicated.

The hybrid mobile application development enables developers to target more than one platform. However, each platform comes with a set of caveats when it comes to its web runtime or WebView. This is especially true with Android, which is inconsistent between OS versions.

Moreover, there might be unique capabilities of platforms to which a developer may wish to target. In those instances, a combination of plugins and platform-specific code must be utilized in order to take advantages of those capabilities. Alternatively, developers can take advantage of 3rd party web runtimes like [Crosswalk](https://crosswalk-project.org/) that can be embedded in your hybrid app.

It should be noted that Android 5.0 introduced updatable WebViews via the [Android System WebView](https://play.google.com/store/apps/details?id=com.google.android.webview). Check out [What Android 5.0’s Auto-Updating WebView Means for Mobile Apps](http://developer.telerik.com/featured/android-5-0s-auto-updating-webview-means-mobile-apps/) for more information about this recent change and the impact it will have for hybrid mobile development.

* **When should you build a hybrid mobile app?**

Before committing to a platform strategy, it’s important to evaluate the technical and non-technical merits of hybrid versus alternatives like web and native – especially as it relates to your mobile application’s requirements. For example:

* + **Which mobile platforms do you wish to target?**

If you wish to target more than one platform, you have a number of choices. Clearly, the web offers a highly attractive solution for this requirement. Your target is the mobile browser. Hybrid lends itself well to this approach as well because of its reliance upon the WebView.

Native – on the other hand – finds itself in a unique space. If you are reliant upon the vendor SDKs and platform-specific programming languages then you are essentially coupled to the platform. In the case of iOS, it’s Objective-C or Swift; for Android, it’s Java; and for Windows Phone, it’s C#.

* + **Do you want to distribute your application via app stores?**

If you want to distribute your application via an app store, you must build a hybrid or native application. You cannot distribute websites through app stores; that’s what your browser’s address bar is for! Despite this restriction, whether you build a hybrid or native application, it’s highly recommended that you have a website available for your mobile application. This will be the first place your users will expect to go if/when they encounter problems.

* + **Are you looking to utilize the capabilities of the mobile device?**

Websites have a restricted set of abilities as opposed to hybrid and native applications. These restrictions are enforced by browser, effectively sandboxing it away from the mobile operating system. Recent developments with mobile browsers have exposed more device capabilities through HTML5 such as the camera, geolocation, and others.

Despite these advancements, support for advanced functionality is quite limited. For example, media capture and streaming remains unsupported in various mobile browsers. Because limitations like this remain in mobile browsers, many developers are compelled to evaluate hybrid and native as alteratives. Both offer the ability for developers to access device APIs – in the case of hybrid, this ability is supported through plug-ins.

It’s often asserted that native is best suited for applications where graphics performance is paramount. Mobile games are a class of mobile application almost entirely reliant upon complex visual interactions on the screen. Even if a game operates flawlessly from a functional perspective, you should expect it to have a very low app store rating if it feels slugglish. For that reason, developers have long-argued against using hybrid as an approach to build games.

That stated, a number of solutions for hybrid mobile applications exist. These include HTML5 Canvas and WebGL, both of which are well-suited for building applications like games. Furthermore, technologies like these are more approachable for developers through libraries like [Paper.js](http://paperjs.org/), [EaselJS](http://www.createjs.com/EaselJS), and others. And it’s not just for game development. For developers building more traditional, line-of-business applications, there are frameworks like [Famo.us](http://famo.us/) and [Kendo UI](http://www.telerik.com/kendo-ui).

* + **What are the technical abilities of your development team?**

Another factor to consider when evaluating your development options for mobile is your team’s technical abilities. If you decide to build a native application, you will require developers who are experienced with the platform SDKs and programming languages of every platform you wish to target. On the other hand, web and hybrid applications are built using web technologies like HTML, CSS, and JavaScript. You can have a team that can target multiple platforms with a single set of technologies. This is something worth considering when evaluating your opinions.

* + **Does the one-size-fit-all approach of hybrid really live up to its promise?**

Finally, it’s important to recognize that hybrid isn’t the be-all and end-all approach for mobile application development. Earlier in this article, I cited the challenge of overcoming the inconsistencies between WebViews. Other challenges remain. In fact, with hybrid, you can find yourself targeting the features of a mobile platform only to discover that they’re inaccessible because the plug-ins for them are out-of-date, unreliable, or (even worse) missing altogether. In this scenario, you are faced with the dilemma of either removing an application feature or writing the plug-in yourself.

If you find yourself developing an app that needs numerous native features, or an app where performance is critical to the success of the app, you may find the hybrid development model limiting. In this situation normally you would be forced to go all in with native development technologies, however, nowadays there are alternative approaches that let you build performant native apps without learning the ins and outs of tools like Xcode and Android Studio.

The JavaScript-based [NativeScript](https://nativescript.org/) framework, for instance, allows you to build iOS and Android apps that forego the WebView for enhanced performance. NativeScript also provides [100% access to iOS and Android APIs](http://developer.telerik.com/featured/nativescript-works/), removing the reliance on plugins that can prove limiting when developing hybrid apps. That being said, NativeScript is also not the be-all end-all approach for developing native apps. Although NativeScript’s approach to app development is more powerful, that power comes with a higher learning curve, as with NativeScript you’re no longer using HTML to build your user interfaces. Nevertheless, NativeScript is a serious option to consider if you’re building an app with non-trivial performance requirements.

Native and hybrid apps are installed in an app store, whereas web apps are mobile-optimized webpages that look like an app. Both hybrid and web apps render HTML web pages, but hybrid apps use app-embedded browsers to do that.