It's Time to Embrace React Native.

WHY WE RECOMMEND THIS BEST-IN-CLASS CROSS-PLATFORM FRAMEWORK

WILLOWTREE®
a TELUS International Company

Introduction.

How are we going to build this product?

What tech stack will help us achieve our vision?

Where can we cut costs without sacrificing quality?

We've heard these questions a thousand times. Every client comes to WillowTree with distinct budgets, capabilities, and goals, so **every digital product we create is bespoke**. Before writing code, one of the most significant decisions we make is selecting a product's **development framework**.

Options typically include progressive web apps (PWAs), hybrid-web "wrappers," native app frameworks, and **cross-platform frameworks**.

Why so many frameworks? Native apps built with tools from Apple and Google are the gold standard, but they're expensive. Third-party tools can simplify the process, but each is nuanced, offering benefits *and* drawbacks. These nuances can lead to client misconceptions and create false cost/time expectations.

To make matters more complex, developer communities have historically disagreed on preferred frameworks, and platforms are rapidly evolving to meet modern demands. For nearly a decade, frameworks like React Native, Xamarin, and Flutter have been duking it out.

A note from WillowTree
Partner & VP, Solutions Architecture
Alex Shafran

Together, these complexities breed confusion – especially for clients new to mobile products. Is it possible to build a product *once* and operate that product *anywhere*?

As experts in digital product development, **it's our job to understand client constraints and recommend the right tool** – or combination of tools – for the job.

WillowTree has worked with all of these frameworks, and we've learned the pros and cons of each. This eBook will focus on cross-platform solutions, which offer a comprehensive capability set with potential pricing efficiencies (15-25%, not 50%).

We created this eBook to help readers:

- Gain clarity on cross-platform options and capabilities
- Aid collaborative decision-making with key stakeholders
- Determine the appropriate development framework for their mobile products

And, spoiler alert, we've seen enough to make a call confidently: if you're going cross-platform, we recommend **React Native** as your development framework.

Here's why.

Be proactive about React Native.

A note from WillowTree
Director, Solutions Architecture
Sean Harvey

Based on recent consensus from the global developer community and WillowTree clients, the verdict is in: **React Native is now the market leader for cross-platform development**. It's a great tool (when used in the right scenarios), and its JavaScript core allows us to cross-train developers while simultaneously hiring dedicated React Native talent.

In short, we're all in.

One of the key reasons we feel confident making this commitment and investment is that we've taken the time to put React Native through its paces.

React Native is a cross-platform mobile framework released in 2015, spawned from the web framework React, created by Meta (Facebook) in 2012. Some early adopters ran into problems (such as Airbnb, who jumped aboard in 2016 and then sunset their React Native practice in 2018).[1] The framework evolved and matured, and in 2021, WillowTree launched our official **React Native Pilot Group** to definitively answer whether this was the proper framework for our developers and our clients.

An initial cohort of six WillowTree developers spent three months diving into the latest iteration of React Native. We used the framework to build an internal clone of the FOX Weather app – a complex and highly successful (hit #1 in the App Store) mobile product that we'd initially helped FOX build natively for iOS and Android. We sought differences, difficulties, and discrepancies. What would it take to upskill our team and recreate this app using React Native?

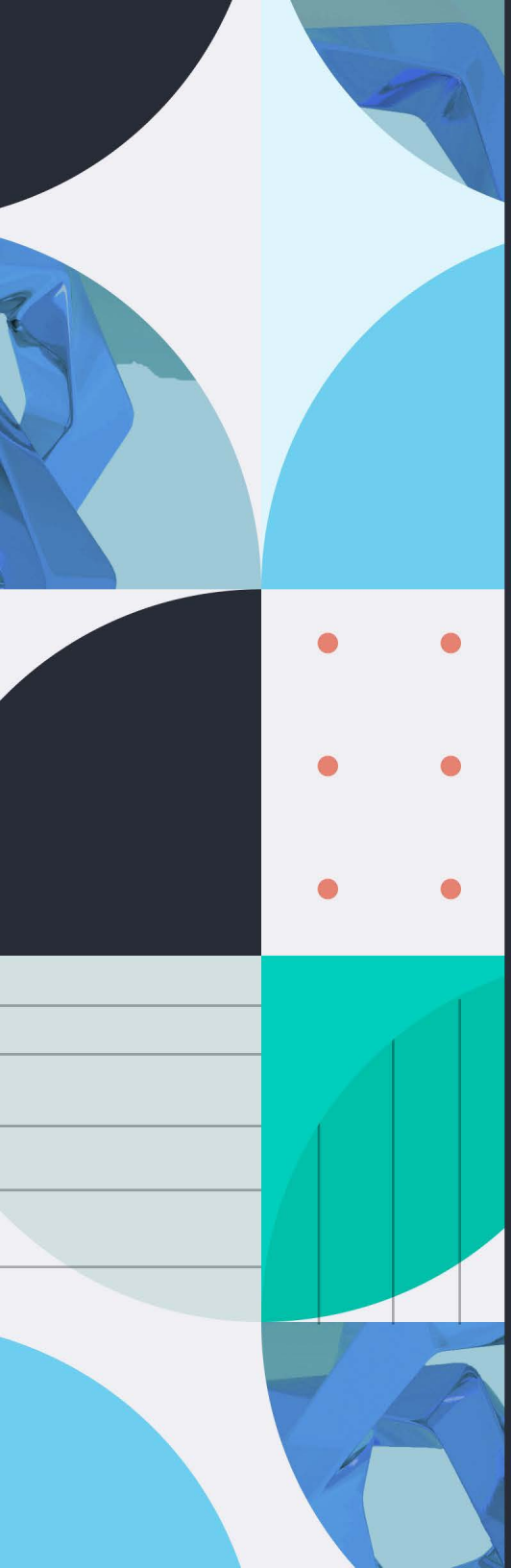
Since we'd already spent nearly a year collaborating with FOX to help natively develop the company's weather product, we could draw on those learnings. Still, we were encouraged by the relative speed and ease of rebuilding the app in React Native.

While there are certainly pros and cons when it comes to developing cross-platform vs. natively – and we'll explore all of these in this eBook – **this effort convinced us that React Native is a best-in-class framework.**

Let's get into it!

Please find sources listed at the end of this document.

Here's what we'll cover in this eBook.

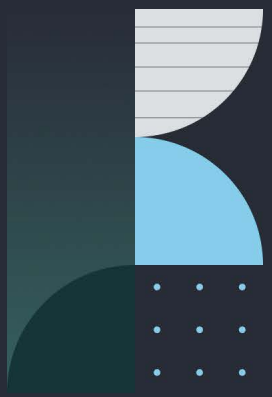


Part 1

Setting the Foundation

The development environment includes three layers: operating systems, development frameworks, and programming languages.

These three layers are intertwined. Understanding them will help you make informed decisions around the pros and cons of cross-platform development.



Two operating systems to rule the world.

For all organizations providing mobile products to a broad user base **it's essential to develop products that work on both iOS and Android devices**.

Apple's operating system (**iOS**) dominates the U.S. market with **58%** of **domestic** market share.[2]

Google's mobile operating system (**Android**) accounts for **72%** of **global** market share.[3]

Together, iOS and Android operating systems account for **99.4%** of global market share.

So. Many. Development frameworks.

Once an organization commits to developing products for both operating systems, the next step is to **choose a development framework** – the overarching structure and set of tools engineers use to write code and build an application.

Native mobile app development builds a product exclusively for one operating system – iOS or Android. Since both operating systems have developer tools, we create one codebase for iOS and another for Android.

Initially, this was the only way to build mobile apps; today, it is still the preferred approach for 87% of apps released.[4] While native frameworks offer the most customized set of tools for building mobile applications, they require the extra effort to create a unique codebase for each operating system.

With **cross-platform app development**, we use special tools to build a single product that functions on *both* operating systems (stay tuned for lots of caveats, but that's the gist). Some of the most popular cross-platform frameworks in recent years include Flutter and Xamarin, among others, and our pick, of course: **React Native**.

Because the magic of React Native is its programming language...

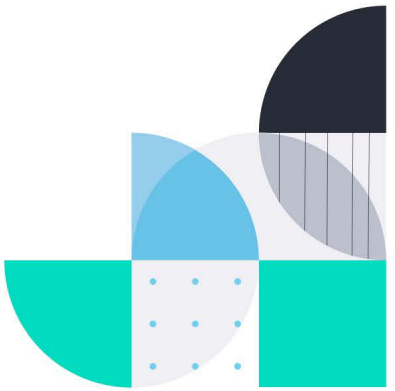
Native mobile app development is still the preferred approach for **87%** of apps.

To translate our vision to machines we use programming languages.

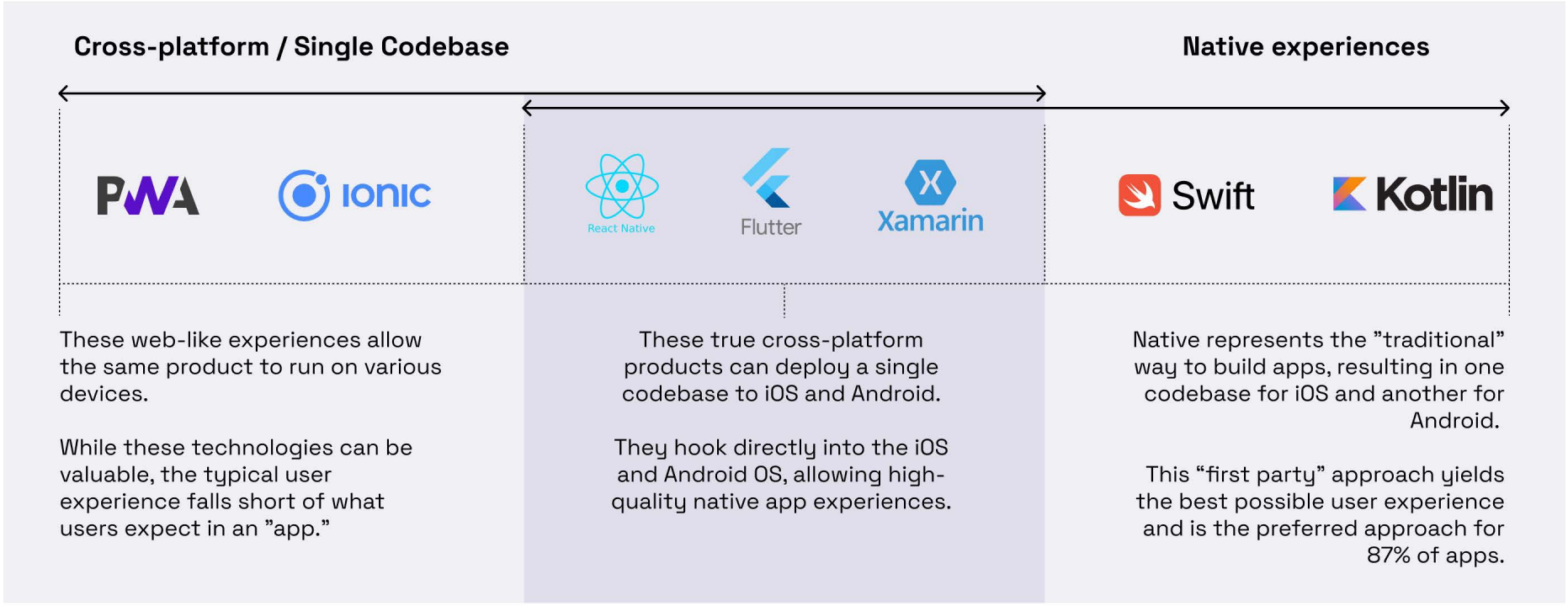
Kotlin is the programming language used by the native Android development frameworks, and **Swift** is the typical language for iOS.

The various cross-platform frameworks we will mention each have their own programming language. Flutter uses **Dart**. Xamarin uses **.NET**. But Dart isn't used for anything other than Flutter. And while .NET is re-inventing itself to run on various devices, it's not a common language among modern front-end developers.

Unlike its competitors, **React Native uses JavaScript** – far and away the most popular programming language worldwide. Spoiler: this is a huge reason we recommend React Native as our best-in-class cross-platform framework.



Nuances of native, cross-platform, and web development frameworks.



In this eBook, we're focusing on these **cross-platform experiences** compared to native or web-like experiences...

...but if you're curious about web experiences, here's a quick explainer. →

AN IMPORTANT NOTE: ADDITIONAL OPTIONS FOR CROSS-PLATFORM DEVELOPMENT

There are other solutions for building apps across platforms, though this eBook will not explore them in-depth. Here's a quick run-through.

Hybrid-Web "Wrappers"

Products like **Ionic** render webviews within an app frame. While these tools have additional integrations with some OS-specific capabilities (e.g., push notifications and camera integration), the core experience is powered by a browser, which looks and feels foreign among the other native apps on a user's device.

Button interactions, scrolling, and other gestures don't always translate well, and the result is a product that simply **replicates the experience of navigating a website**. We prefer to create experiences that add value by leveraging the unique capabilities of mobile devices, so we do not actively recommend these frameworks.

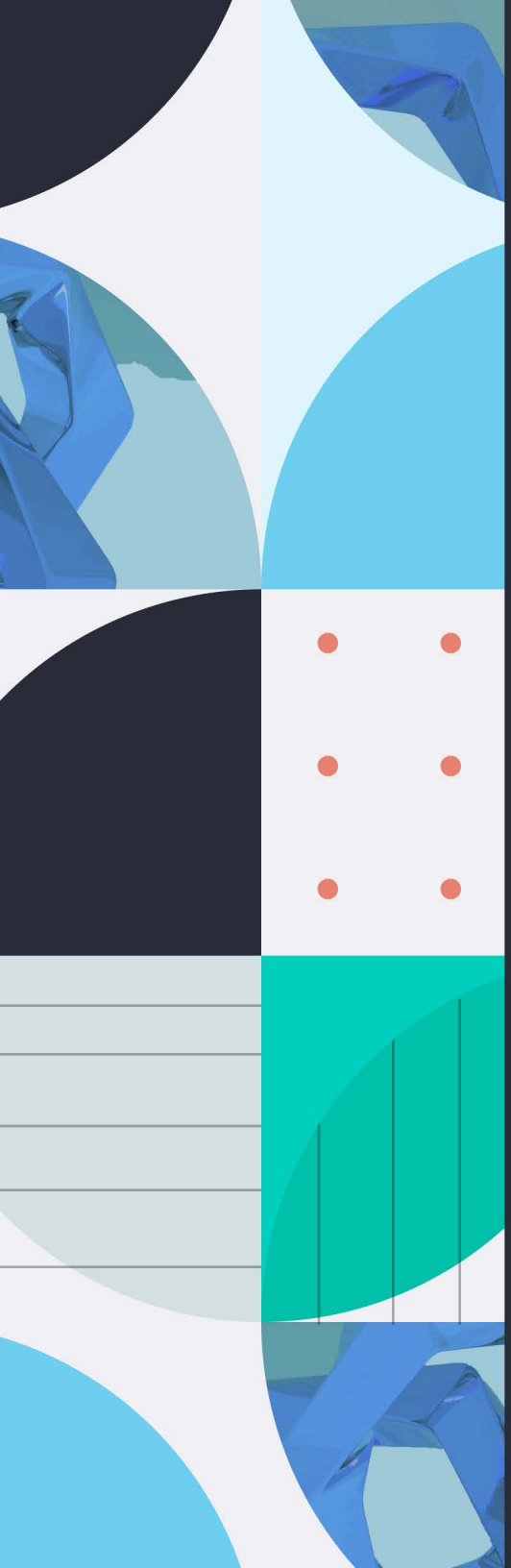
Progressive Web Apps (PWAs)

We love PWAs, but the term "Progressive Web App" is a bit of a misnomer. PWA is a technology designed to make web apps more friendly to mobile devices rather than supplant native apps.

However, PWAs have a significant adoption issue: they're unavailable in app stores. While they run seamlessly in a browser when accessing a website, their **required multi-step "installation" process** is not ideal (mobile users must navigate to Chrome/Safari, locate the browser menu, click "Add to Home Screen," then select "Add" on a pop-up).

Even if we overcome such hurdles, these PWAs also run in a framed webview, facing the same interaction and experience challenges as Hybrid-Web "Wrappers." To make matters worse, iOS users have historically faced technical limitations, such as a lack of support for push notifications.[5]

We're nonetheless big advocates of PWAs. Frequently, we recommend deploying them as a **part of our clients' overall mobile strategy** – rather than their only strategy. Starbucks and Uber, among many other brands, leverage PWAs and native apps for different portions of their digital portfolios.[6]

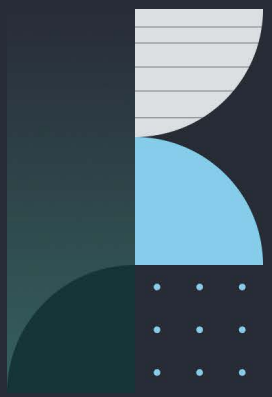


Part 2

But First, Why Embrace Cross-Platform?

Why are we advocating for a cross-platform approach in the first place?

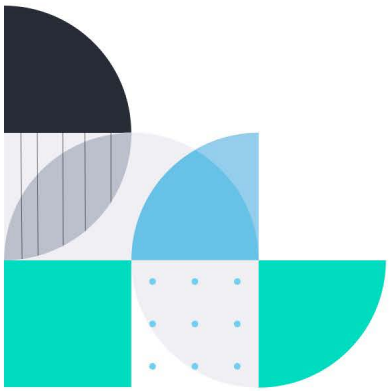
Before we dive into React Native specifically, let's back up a bit. The benefits we see in cross-platform come down to these key factors: **time and cost savings**, **market demand**, and **continuous innovation**.

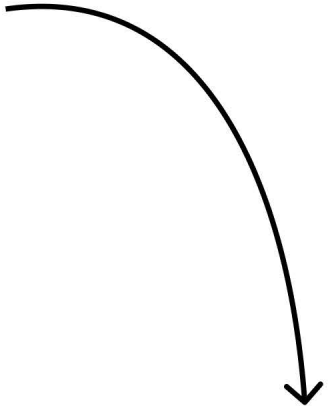


1 Code reuse drives increased velocity and cost savings.

Here's the main reason to consider cross-platform: developers can write one application and deploy it to both iOS and Android, which reduces a wide range of development costs.

As we'll discuss in the "What to Expect: Cost and Time Savings" section, a significant portion of the codebase on iOS or Android platforms can be written exclusively using React Native, and a minimal amount needs to be platform-specific native code.





The single codebase offered by React Native also reduces **future costs** of supporting two distinct codebases, including bug fixes, product maintenance, and updates.

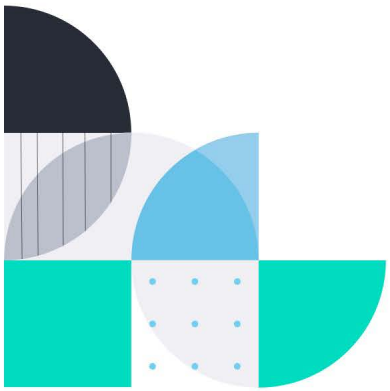
2

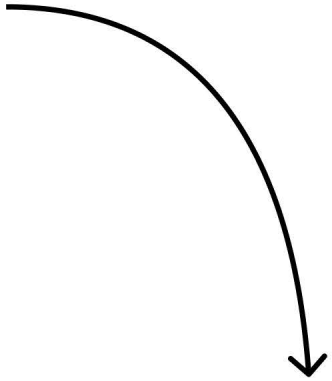
There's a growing market demand for cross-platform frameworks.

Massive brands like Bloomberg, Shopify, Microsoft, and NFL are now using cross-platform frameworks, and more of our clients are now explicitly asking for cross-platform tooling.[7]

Moreover, the cross-platform market is forecasted to grow sixfold in the coming years to a market value of $365 billion by 2028.[8]

In short, the movement is happening, and as leaders in mobile development, we need expanded solution sets for our customers.





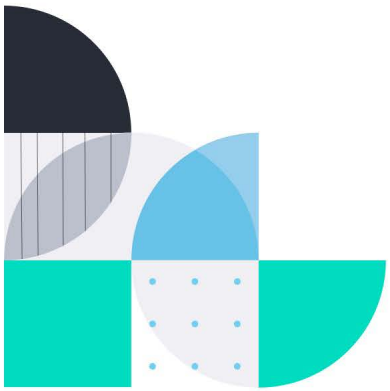
> The good news? There's supply to match this demand, with cross-platform tools developers actually enjoy using (*cough* React Native).

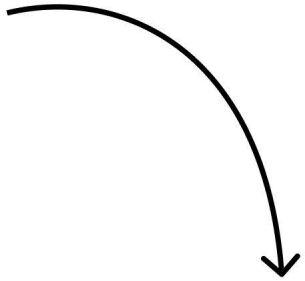
3

Industry leaders anticipate and embrace change.

In 2008, WillowTree swiftly embraced the iOS SDK when it was released. Sure, now it seems like an obvious choice, but back then, the iPhone was unproven technology, and it took time for big brands to jump on board. This is how innovation works: those on the leading edge (or fast to follow) can ride the wave and maximize growth.

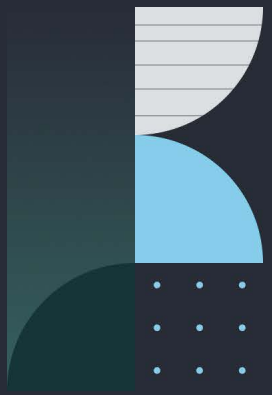
Our clients expect us to bring innovative methods to the table, and in presenting cross-platform, we're using the word "embrace" intentionally. We don't want to *replace* all native and alternate mobile solutions. We will only recommend a solution once we've tested it and we're now confident enough to add cross-platform to our bag of tricks.





> The cross-platform market is now in that innovation sweet spot where it's mature enough to trust while also rapidly evolving.

With those advantages in mind, let's take a look at React Native.

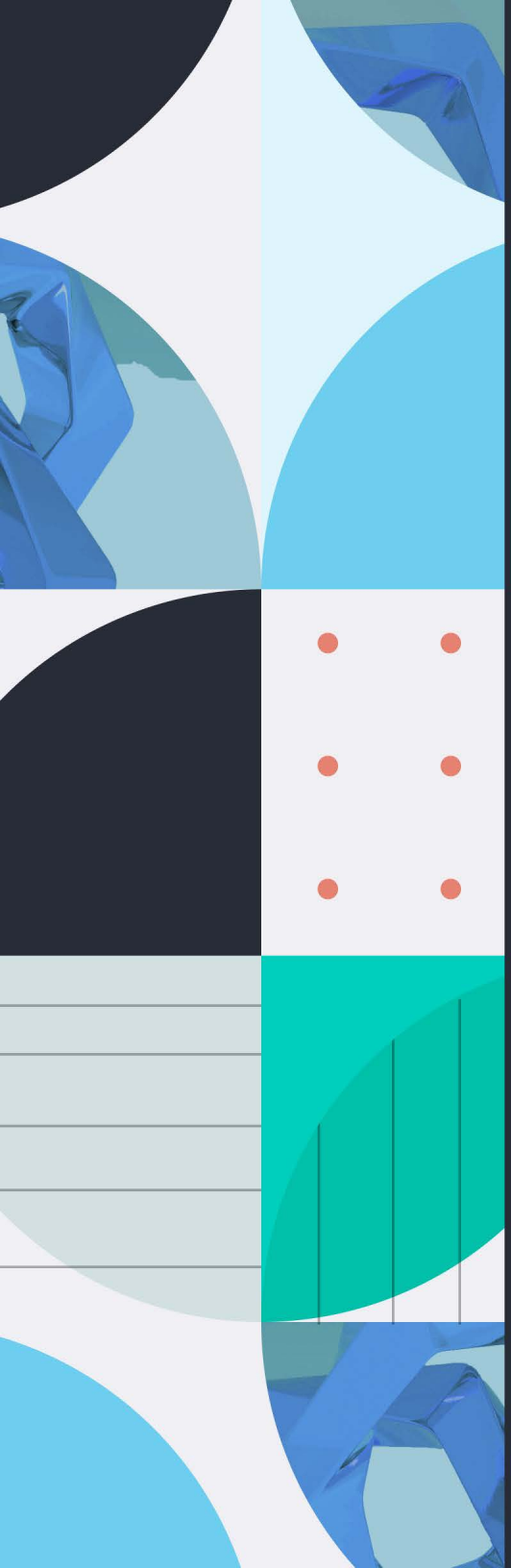


REACT NATIVE BUILDS PREMIUM PRODUCTS THAT GO TO MARKET WITH EXCEPTIONAL SPEED

"With React Native, there's a real benefit in terms of the speed at which you can go to market. React is a straightforward framework to learn, and JavaScript is the most popular language in the world. The ubiquity of the programming language means you have access to a large talent pool familiar with this mature and popular framework. You get all of those benefits essentially within native application development. React Native is a powerful solution whether you're deliberately developing a premium product or quickly testing a new idea that must be built and deployed in a matter of weeks."

Insights From WillowTree Experts

Sean Harvey
Director, Solutions Architecture



Part 3

Why React Native?

React Native offers several key differentiators.

While we weren't so confident a few years ago, the case for React Native has become relatively straightforward. While all the cross-platform tools we tested offer production-level quality, React Native adds these critical differentiators:

1 Market popularity

Over the last 12 months, our cross-platform work has been primarily in React Native. At WillowTree, we see continuous, high-quality React Native lead flow, with little to no leads for Xamarin or Flutter.

2 Developer preference

Engineers enjoy React Native's advanced tooling and the broad developer community they can rely on for support. Put simply, they enjoy using the framework.

3 Low barrier to entry

React Native's underlying programming language (JavaScript) makes it easier to cross-train web developers, so they can contribute to a mobile product.

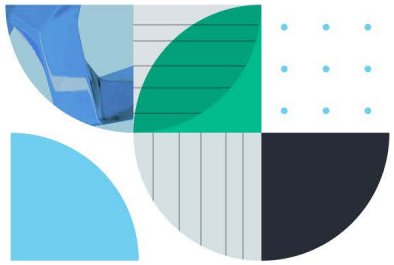
Let's break this down a bit further.

React Native is thriving across verticals.

REACT NATIVE DOMINATES THE MARKET

The platform isn't going away anytime soon. React Native currently leads the cross-platform market in Top 500 app installs. It's even more prevalent in healthcare and financial services verticals.

At WillowTree, zero clients have inquired about new Xamarin products, despite our roster of trained engineers and deep delivery experience. Requests for Flutter have only surfaced in the context of prototypes and one-offs.



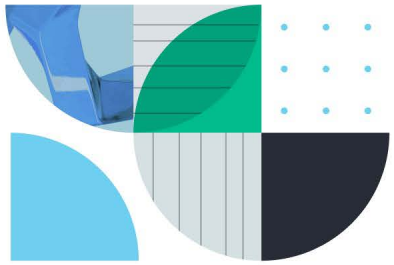
React Native dominates cross-platform market share at **6x** the competition in Top **500** app installs and **20x** GitHub contributor count.

Developers like React Native – a lot.

HAPPY TEAMS CREATE BETTER PRODUCTS AND MORE SATISFIED CLIENTS

Cross-platform tools introduce a new "layer" between the developer and the operating system. This additional step can introduce friction points at every stage in the development process.

While we can't eliminate all friction points, our teams have shared specific praise for the React Native tooling, the fast pace of development (e.g., hot reload, rich UI libraries, etc.), and the robust developer community providing support.



Developers praise React Native's tooling, fast development pace, and robust community support.

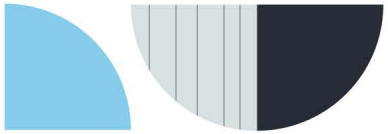
React Native offers a low barrier to entry.

EXISTING WEB ENGINEERS FEEL AT HOME IN JAVASCRIPT

We've said it before, but it's an important distinction: React Native has strong, easy-to-implement cross-training potential due to its foundation in JavaScript.

Because it shares conventions associated with React – the most popular JavaScript framework – developers can quickly feel at home when they begin using it. For product companies, this means a smaller development team. For agencies, this means you can cross-train your existing web engineers to build native apps.

We've also established a development training program to help clients develop a deeper bench of React Native engineers.



React Native's foundation is in JavaScript, which is used by **~70%** of professional software developers. **Kotlin** is the second most popular cross-platform language, with just **9%** of devs reporting extensive usage.[9]

Comparing pros and cons across frameworks

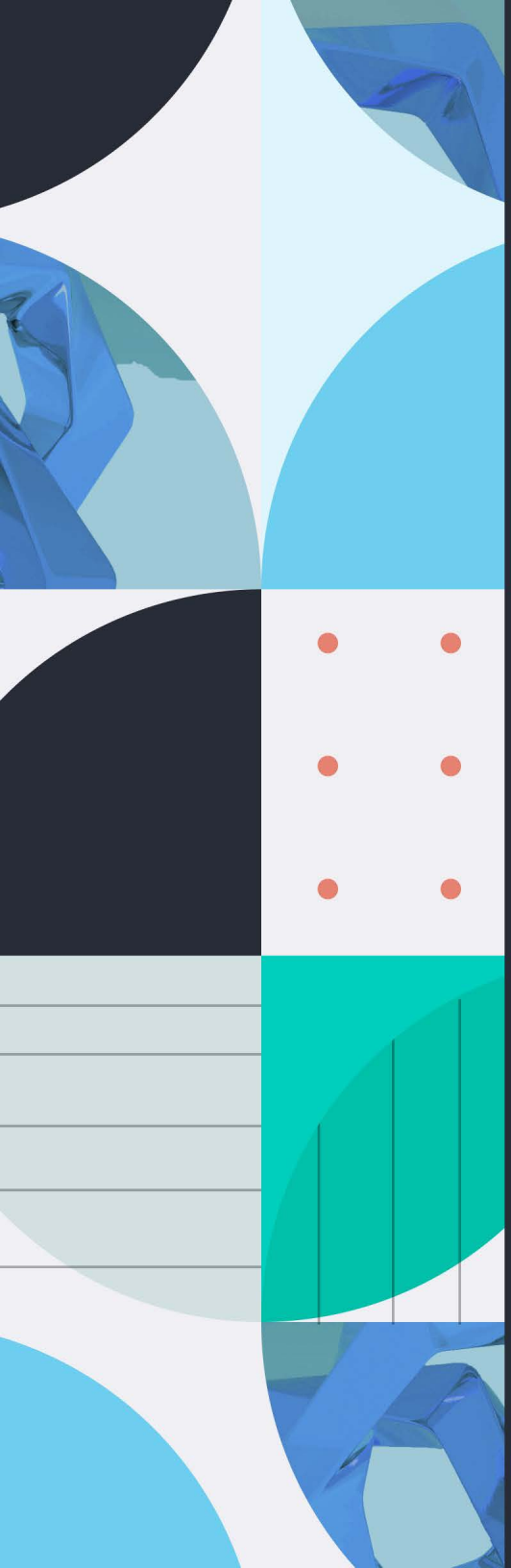
Framework	React Native		Xamarin		Flutter	
Maturity	Battle tested by a wide variety of industry-leading brands and mobile products	✓	Despite low and stagnant adoption, Xamarin has stood the test of time.	✓	Evolving too quickly for WIllowTree to recommend as a stable, long-term solution	X
Language Familiarity	Javascript: Robust developer support and low barrier to entry; used by 70% of developers	✓	C#: Familiar to .NET development teams, which account for 30% competency in the market	—	Dart: Emerging, with only 6% of developers reporting extensive usage	X
Market Popularity	Leads the market with 8.63% of of Top 500 app installs (as of Feb 2023)[4] and 20x GitHub contributor count	✓	Much less popular, with only 0.1% of Top 500 app installs	X	Minimal, and only in context of prototypes, with 1.24% of Top 500 app installs	X
Developer Experience	One of the first platforms to introduce "hot reloading"; developers continue to enjoy UI frameworks like Litho and ComponentKit	✓	Smaller developer community affects developers' ability to learn and collaborate	X	Initial developer feedback seems positive, citing overall ease of ramp up and appreciation for new features	✓

REACT NATIVE'S EVOLUTION POWERS NATIVE-LIKE CAPABILITIES

"React Native has experienced many updates in a short time. Five or six years ago, I often had to go to a native framework or read tons of documentation to do complex tasks. Back then, React Native was still getting started. Now that it's received more attention and developers have improved the framework, you don't have to write native code unless you want to connect with a specific module. It's become much easier to write in React Native, and the results are almost the same as in Android- or iOS-native."

Insights From WillowTree Experts

Stephanie Cure
Staff Software Engineer



Part 4

Will React Native Work for Me?

Features, UX, and additional criteria will influence your React Native evaluation.

We've now spent several years stress-testing React Native for our clients. In that timeframe, we've seen it mature into a leading cross-platform framework, and we know from experience what we can efficiently achieve with React Native.

Here's how we evaluate whether React Native is the best solution for a particular product:

1

Is it possible to build the product in React Native? Some features are just not feasible to build in React Native. Developers can always "dip out" of React Native and write native code, but as the amount of native code increases, the value of using React Native decreases.

2

If so, is React Native the best tool to achieve the desired business objectives? When deployed correctly, React Native introduces many efficiencies in app development. However, for our clients who require the best user experience or want to innovate within the latest Apple and Google technologies and devices, React Native is not always the best choice.

1

Step 1: Is it possible?

React Native can support most features. But, in certain device-specific cases, developers must incorporate native code to connect an app with OS-specific products. React Native isn't an optimal solution for apps that will depend on one or more of these features:

- Wearable technology (e.g., Apple Watch)
- Widgets
- Bluetooth-layer logic
- Voice assistants (e.g., Siri, Google)
- Connected home device integration (e.g., HomeKit)
- Multi-threading

It's always an option to start with React Native and bolt on native code later in the process. But, there are downsides to that approach. Bouncing between React Native, iOS, and Android increases complexity and poses hiring challenges (you'll need developers proficient in all three languages).

2

Step 2: Is it a good idea?

Even if React Native *can* support a feature, it is not *always* the ideal framework. To simplify the decision-making process, we've developed a set of criteria for understanding whether React Native is suitable for your application.

Some clients come to us with ideas on the specific tech or approach they want us to use in building their products. Other clients desire a lower price point or have immovable deadlines they need to hit.

And others still don't have a particular point of view on native vs. cross-platform – they want the right tool for the job.

Going cross-platform doesn't always offer the best solution. There's some gray area here; the ultimate answer isn't a simple **yes** or **no**. It's a spectrum. Consider the criteria on the following page and see where you land.

Criteria and tips for success.

Check off all that apply:

- ☐ You prioritize cost and developer efficiency.
- ☐ You won't gain anything financially by incorporating the most up-to-date iOS/Android capabilities.
- ☐ You're less concerned about having the best app in your field.
- ☐ Your app provides basic user input and output. Hardware integrations, processing speed, and other advanced functionality are not required.
- ☐ You have a dev team familiar with JavaScript and can help maintain the product.
- ☐ This app will not be deployed into a family of apps that could share components.

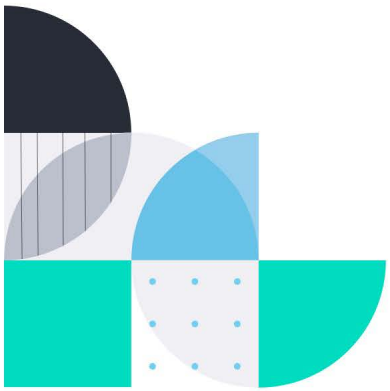
If you selected more answers on the left, **cross-platform** might be the right choice.

- ☐ You prioritize being ranked #1 in the App Store.
- ☐ Your revenue-generating product will lose money if you don't have the latest and greatest features on day one.
- ☐ You're not willing to compromise product/UX for 15-25% cost savings.
- ☐ Your app requires specific integrations with device-specific hardware (e.g., features unique to Apple Watch or Home Screen Widgets).
- ☐ You don't foresee staffing an internal dev team to maintain this product.
- ☐ There may be additional apps released that could benefit from code sharing.

If you're more aligned with the right side, you likely want to stick with **native iOS and Android** development.

WE ENCOURAGE STAKEHOLDERS TO KEEP THESE BENEFITS TOP OF MIND:

Platform independence: React Native fits well for products not directly dependent on innovation roadmaps followed by first-party tech companies. If you don't need to integrate Apple's newest features or Google's newest SDK, React Native might be a great fit. Since React Native is community-based, it takes some time for adoption to catch up to Apple and Google.



Internal capabilities: If you have internal engineers already familiar with React on the web, the conventions and framework will be very similar. Onboarding and cross-training will be reasonably efficient: **WillowTree's development training program** has often proved valuable to our clients. This existing talent pool inside a company can quickly become an untapped investment area for clients to leverage.

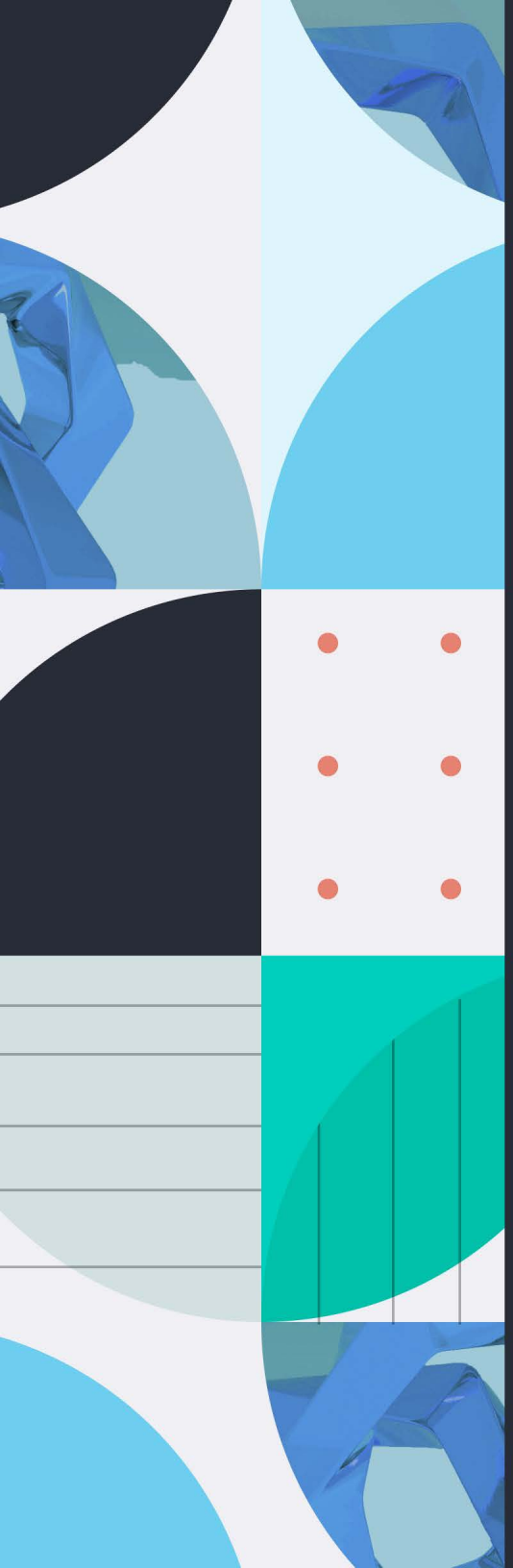
Scaling beyond mobile: When we think of multi-platform, we can also think beyond mobile into other product areas, such as over-the-top (OTT) video streaming platforms. One of our clients wanted to explore developing and simultaneously deploying a new streaming platform on five distinct platforms. Building in React Native was a quicker, cost-efficient option – taking a single codebase written in JavaScript and then testing and deploying it on many platforms simultaneously.

REACT NATIVE BUILDS SIMILAR APP EXPERIENCES ACROSS PLATFORMS

"We've been pushing the boundaries to discover where we wouldn't use React Native. I've not seen that spot yet. A few years ago, we saw more glaring omissions. These days, if you need to be in full lockstep with Apple Watch, or build some custom Bluetooth application, then absolutely go all-in on the native tech stack. Otherwise, if you're trying to build a reasonably similar experience across platforms and are willing to tolerate minor discrepancies and compromises, React Native will get you there with built-in efficiencies."

Insights From WillowTree Experts

Tim Troxler
WillowTree Engineering Director



Part 5

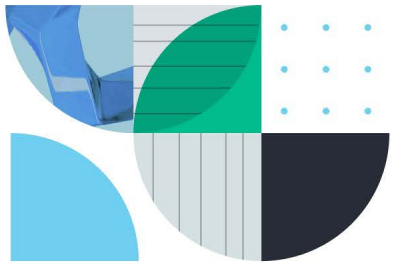
What to Expect: Cost and Time Savings

Let's talk numbers.

Here's the bottom line: from a purely economic standpoint, React Native often makes sense.

Under the right circumstances, React Native enables an agency like ours to build premium solutions efficiently.

And if an organization hopes to build or maintain its digital product in-house, it'll be easier and economically advantageous to hire and cross-train for React Native vs. multiple native or other cross-platform frameworks.



React Native delivers premium solutions with an estimated **15-25%** cost savings. The shared codebase specifically offers a **30-50%** reduction in developer effort.

React Native saves development time!

Before we walk through these estimates, let's discuss a few important considerations. One might assume that creating a single codebase for two operating systems cuts costs by 50%. For pure development effort, this can *sometimes* be true.

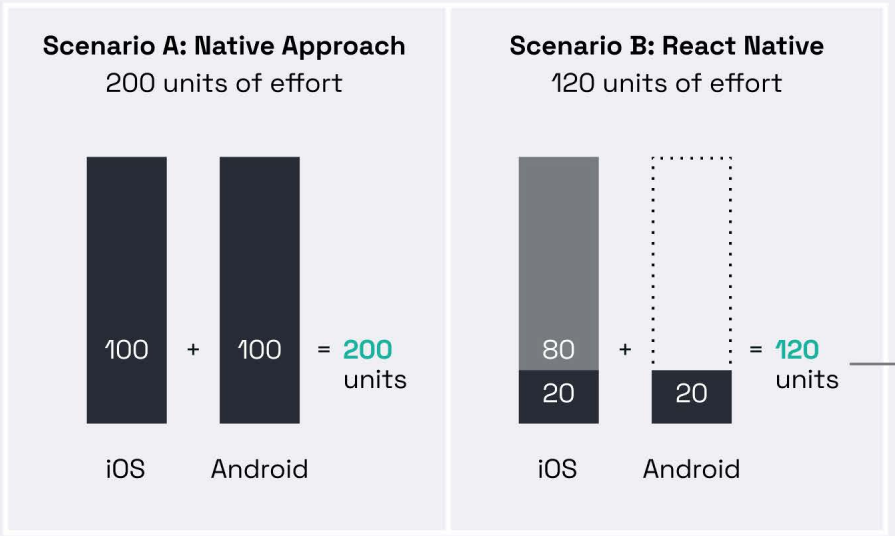
However, we don't always see this level of savings on client projects, and Agile product teams include many essential roles beyond writing code.

With that in mind, the magical 50% number doesn't apply to the entire team equally. Any digital product company or agency needs to think critically about total team cost rather than developer cost alone.

With these nuances in mind, here's how we arrived at those efficiency estimates:

Imagine an app that takes 100 "units" of effort to develop where 20% of the code cannot be shared across platforms:

- Platform-specific Code
- Shared React Native Code
- React Native Savings (React Native code that Android and iOS can use)



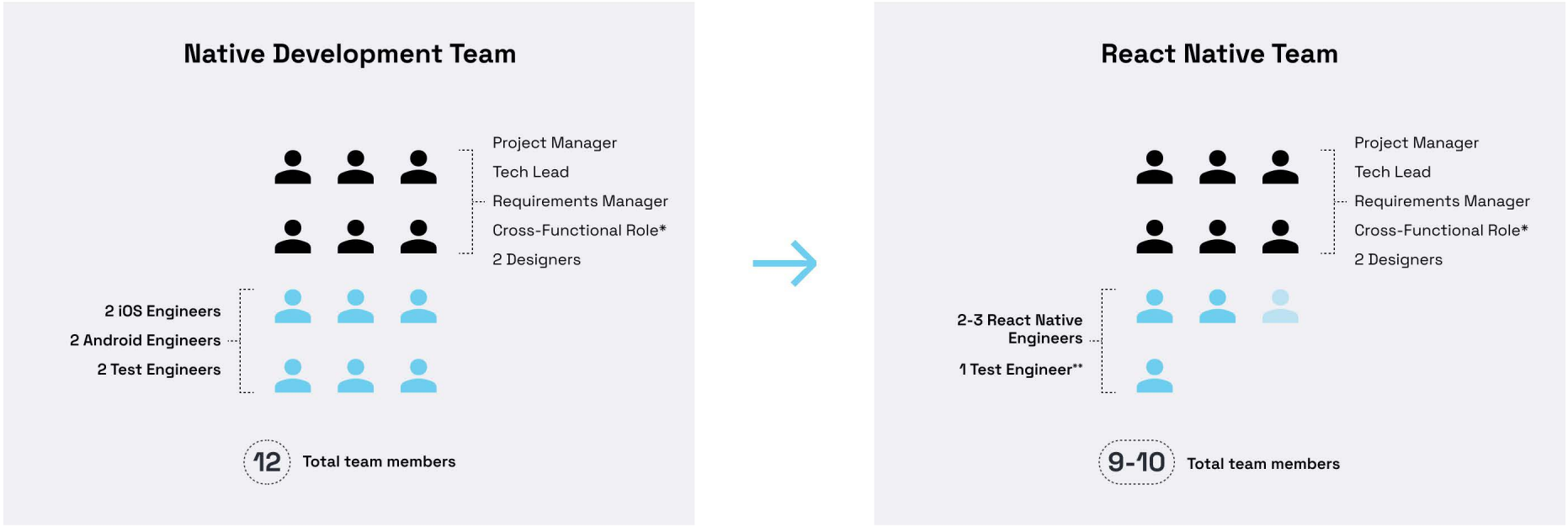
- 80% of the code on iOS (or Android) is written exclusively with React Native
- 20% of the features require using iOS or Android native code (e.g., platform-specific UI, widgets, etc.)

40% reduction in development effort*

*The likely range is **30-50% reduction**, depending on the extent of code-sharing. More details and calculations appear on the following pages.

But, development effort is only one part of the overall equation.

Remember: producing a mobile application takes more than developer talent. At a digital product consultancy like WillowTree, our clients ask us to own their entire strategy, which requires dedicated team members with a variety of expertise: design, quality assurance, user testing, and project management. For WillowTree and many other agencies, **a complete mobile product development team** might look like the breakdown below. Note that this structure isn't unique to WillowTree; we encourage any agency or product company to consider efficiency in these more holistic terms:



12 Team Members → 9-10 Team Members = 15-25% Cost Savings

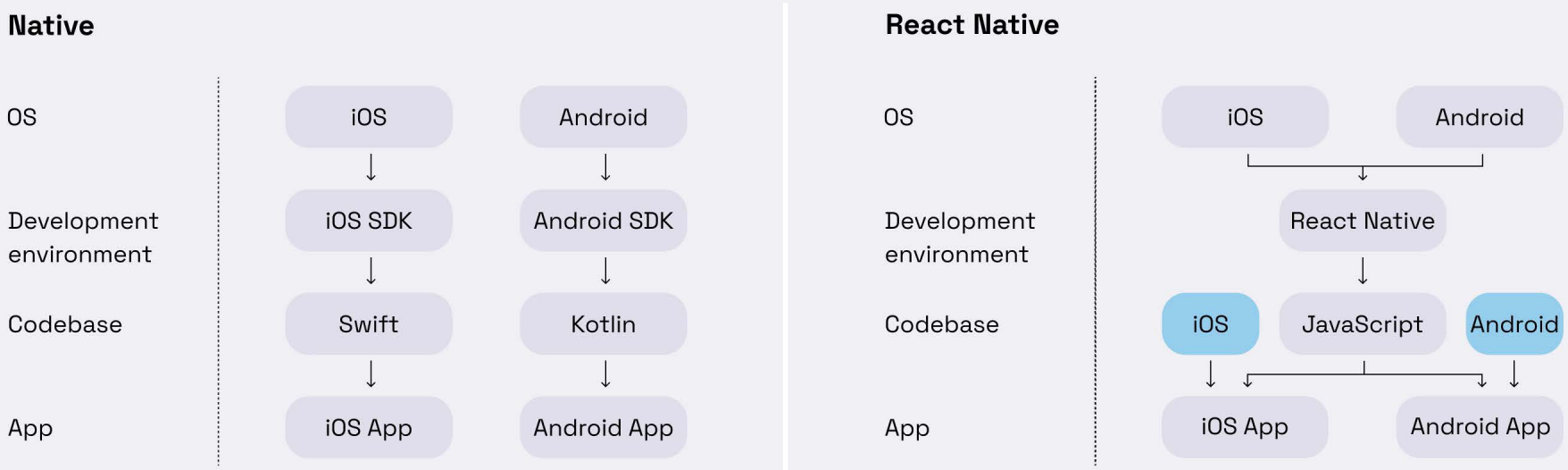
* This cross-functional role could incorporate support from analytics, user testing, and security

** Additional test engineer roles should be considered when considering new target platforms.

Wait, so do we still need native developers?

Yes. The reason we still need some native talent on the team is that almost all apps require some essential integration with platform-specific tools. Build systems, dependencies, SDKs, and the app store release process will still be platform-specific, so a working knowledge of iOS and Android is a must-have. Plus, there will always be bugs to investigate, so it helps to understand what's going on "under the hood."

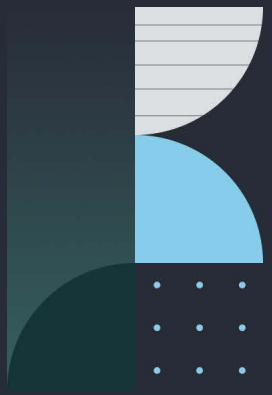
WITH REACT NATIVE, JAVASCRIPT PLUS SOME NATIVE CODE PROVIDES THE FOUNDATION FOR TWO DIFFERENT APPS



An important note: When staffing React Native teams, intentionally include developers who have iOS and/or Android experience as well!

The bottom line.

In short, depending on the complexity of the mobile product, React Native requires about **2 to 3** fewer team members on a 10- to 14-person Agile product team. The math typically works out to saving approximately **30-50%** in developer effort and **15-25%** in total associated costs.

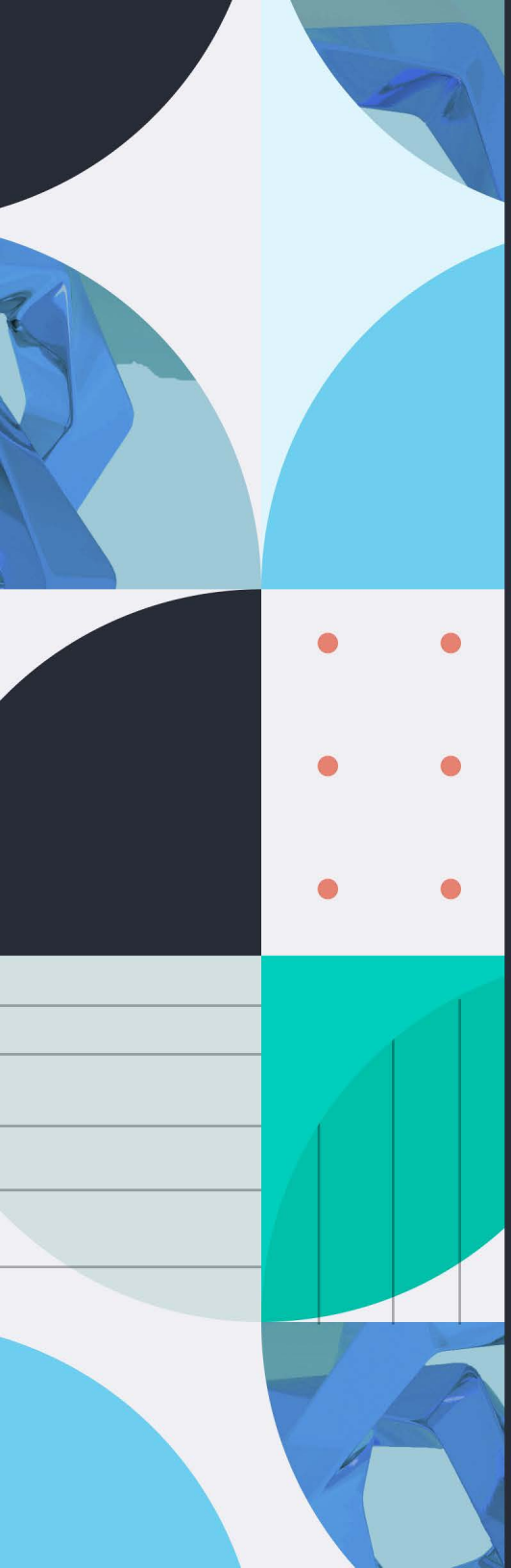


REACT NATIVE REQUIRES FAR LESS NATIVE CODE

"One of the trade-offs you make when you go the React Native route – where you only have one codebase – is that you still have to do a certain amount of Android- and iOS-specific coding; it's just a question of it being much less. The cross-platform framework provides many modules, but you still have to bolt on some percentage of platform-specific code to make sure it operates properly on both iOS and Android."

Insights From WillowTree Experts

Stephanie Cure
Staff Software Engineer



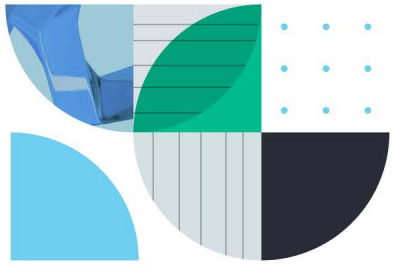
Part 6

Working with WillowTree to Build What's Next in React Native

The final word? Not hardly.

Our clients trust us to recommend tooling that is capable and stable for the long run, which is why we recommend React Native. That doesn't mean we *won't* use other tools (e.g., Xamarin for .NET clients, or Flutter for more risk-tolerant, innovation-forward clients), but all things being equal, React Native is our pick.

As we continue to dive deeper on React Native and other topics, stay tuned for more articles and insights from the team here at WillowTree, or reach our directly to hear our latest thinking on all things cross-platform.



Technology changes daily, however, and our perspectives are always evolving.

Ending where we began:

If you're going cross-platform, React Native is best-in-class.

Our work comes down to improving experiences. That means thinking beyond the end user's experience and considering the experiences of our clients and developer teams in how we collaboratively create and maintain these products.

Ultimately, that's why we stand behind React Native. While this leading cross-platform framework may not offer all the bells and whistles of native development, we're confident React Native can save our clients time and money – in both the upfront build and future-state maintenance of a digital product – while still delivering a premium, consistent user experience across iOS and Android devices.

Let's talk: React Native and your company's next digital product.

↓

Is React Native right for you? Let's find out. Our experts are ready to work with you to determine the optimal development framework for your digital product.

Get in touch, and let's get started.

info@willowtreeapps.com
1-888-329-9875

Sources.

1. TechAhead: The history of React Native: Facebook's Open Source App Development Framework
2. Statcounter: Mobile Operating System Market Share in United States of America - January 2023
3. Statistica: Mobile operating systems' market share worldwide from 1st quarter 2009 to 4th quarter 2022
4. AppBrain: Android app frameworks
5. Hongkiat: A Guide to Progressive Web Applications
6. Designveloper: 6 Best Progressive Web Apps Examples (PWAs) over the Past Decade
7. React Native: Who is Using React Native
8. Verified Market Research: Top 5 cross-platform and mobile advertising companies offering ways to attract customers
9. Stack Overflow: 2021 Developer Survey