**React Native**

**Pros:**

1. **One code base for all the platforms**

“React Native isn’t perfect but the benefits of a single codebase are huge,” this is what the majority of companies opting for RN admit.

1. **Ready-to-use native components: great development speed**

React Native offers a set of ready-to-apply [components](https://reactnative.dev/docs/components-and-apis#basic-components) to save you time on designing common blocks from scratch. But the Facebook team doesn’t limit you to the core collection.

1. **Ease of UI Design**

The React Native framework makes laying out User Interfaces a breeze. Developers simply use CSS-like language to create each UI view, just as they would if they developed the view in a native app environment.

1. **Fast Refresh and Flipper: better debugging experience**

Not long ago, debugging was a sore spot with RN developers. But after introducing the [Fast Refresh](https://reactnative.dev/blog/2019/09/18/version-0.61) feature and out-of-the box support for mobile app debugger [Flipper](https://fbflipper.com/).

1. **Live Reload Feature**

React Native eliminates the long recompiling process of other development platforms by offering a feature called Live Reload. This allows developers to see the results of their coding immediately and make quick changes with the click of a button.

1. **A large community of developers**

who are contributing day in, day out

1. **Expo**

It’s a great tool for the fast development of apps for many platforms, with many ready-to-go libraries in the SDK.

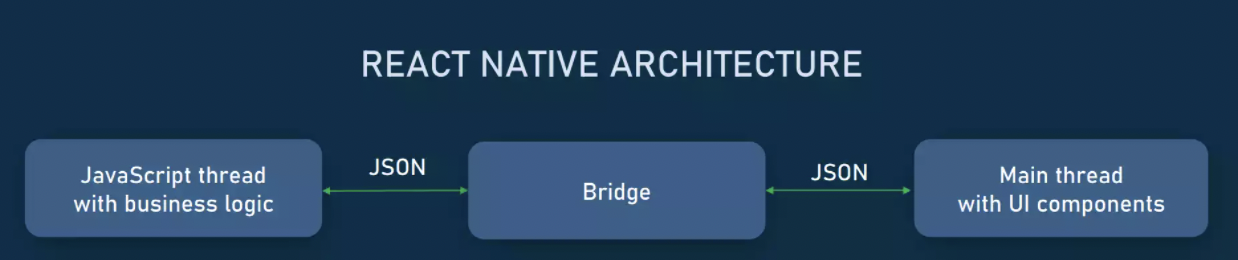
1. **Compatible with third-party plugins**

Does not require high memory to process. No specific web views functions are required and native modules are linked with the plugin through the framework. Smoother running and faster loading are key features for it.

**Cons:**

1. **Performance issues**

While RN apps are fast enough, they still can’t beat the performance of native solutions. The bottleneck lies in the React Native architecture itself.



A React Native app combines JavaScript and native code parts, which are executed in two separate threads.

1. **The JavaScript thread**

It runs the app’s business logic. It dictates which APIs to call, which views to display, and how to do it.

1. **The UI or main thread**

It manipulates UI components, renders views, and “listens” to user actions — like taps, swipes, and so on.

1. **Android Iterations Can Be Buggy**

While React Native does not require separate development for iOS and Android, it’s not flawless. Once the iterations are split out, the Android and iOS versions can often run differently than its counterpart.

These issues are easily surmounted, but you may want to build the appropriate testing time into your development schedule to ensure your app is ready for its Android launch. This is an important consideration when weighing the pros of using React Native and React Native disadvantages.

Reference links:

<https://www.altexsoft.com/blog/react-native-pros-and-cons/>

<https://procoders.tech/blog/react-native-pros-and-cons/>

<https://www.toobler.com/blog/react-native-pros-and-cons>

**Cons for Web:**

1. **Limited web API support**

Not all APIs that are available in React Native are available in React Native for Web. It means to re-implement the functionality or simply not support it at all when on the web.

1. **Styling issues**

In terms of styling, React Native for Web inherits all the disadvantages of React Native. In particular, styling in React Native is very limited. Among the missing capabilities are:

* Styling child components
* Selecting sibling elements
* Using pseudo-classes like :hover or :focus

1. **Missing core components**

Not all core components available in React Native are available with React Native Web. Sometimes a third party developer will create a separate, web version of the component, for example [React Native Web Refresh Control](https://www.npmjs.com/package/react-native-web-refresh-control?ref=retool.com).

1. **Compatibility**

A number of popular React Native libraries do not have proper support in React Native for Web.

Reference link:

[**https://retool.com/blog/how-to-make-your-react-native-apps-work-on-the-web/#:%7E:text=React%20Native%20for%20Web%20is%20a%20good%20option%20for%20those,app%20does%2C%20and%20no%20more**](https://retool.com/blog/how-to-make-your-react-native-apps-work-on-the-web/#:%7E:text=React%20Native%20for%20Web%20is%20a%20good%20option%20for%20those,app%20does%2C%20and%20no%20more)

**Flutter**

**Pros:**

1. **Hot Reload**

Hot Reload strengthens the bond between developers and designers when they are looking for improvements on how the app looks and check effects immediately.

1. **High Performance**

There’re many factors that impact the performance of an app, including CPU usage, frame number per second, request number per second, average response time, and many more. The rate of Flutter is 60fps, at which contemporary screens display a smooth and clear picture.

With this frame rate, a human eye can identify any lag. **If you compare it with React Native and Xamarin, this framework is ahead with 220 millisecond launch time and 58fps**.

1. **Immediate Updates**

Flutter offers hot reload functionality that allows you to instant updates without the need for plugins. A hot reload also allows you to view updates in real time. If you face an error while running the code, the framework lets you fix it immediately and carry on without having to restart it.

1. **Custom Widgets for Quick UI Coding**

Flutter has ready-designed and custom widgets. The widgets of Flutter are consistent and have extensive capabilities.

1. **Single Code for any device**

Flutter is a single codebase framework, which means one code can be used in IOS, Android, or web apps as you need. This might sound like just another cross-platform framework, but just until you try the extremely easy process and beautiful results that flutter offers.

1. **Quick Development**

In the competition that is going in the digital space, the quicker is the better. Choosing Flutter to build your apps can help you save hours more than any standard native application.

1. **Own Rendering Engine**

The problem with some cross-platform solutions is that they are very similar on iOS and Android. This is why Flutter is the best option, because it consists of packages that contain a set of unique widgets for both operating systems.

**Cons:**

1. **Large File Sizes**

One big loophole that cannot be ignored is the large file size of apps developed in Flutter. Now in some cases, these file sizes could be a significant issue and cause a developer to choose an alternative tool for the development

1. **Limited number of third-party libraries**

Flutter being relatively cannot be compared to native programming languages. Developers still need to spend more time building as many libraries as possible.

Reference links:

<https://waverleysoftware.com/blog/why-use-flutter-pros-and-cons/>

<https://theonetechnologies.com/blog/post/flutter-mobile-application-development>

<https://nyl.technology/blog/mobile-app-development/advantages-disadvantages-flutter-mobile-app-development/>