Cross-Platform Development

What does Cross-Platform Development mean?

Cross-platform development is the practice of developing software products or services for multiple platforms or software environments. Engineers and developers use various methods to accommodate different operating systems or environments for one application or product.

The pros & cons of cross-platform mobile app development

The advantages of building a cross-platform application:

· Smaller Teams, native development requires two separate teams for Android and iOS. It can hinder communication between developers and, accordingly, slow down the development. If you choose a Cross-Platform Development tool you will only need the developer to know one language to write to both platforms. Obviously, apps with more native features will require help from native developers as well. Notwithstanding this fact, in most cases, the team will be smaller in size, and thus, easier to manage.

· **Shorter development time, as you will not need to develop your application separately for each platform for your application.**

· **Cost-effectiveness.** Building a native mobile application is very expensive, usally costing around tens of thousands of dollars. Multiply the cost by two (iOS and Android) and add 30% (Android development is more expensive), and you’ll get the approximate cost of launching an application on both the App Store and Google Play.

· **Updates synchronization.** In a world where successful app publishers roll out updates up to 4 times per months, maintenance costs can consume a great part of the entire app revenue — and that’s where cross-platform development wins.

The cons of Cross-platform Development:

· **Performance issues.**Smartphone’s computing power is relatively small. The rendering of heavy HTML5/CSS UI components, on the other hand, takes a lot of GPU/CPU resources and may increase an app’s response time.

· **UX issues.** Meeting both platforms UX requirements can be a challenge. Apple is particularly notorious about its Human Interface Guidelines and turns down mobile websites wrapped in native containers.

There are two well known tools for Cross-Platform Development that will be reviewed in this report, React Native and Xamarin.

What Is React Native?

React Native is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android. It’s based on React, Facebook’s JavaScript library for building user interfaces, but instead of targeting the browser, it targets mobile platforms. In other words web developers can now write mobile applications that look and feel truly “native” all from the comfort of a JavaScript library that we already know. Similar to React for the Web, React Native applications are written using a mixture of JavaScript and XML-esque markup, known as JSX. Then, under the hood, the React Native “bridge” invokes the native rendering APIs in Objective-C (for iOS) or Java (for Android). Thus, your application will render using real mobile UI components, not webviews, and will look and feel like any other mobile application.

React Native Advantages

Faster to Build

The major selling point of React Native is shorter development time. It provides numerous ready-to-apply components that can accelerate the process.

Hot Reloading

Thanks to hot reloading, a developer can keep the app running while implementing new versions and tweaking the UI. It makes changes in the app instantly visible without the need for the developer to save them. It is a major benefit for developers. Not only do they save time on the compilation, but also they don’t lose any state of an app while making changes.

Fast Applications

Many argue that React Native code might hinder app performance. JavaScript won’t be as fast as native code, but in most cases, you won’t see the difference. In the case of more complex applications, the framework might be less effective, but you can always transfer some code to a native module, and it is no longer an issue.

React Native Drawbacks

The largest drawback is probably React Native’s maturity, as the project is still relatively young. iOS support was released in March 2015, and Android support was released in September 2015. The documentation certainly has room for improvement, and continues to evolve. Some features on iOS and Android still aren’t supported, and the community is still discovering best practices.

Less Smooth Navigation

React Native still lacks navigation components to provide users with seamless UX. There is no ideal solution in RN for navigation between displays. It will be getting better and better, but probably it will never be as good and smooth as native navigation.

Native Developers Still Needed

Implementing some native features and modules necessitates detailed knowledge of a particular platform. React Native does offer custom modules that you can refactor across operating systems, but things such as access to device sensors, camera or push-notifications require help from iOS and Android developers.

Facebook Rules

Facebook puts a lot of resources to ensure React Native’s reliability and also to guarantee a constant development of the platform. It is unlikely that Facebook kills the framework overnight. But you still operate on Facebook’s license and, at the end of the day, you’re always dependent on them. Finally, Facebook has a right to revoke the licence (BSD) to use React and React Native if you get into a dispute with them

What Is Xamarin?

The platform was built by the developers behind [Mono](http://www.mono-project.com/), an open source development platform based on the .NET Framework, led by Miguel de Icaza and first introduced in 2001. The Xamarin company was founded on May 16, 2011. Xamarin was created as a commercial project until the company was acquired by Microsoft in 2016. Xamarin became a popular cross-platform product for developing mobile apps within the Microsoft ecosystem. As Microsoft made Xamarin SDK open-source, it became part of Xamarin Visual Studio Integrated Development Environment.

Xamarin uses a single language, C#, to create apps for all mobile platforms. C# is a mature language with strong safety-typing that prevents code from unexpected behavior. As C# is one of the .NET framework languages, it can be used with a number of useful .NET features such as Lambdas, LINQ, and Asynchronous programm­­­­ing. Technically speaking, Xamarin uses C# and native libraries wrapped in the .Net layer for cross-platform app development. Such applications are often compared to native for both iOS and Android ­­­­experience. Additionally, Xamarin can leverage all native and the latest API access to utilize underlying platform capabilities in Xamarin apps such as [ARKit](https://developer.xamarin.com/api/namespace/ARKit/) on iOS or [Android Multi-Window](https://blog.xamarin.com/optimizing-android-apps-for-multi-window-mode/).

**Compilation**

The platform has two major products: Xamarin.iOS and Xamarin.Android. In the case of iOS, the source code is compiled directly into native ARM assembly code (Ahead-of-Time compilation), while Android Xamarin apps are first compiled down to Intermediate Language and then – into native assembly code at runtime (Just-in-Time compilation).

**Xamarin project structure**

Most of the Xamarin-related work is expected to be run via a Windows development computer with Visual Studio and Xamarin installed. The apps can be debugged straight from the desktop or on devices and emulators. If you plan to develop iOS apps on Windows, it’s also possible as Visual Studio connects to the iOS storyboard designer and iOS simulator. There’s also Visual Studio for Mac which allows for running a simulator on the Mac or directly on a tethered iPhone. Debugging is run via Visual Studio.

Pros of Using Xamarin for Development

Performance Close to Native

Unlike traditional hybrid solutions, based on the web technologies, a cross-platform app built with Xamarin can still be classified as native. The performance metrics are comparable to those of Java for Android and Objective-C or Swift for [native iOS app development](https://www.altexsoft.com/blog/mobile/ios-app-development-checklist-27-items-to-consider-before-starting-your-next-project/). Visual Studio offers a complete solution for building, testing, and tracking the app’s performance: [Visual Studio App Center](https://appcenter.ms/) allows you to run automated UI tests and identify performance issues before the release. However, this service is provided at an additional fee.

Native User Experiences

Xamarin allows you to create flawless experiences using platform-specific UI elements. Simple cross-platform apps for iOS, Android or Windows are built using Xamarin.Forms tool, which converts app UI components into the platform-specific interface elements at runtime. Yet, there might be a slight decline in performance due to the extra abstraction layer. For custom app UI and higher performance you can still use Xamarin.iOS and Xamarin.Android separately to ensure excellent results.

Open Source Technology with Strong Corporate Support

After the company was acquired by Microsoft in February 2016, Xamarin licensing policies underwent radical changes. Xamarin SDK, including runtime, libraries, and command line tools, has become open source and available for all under the MIT license as a part of Visual Studio. By eliminating the major barrier, the cost of license, Microsoft paved the way for the further growth of the platform.

Xamarin Drawbacks

Limited Access to Open Source Libraries

Native development makes extensive use of open source technologies. With Xamarin, you have to use only the [components provided by the platform](https://components.xamarin.com/) and some .Net open source resources.

Basic Knowledge of Native Languages Required

When using Xamarin.iOS and Xamarin.Android to build mobile apps with truly native look and feel, you will still need to write a platform-specific layer of code. Thus, at least a basic knowledge of native technologies (Java/Kotlin for Android and Objective-C/Swift for iOS) is required.

Not Suitable for Apps with Heavy Graphics

Xamarin’s main benefit is the ability to share your code across the platforms. Yet, you can only share the logic, UI code will be mostly platform-specific. This makes building games, rich custom UI, or complex animations in Xamarin pretty pointless. With a small amount of shared code, Xamarin becomes unviable option.

Larger App Size

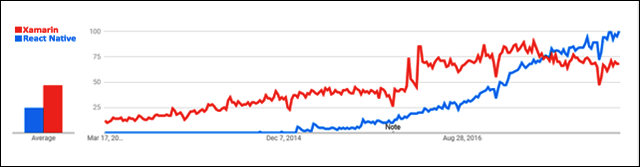
Depending on their type and complexity, Xamarin apps are typically larger than native ones. A simple “hello, world” app for Android might take up to 16 MB, much of it being used by the associated libraries, content, Mono runtime, and Base Class Library (BCL) assemblies.



React Native and Xamarin Comparison

Market Share

**React Native**: The trend for using React Native has improved considerably. Thousands of websites use this tool, Facebook, Skype, Facebook Ads Manager, Tesla and all use this tool.



**Xamarin** :Acquired by Microsoft, and a part of its Visual Studio development, Xamarin has been the technology choice for 15,000+ companies. Apart from developing cross-platform apps, it lets the developers reuse code, supports the test ecosystem and debug products.

Availability

**React Native**: it’s a free tool, and you can build for both Android and [IOS platforms](https://www.cabotsolutions.com/ios-app-development/).

**Xamarin**: There is a free version, but you need to spend for an enterprise level application.

Compilation

**React Native**: JIT (Just in Time) compilation is not possible while developing iOS applications, so React Native falls back to interpreting JavaScript code. And on an internal level, it uses the iOS provided JavaScriptCore. JavaScriptCore is also used on Android, but there’s nothing preventing JIT from being used on Android.

**Xamarin:** The language, C# makes both JIT compilation and AOT (Ahead of Time) compilation possible. However, as JIT compilation is not possible with iOS, Xamarin AOT-compiles the application.

Development Environment

**React Native**: It allows the developers to choose an IDE they are comfortable with.

**Xamarin**: More powerful, because, you can write a code for iPhone app on Windows and compile it for Mac. However, you need a different platform that would be required for the build.

Ready Made Components

**React Native**: The components can all be found in place , with excellent documentation.

**Xamarin**: The components are split between NuGet and component store, but the documentation part could be improved.

Community and Assistance

**React Native**: It is a younger tool, and the developer support is there, but much smaller. And often, the depth of knowledge is small.

**Xamarin**: Great deal of assistance can be found in websites like Stack Overflow, blogs and community support forums.

References:

<https://www.netguru.co/blog/react-native-pros-and-cons>

<https://www.oreilly.com/library/view/learning-react-native/9781491929049>

<https://hackernoon.com/xamarin-vs-react-native-a-comparison-study-5ebea64cd801>

<https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/#pros>

<https://www.techopedia.com/definition/30026/cross-platform-development>

<https://medium.com/all-technology-feeds/cross-platform-vs-native-mobile-app-development-choosing-the-right-dev-tools-for-your-app-project-47d0abafee81>