

# Hybrid vs. Native



**An Introduction to Cross-Platform Hybrid Development  
for Architects and App Development Leaders**

# The mobile delivery gap

Back in the early days of mobile, there was really only one way to give your users the performance and features they expected. You had to use each platform's native toolset.

Of course, that came with real and measurable tradeoffs:

- **Building in parallel for each mobile platform**
- **Managing multiple codebases**
- **Hiring and retaining highly specialized and costly native developers**

Meanwhile, demand for mobile experiences continues to grow exponentially. In 2022, consumers downloaded 255 billion mobile apps to devices, up by more than 80% from just five years prior.<sup>1</sup>

Given the time and cost of traditional native development, it's no surprise that many development teams are struggling to keep up with this demand.

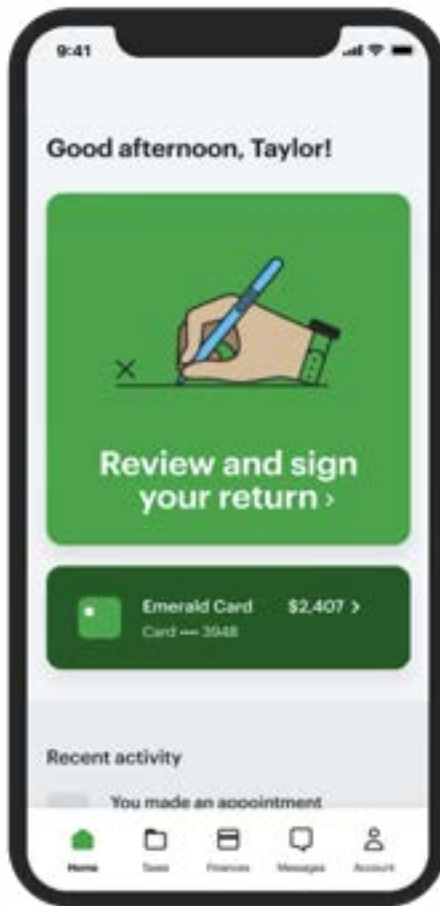
Thankfully, times have changed. As mobile and web technology have evolved, hybrid has emerged as a viable alternative to native. Many are now looking at hybrid development as a way to simplify and speed up app creation. Let's take a look at the reasons for this shift.



**Demand for mobile experiences is growing 5x faster than internal IT teams can deliver.**

GARTNER

<sup>1</sup> Number of mobile app downloads worldwide from 2016 to 2022 - Statista



H&amp;R BLOCK APP

# The choice to go hybrid

The growing adoption of hybrid is evident in a recent Stack Overflow survey that shows an equal share, if not at times close to double the amount, are using a cross-platform or web-based approach over native tools for their development needs.<sup>2</sup> Meanwhile, [top brands](#) like H&R Block, AAA, Burger King, and PwC have chosen a hybrid approach over native to power apps for their customers and employees.

**The top reasons for making the switch from native to hybrid, based on independent research and testimonials, are:**

**SPEED**

Building for multiple platforms from a single codebase often makes delivering cross-platform apps 2-3x faster than native.

**EFFICIENCY**

Reduced development times, and the avoided costs of hiring and retaining specialized native talent, can save teams more than \$31,000 per developer on average.

**DESIGN & UX CONSISTENCY**

With one codebase running on desktop, mobile, and web, hybrid apps provide better design and UX consistency across channels.

**SKILLSET**

Hybrid gives web developers and businesses with in-house web teams the tools to build powerful mobile apps using their existing skills and talent.

**Put together, the advantages of hybrid have helped centralized development teams close the gap and better satisfy the demand for mobile apps for customers and internal employees.**

# What is a hybrid app?

Hybrid apps are native apps. They're downloaded from the platform's app store or marketplace and offer the same native features, offline support, and hardware-based performance acceleration as any app built with a native SDK.

## WHAT IS A HYBRID APP?

The key difference is that hybrid apps are built using open web technologies like HTML, CSS, and JavaScript, rather than the proprietary or specialized languages used by iOS, Android, and others. That means anyone with a modern web developer skill set can begin building an app using the hybrid approach.

Hybrid apps run in a full-screen browser, called a webview, that is invisible to the user. Through customizable native plugins, they can access the native features of specific mobile devices (such as the camera or biometrics), without the core code being tied to that device.

That means cross-platform hybrid applications can run on any platform or device, all from a single codebase, while still delivering native performance.



# Comparing hybrid vs. native

In the sections that follow, we'll provide a one-to-one comparison of hybrid vs. native, highlighting the pros and cons of each approach.

It's important to keep in mind that the decision to choose hybrid or native should be based on the unique goals of your organization, the circumstances of a given project, and the composition of your existing development team.

Of course, we're betting big on the power of the web and the promise and potential of cross-platform hybrid development. But we understand that individual circumstances should drive which route you choose.

Hopefully, this comparison guide will give you some useful tips to help you choose the right approach for your next project. But keep in mind, there's often plenty of room to accommodate different approaches in a single organization. It's not always an either-or decision.

# Why hybrid?

Let's take a look at each of these.



Write once, run anywhere



Use the talent you already have



Deliver a consistent user experience across platforms



Deploy instant updates directly to users



Build for the future





# Write once, run anywhere

Rarely is a mobile app only designed for a single platform.

Consumers, partners, and employees all have a choice of platforms and devices.

## With a hybrid framework like Ionic, you can run your app on any platform or device.

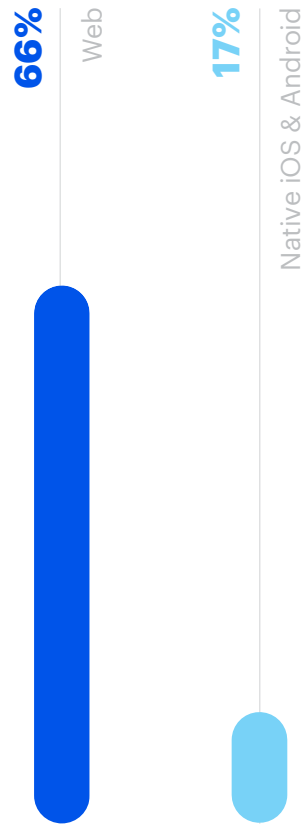
Following the native approach, that means you need to build separate apps for each mobile platform, and sometimes specific apps for tablets and smartphones.

This is where hybrid development shines.

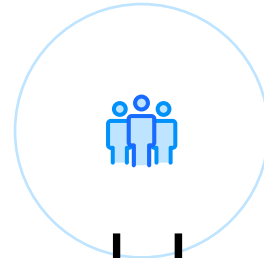
With a hybrid framework like Ionic, you can run your app on any platform or device, all from a single codebase. Ionic also provides adaptive styling, so the look and feel of your app isn't one-size-fits-all. It automatically adapts to each platform, respecting the differing characteristics like border radius, drop shadow, font weights, title locations between the platforms. This adaptive styling gives users an experience consistent with what they are familiar with and feels like a native app. And because hybrid technology is web-based, you can run your app in any modern browser as a Progressive Web App, or PWA.

That means your users get a great experience across platforms and devices, and you only have one codebase to worry about.





According to the 2023 Stack Overflow Survey, less than 10% of professional developers cited Kotlin, less than 5% cited Swift, and just about 3% cited Objective-C as familiar languages. In contrast, web developers made up nearly 66% of respondents.



# Use the talent you already have

The web developer community is about 7x greater in size than the number of native mobile app developers, according to the latest Stack Overflow survey.<sup>3</sup> Many development teams already have a deep bench of programmers who understand HTML, CSS, and JavaScript.

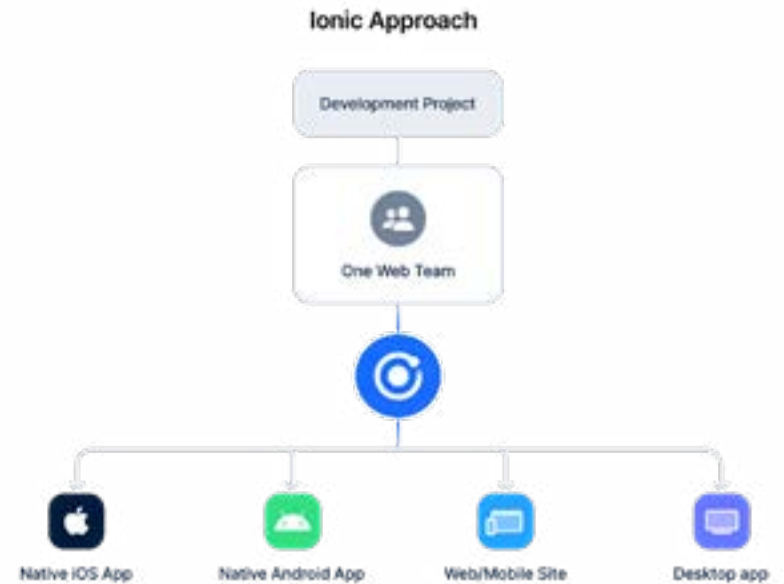
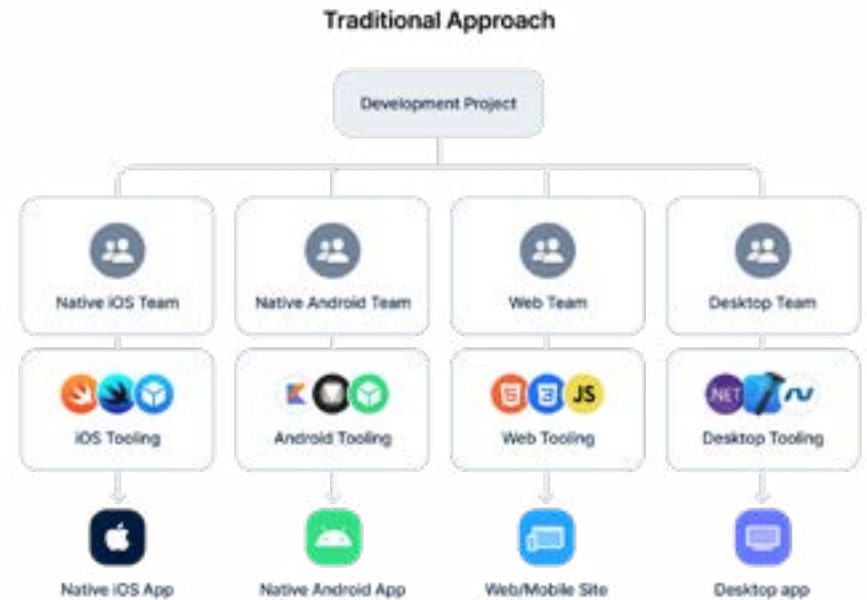
<sup>3</sup> 2023 Stack Overflow Developer Survey

**WHY HYBRID? | USE THE TALENT YOU ALREADY HAVE**

Why not leverage the talent you already have in-house to build your next mobile app?

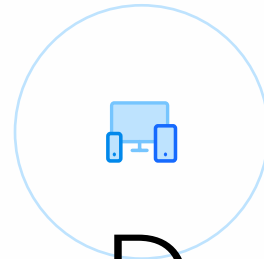
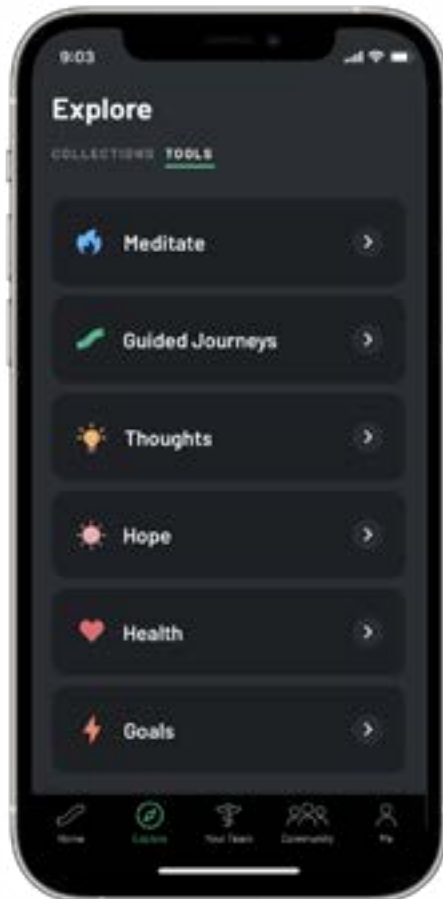
With a hybrid framework like Ionic, your existing web teams can build high-performance apps that run on any platform or device, using the tools and technology they already know and love.

That's a lot easier than outsourcing development, or recruiting, training, and hiring specialists. Plus, centralizing on a single skill set makes it much easier to reassign teams when a project is finished—whether that's a desktop web app or another mobile project.





SANVELLO APP



# Deliver a consistent user experience across platforms

Using hybrid development, the top-rated mental health app, Sanvello, consistently delivers a high-quality user experience to its 3M+ users, earning a 4.8 out of 5 rating on the Apple App Store.

Native advocates will claim that only a native approach can deliver the speed and performance that you need to create a great user experience. That may have been true a decade ago, but modern solutions offer the same hardware-based performance acceleration as native apps. Ionic is committed to providing a framework that is equal in performance to its native counterparts.

And as the Sanvello team found, user experience isn't just about performance. Simplifying the development process and consolidating onto a single codebase means more time to add features, fewer potential defects, and more time to fix bugs that find their way through.

Most importantly, users expect a seamless experience as they move across platforms and devices. If your mobile app is completely out of sync with your desktop or tablet app, that's not a good experience. By leveraging the web platform for both your mobile and desktop solutions, you will have a wider shared codebase that will yield a more consistent user experience.

**“ With hybrid, we can run our app across multiple platforms from the get-go, so we can focus on iterating features that are important to the product, as opposed to getting mired in the intricacies of native development. ”**

4.8

With over 3 million users



DALE BEERMAN, **SANVELLO**

*On building a hybrid app with Ionic*



# Deploy instant updates directly to users

No matter how rigorously an app is tested, it's almost inevitable that a bug will get pushed to production or an important update will need to be released to users quickly.

## WHY HYBRID? | DEPLOY INSTANT UPDATES DIRECTLY TO USERS

With native apps, there is no guarantee *when* an updated release will make its way to users' devices. These changes must pass through formal review by the Apple App Store and Google Play Store before the updated version is available for users to download and install.

Hybrid applications have the added benefit of receiving instant over-the-air updates. This means development teams can iterate quickly and ship critical bug fixes or feature updates for immediate impact directly to their users.

Ionic's Live Updates functionality unlocks the superpower of immediate deployments while remaining compliant with app store requirements. By leveraging instant updates, teams can rest easy knowing critical code changes aren't left pending in formal review processes and users can continue to enjoy the best possible user experience.







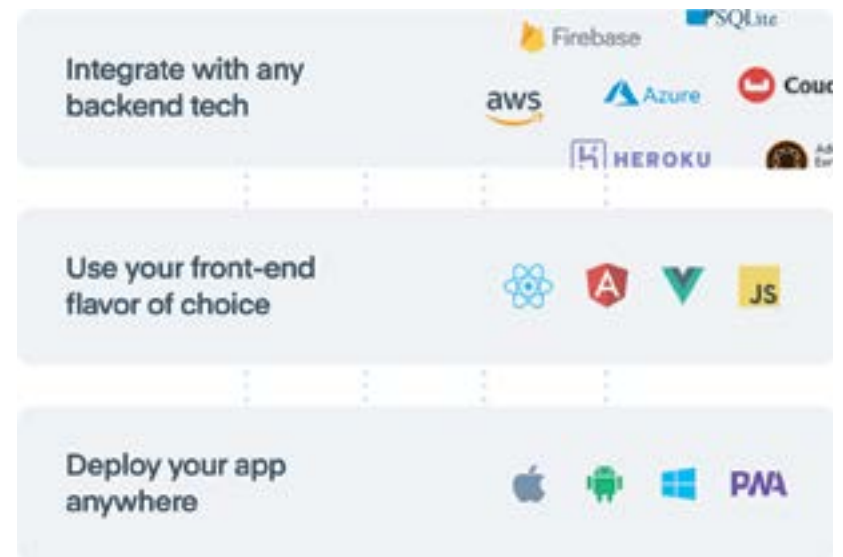
# Build for the future

Development organizations are tasked to build applications for the future. Moving now to the web platform offers you richer, more innovative options moving forward.

## The web represents the most stable and time-tested universal runtime in the world.

The web represents the most stable and time-tested universal runtime in the world. Today, the web is powering a growing and diverse set of applications — from traditional mobile and desktop apps to Progressive Web Apps, wearables, and IoT devices.

Additionally, by choosing a hybrid framework based on open web standards, you'll future-proof your development strategy in some other important ways. For example, the latest version of Ionic works with any JS framework, including React, Vue, and Angular, so your teams can continue to use the diverse set of tools and approaches they prefer — today and tomorrow.



# Tradeoffs of hybrid

Of course, hybrid applications are not without their tradeoffs:



## SYSTEM OVERHEAD

The use of the webview may introduce a degree of overhead compared to native. The abundance of performance APIs and increasingly powerful hardware have made this less of a factor in recent years, but it's still something to consider. For most applications, the difference in performance is hardly noticeable. But for 3D games and other performance-intensive applications, a hybrid solution may not be the best choice.



## PLUGIN MANAGEMENT

Cross-platform solutions like Ionic — as well as React Native and others — are able to access nearly every native feature of a device, like the camera or gyroscope, by using native plugins that unlock native device features and integrations using basic JavaScript. In a hybrid app, Capacitor plugins are the most popular solution to this problem. The use of these plugins does add complexity to development. Ionic offers a curated library of popular plugins known as [Ionic Enterprise SDK](#), with optional support and maintenance for teams building mission-critical apps. Nonetheless, this is a factor to consider.



## THIRD-PARTY DEPENDENCE

Choosing a cross-platform approach means you're placing trust in the maker of the framework vendor (whether it's Ionic, React Native, Flutter, etc.) to keep up with the latest and greatest native features and design patterns of each mobile platform. While Ionic is committed to keeping up with new Android and iOS versions, there's still a dependency any time you choose something other than the native SDK.

# Why native?

Let's explore the common reasons for choosing native.



Performance



Rich native library



No third-party  
dependencies

# Benefits of native development

Native is still the preferred approach for many mobile developers. And there are some good reasons for that. While part of that is based on the legacy of having few viable alternatives, native still has its advantages today.



## **PERFORMANCE**

Native code is still faster than Javascript and HTML. This matters when developers are looking to build demanding graphical applications such as games and other intensive animation applications. Mobile browsers are coming closer to bridging the gap for these types of intensive applications using WebGL; however, native still has the advantage here.



### **RICH NATIVE LIBRARY**

Using native SDKs allows the developer to access the latest features specifically designed for those platforms, without the complexity of dealing with native plugins. This is a great option if you're already familiar with native tools and languages and don't want or need an additional layer of abstraction.



### **NO THIRD-PARTY DEPENDENCIES**

By building exclusively with native SDKs, developers aren't bound to any third-party to keep up with support, and there's not as much of a dependency on open-source communities to keep up with the latest features.

# Challenges of native development

Here are the most frequently cited challenges associated with native development:



## **LONGER DEVELOPMENT CYCLES**

Native apps usually have longer development cycles, especially when building for multiple platforms, which requires two or three different code bases for iOS, Android and desktop. Each platform has its own nuances which require specific changes, updates, and maintenance which bloat the cost of an application and add development time. This creates a lot of iterations within the development process in order to customize and test for each platform, and also reduces your agility to launch your application or push updates.



### **HIGH DEVELOPMENT COSTS**

Simply put, developing mobile applications natively is time-consuming and expensive—think tens of thousands of dollars per developer more expensive. This is mostly driven by the time it takes to build for each platform along with the cost of hiring and retaining highly specialized native talent.



### **NATIVE TALENT HARD TO FIND**

Finding and hiring iOS and Android developers is difficult, given that less than 7% of all developers are versed in the necessary programming languages. It's also difficult to repurpose those developers for other projects outside of mobile.



### **LIMITED CUSTOMIZATION**

Creating custom UI components to match your company's design system or pattern library can be more challenging with native components, because you're limited to the design patterns supported by each platform. And unlike web components, native components can't be shared outside of their native platform, so you'll need to maintain multiple UI libraries.



# Comparing cross-platform approaches

While the various cross-platform frameworks available today—Ionic, Flutter, and React Native—may appear similar on the surface, there are a lot of differences between them that you'll realize as you dig in.

To summarize *very* briefly, Ionic Framework and tooling are all based on open web technologies, from the languages that you use to build Ionic apps (HTML, CSS, JavaScript), to the standards-based UI components running inside your app. In that sense, when you choose Ionic, you're

really choosing the web—the most time-tested, universal runtime in the world. You build with the web, and we give you the tools and component libraries to help you succeed.

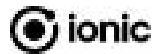
## COMPARING CROSS-PLATFORM APPROACHES

Most of the other frameworks add some level of custom UI rendering to work across platforms. For example, React Native translates your JavaScript code into native code at runtime, and uses the native UI elements provided by iOS and Android. Flutter works with the Dart language and uses its own custom graphics engine to work across devices.

Each approach has its benefits and tradeoffs, which typically come down to debates about performance, code sharing, customization, and portability.

There's a lot to cover here, so if you're interested in learning more, check out our comparison guides: [Ionic vs. React Native](#) and [Ionic vs Flutter](#).





# About Ionic

Ionic is best known for the popular open-source Ionic Framework, a rich library of front-end building blocks and UI components that make it easy to design beautiful, high-performance mobile and Progressive Web Apps (or PWAs) using web technologies like HTML, CSS, and JavaScript.

On top of that, Ionic is an end-to-end platform to help professional developers and teams speed up and simplify app development and delivery across mobile, desktop, and the web.

Today, Ionic is powering apps for [major brands](#) like H&R Block, AAA, Burger King, and PwC. Ionic is backed by a vibrant community with millions of developers all over the world, and thousands of global meetups, forums, and community-driven events that make it easy to learn and grow with our platform.

# Additional resources

Ready to get started? Check out our [world-class developer docs](#) for install instructions or our [Quick Start video](#) on YouTube.

Here's some additional reading if you want to learn more:

- [What is Cross-Platform App Development](#)
- [What is Hybrid App Development](#)
- [What is a Progressive Web App and Why Do You Need One](#)
- [The Architect's Guide to Progressive Web Apps](#)
- [Evaluating Ionic for Enterprise Development](#)
- [Ionic React vs React Native: What are the Differences Between Them?](#)
- [Ionic vs. React Native: A Comparison Guide](#)
- [Ionic vs Flutter: A Comparison Guide](#)