20 May, 2017

The Good and the Bad of ReactJS and React Native

Share:

ReactJS and React Native are the new technologies for web and mobile development introduced by Facebook. The project was started by Jordan Walke, a Facebook software engineer, in 2011. To simplify the development process and foster a more comfortable user experience, he decided to create a library that would allow for building a web interface with JavaScript.

In this article, we'll address why React was created and the advantages and disadvantages of using React technologies.

What is React? A brief story behind the library

Prior to developing ReactJS, Facebook was confronted with a major user experience task – building a dynamic UI with high performance. For instance, the engineers wanted to make news feed updates happen simultaneously with people using chat.

To achieve that, Facebook had to optimize the development process itself and Jordan Walke decided to do it with JavaScript. He proposed putting XHP, the Facebook markup syntax, into the JS coordinate system. The idea seemed impossible, but in 2011 his team released the ReactJS library on the basis of JavaScript and XHP symbiosis. Then Facebook realized that ReactJS was working faster than any other implementation of its kind.

ReactJS is a JavaScript library that combines the speed of JavaScript and uses a new way of rendering webpages, making them highly dynamic and responsive to user input. The product significantly changed the Facebook approach to development. After the library was released as an open-source

tool in 2013, it became extremely popular due to its revolutionary approach to programming user interfaces.

Two years later, the same group of engineers released **React Native,** a hybrid mobile-app development framework for iOS and Android**.** The tool was based on the very same principles as ReactJS and shortly was accepted by the engineering community and companies that adhere to the mobile-first strategy.

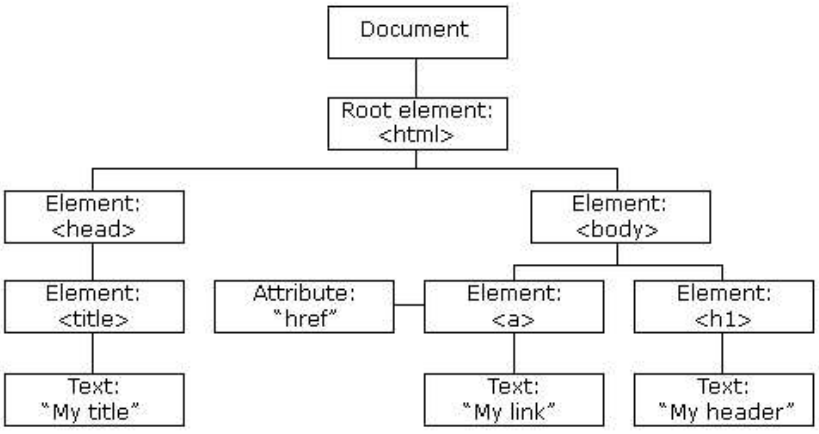
React Native combines native application development with JavaScript UI development. While computationally heavy features can be implemented with native modules for iOS and Android, the rest of the code can be written with JavaScript and shared across platforms. Unlike PhoneGap and Cordova cross-platform tools, React Native doesn't use WebView, a mobile engine that wraps code. Instead, it provides access to native mobile controllers which allows for achieving native look and feel to apps.

So, let's take a closer look at ReactJS' pros and cons as the main product. Then we'll explore how React Native is different and which benefits and drawbacks it inherits from ReactJS.

Pros of ReactJS

1. Virtual DOM in ReactJS makes user experience better and developer's work faster

DOM (document object model) is a logical structure of documents in HTML, XHTML, or XML formats. Describing it in layman's terms, it is a viewing agreement on data inputs and outputs, which has a tree form. Web browsers are using layout engines to transform or parse the representation HTML-syntax into document object model, which we can see in browsers.



The HTML DOM tree of objects
Source: W3Schools

The main concern about traditional DOM construct is the way it processes changes, i.e., user inputs, queries, and so on. A server constantly checks the difference caused by these changes to give the

necessary response. To respond properly, it also needs to update the DOM trees of the whole document, which is not ergonomically valid because DOM trees are fairly large today, containing thousands of elements.

The team behind React managed to increase the speed of updates by using virtual DOM. Unlike other frameworks that work with the Real DOM, ReactJS uses its abstract copy – the Virtual DOM. It updates even minimalistic changes applied by the user, but doesn't affect other parts of interface. That is also possible thanks to React's components isolation, which we'll get to in a minute, and a special data structure in the library.

Real and Virtual DOMs

This makes updates really quick, allowing for the building of a highly dynamic UI. We can notice it writing in Facebook chat and seeing a simultaneously updating news feed. Moreover, in ReactJS, developers don't have to bind DOM to functionality in the front-end because React elements are already connected to it.

The approach enabled developers to work with UI-objects faster and use hot reloading (applying changes in a real-time mode). Not only did it increase performance, it also made programming faster.

2. ReactJS permits reusing code components – significantly saving time

Another advantage that Facebook introduced with React is the ability to reuse code components of a different level anytime, another meaningful time-saving effect.

Think of designers. They constantly reuse the same assets. If they didn't, they'd have to draw corporate logos, for instance, over and over again. It's pretty obvious: Reusing is a design efficiency. In programming, this process is a bit more difficult. System upgrades often turn into a headache as every change can affect the work of other components in the system.

Managing updates is easy for developers in ReactJS because all components are isolated and change in one doesn't affect others. This allows for reusing components that do not produce changes in and of themselves to make programming more precise, ergonomic, and comfortable for developers. Check this guide by Alex Grigoryan from WalmartLabs on how to achieve efficient code reusability with ReactJS.

3. One-direction data flow in ReactJS provides a stable code

ReactJS allows for direct work with components and uses **downward data binding** to ensure that changes of child structures don't affect their parents. That makes code stable.

Most complex view-model systems of JS-representation have a significant but understandable disadvantage – the structure of data flow. In the view-model system, child elements may affect the parent if changed. Facebook removed these issues in React JS, making it just the view system.

Instead of using explicit data binding, ReactJS uses one direction – downward – data flow. In such a structure, child elements cannot affect parent data. To change an object, all a developer needs to do is modify its state and apply updates. Correspondingly, only allowed components will be upgraded.

4. An open-source library: constantly developing and open to contributions

ReactJS was one of the first JavaScript-connected projects released as open source by Facebook. That means that ReactJS uses all advantages of free access – a lot of useful applications and additional tools from off-company developers. Facebook's Pete Hunt says that at least two main features – batching and pruning – were created by developers that noticed the project on GitHub. Now ReactJS is 7th in Trending on GitHub with over 67,000 stars. And, there are more than 1000 open-source contributors working with the library.

Cons of ReactJS

1. High pace of development

This disadvantage are aptly described by developers Michael Jackson and Ryan Florence on Modern Web: "In case you didn't notice we're driving a car here with two flat tires, the hood just flew up in front of the windshield, and we have no clue what's going on anymore!" The environment constantly changes, and developers must regularly relearn the new ways of doing things. Everything is evolving, and some developers are not comfortable with keeping up with such a pace.

2. Poor documentation

The problem with documentation traces back to constant releases of new tools. Different and new libraries like Redux and Reflux are promising to accelerate the work of a library or improve the entire React ecosystem. At the end, developers struggle with integrating these tools with ReactJS. Some members of the community think that React technologies are updating and accelerating so fast that there is no time to write proper instruction. To solve this, developers write their own documentation for specific tools used by them in current projects.

3. 'HTML in my JavaScript!' – JSX as a barrier

ReactJS uses JSX. It's a syntax extension, which allows mixing HTML with JavaScript. JSX has its own benefits (for instance, protecting code from injections), but some members of the development

community consider JSX to be a serious disadvantage. Developers and designers complain about JSX's complexity and consequent steep learning curve.

ReactJS and React Native: what the difference?

Two years after the 2015 ReactJS release, Facebook created React Native. While the ReactJS library is developed for creating web interfaces, React Native is a hybrid app-development framework for iOS and Android. The most promising thing about React Native is that it inherits ReactJS principles and syntax.

All technical differences between them are caused by platform aims.

- While ReactJS uses Virtual DOM to render browser code, React Native uses native APIs as a bridge to render components on mobile. For example, for Android components, it uses Java APIs and it invokes Objective-C API to render to iOS.
- React Native doesn't use HTML. So, if you worked with ReactJS before, you'll have to get familiar with React Native syntax. For example, it uses <Text> instead of <p> and <View> instead of .
- Because React Native doesn't use CSS, standard CSS-features like animation run with React Native special APIs.

Pros of React Native

Right now, React Native framework is one of the fastest and most efficient environments for mobile app development. It's analogous to ReactJS, and here's what you should know in terms of mobile:

1. React Native uses JavaScript – fast and popular programming language

Let's reiterate – JavaScript remains one of the fastest and widely-used programming languages. As we discussed in our article on pros and cons of full-stack JavaScript development, 55.4 percent of developers use JavaScript according to annual Stack Overflow survey. The maturity of the JS community allows specialists to quickly learn the language and constantly progress working with it. So, you should typically be able to quickly find a React Native developer for your project.

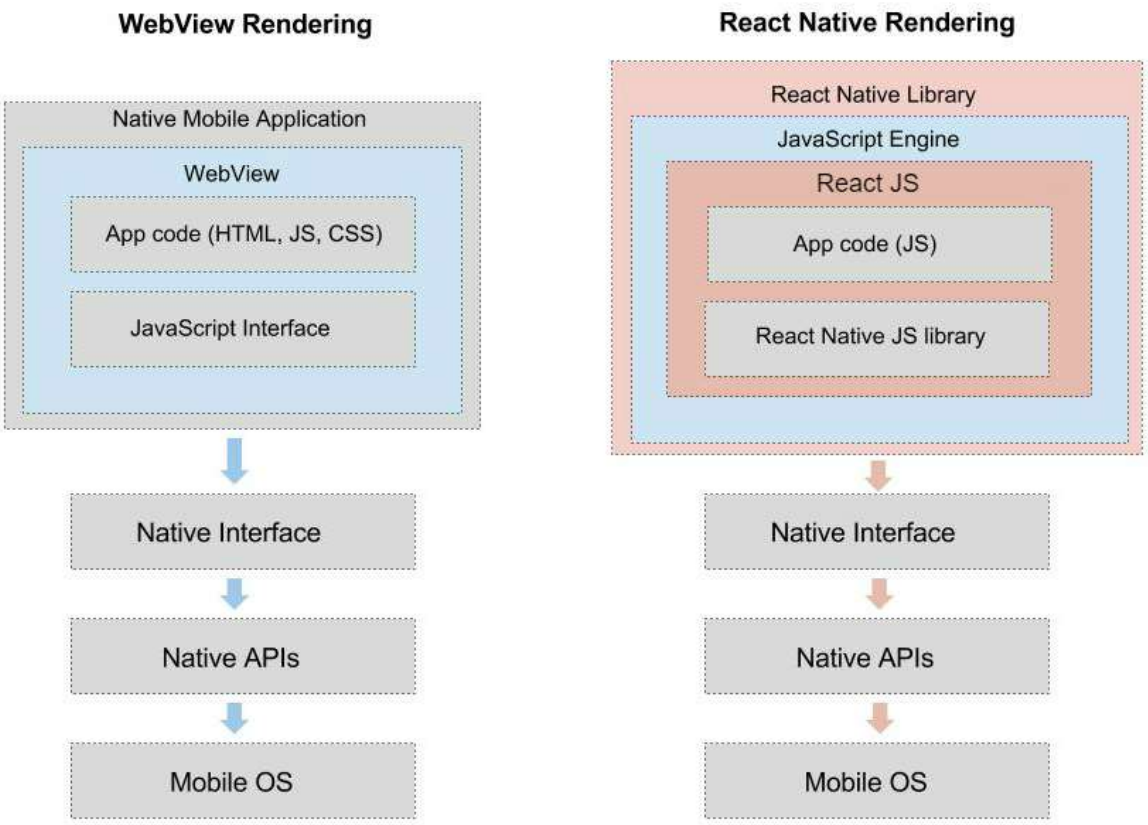
Besides, React Native leverages and combines the main advantages of JavaScript and ReactJS. Due to the prevalence of JS code, engineers can work faster and more efficiently because they don't have to restart a developed app after each update to see changes; they can simply refresh the viewing page.

In some cases, after-launch updates can be done faster. For instance, Apple allows JavaScript-based changes of application behavior in real-time mode with no review cycle required.

2. Native controls and native modules in React Native improve performance

React Native renders some code components with native APIs, unlike other cross-platform frameworks such as PhoneGap, which render code via WebView, a mobile engine. While the WebView

approach greatly reduces performance, React Native communicates with targeted components for iOS or Android and renders code to native APIs directly and independently. Doing that, React uses a separate thread from UI, which also increases the performance score.



WebView Rendering vs React Native Rendering

Native modules perform another important role. Currently, React provides a set of native modules written in Objective-C and Java out-of-the-box. These modules can't be reused across two platforms but they aim at higher performance in computationally heavy operations like image editing or video playback. Basically, your team can apply the existing modules or, if they have Java and Objective-C experience, write custom modules themselves. As the React community grows, some things are already available. The rest of the codebase can be reused. For instance, Facebook Ads Manager shares 87 percent of code across Android and iOS.

On top of that, React Native provides simple debugging and error messaging tools. For instance, similar to web-programming, specialists can use Chrome or Safari developer tools, both of which they're usually familiar with.

3. React Native contains all ReactJS features, aimed at improving UI

React Native uses ReactJS as the JavaScript library, so it has all its advantages. To build a cross-platform app with React Native, developers don't need to know the language of the native platform. They must only be proficient in JavaScript and familiar with the React syntax. But, as mentioned, they can easily add native components to code as well.

Downward data flow is also preserved, so core components can be edited without an influence on child components. This makes UI development smooth while positively impacting user experience. The apps look and feel native.

Cons of React Native

1. Documentation

Unfortunately, React Native inherits the main ReactJS disadvantage. The community is young so the available documentation is poor, especially for integration with additional tools.

2. Expertise for native modules

We've discussed the native modules in the pros section. They definitely bring flexibility to the framework by filling the missing performance links. If you need to handle computationally heavy operations, you can inject native modules and achieve a truly native feel to your app.

However, to some extent, it could negate the point of cross-platform development as you still need some native engineers (Objective-C, Java, or both) in reserve. So maybe you don't need React Native to build the next mobile Photoshop. But, if you aren't aiming at brute-force demanding tasks, JS expertise is enough.

3. Third-party components

That said, React Native has a number of native modules for iOS and Android out-of-the-box, but the number of third-party components is still limited. And we can't know for sure that community-built modules will be supported by the next releases of the framework. It remains a disadvantage, because React Native doesn't offer a wide spectrum of possible features that developers may want to implement in their apps.

4. Lagging SDK Updates

React Native often drags its feet when iOS or Android updates their SDKs. React Native's team should integrate a code library with new software. And despite the fact they work pretty fast, they cannot update every part of the APIs at once. That's why the full synchronization between React Native and new SDKs often takes too long.

ReactJS and React Native Use Cases

Companies who've already adopted ReactJS or React Native did so for different reasons. Here are the most notable adopters besides Facebook itself.

Instagram. The Instagram team wanted to build a website, a one-page app, to let web visitors also access the social platform. And ReactJS seemed the best fit for this purpose. The web app was optimized and appeared to be quick and comfortable for users. Now, both mobile and web Instagram apps are built with React.

Netflix. Netflix, which adopted ReactJS in 2015, is now using it together with Gibbon, a rendering layer. Back then, Netflix chose React due to its one-way-ticket model of data flow and declarative approach to programming.

Airbnb. The company decided to switch to ReactJS due to its component reusability, very simple code refactoring, and iterating. Now it's used in internal structures of the company's mobile app and webpage.

Bloomberg. The online magazine chose React Native when developing its mobile app. Their post on the Tech at Bloomberg blog says that it is "the first tool that truly delivers on the promise of cross-platform native app development."

In terms of performance, the React platform is really progressive considering how many companies have adopted it. While other frameworks improve UI in terms of software performance, the young and sometimes messy React Native aims to change the way apps communicate with software and hardware. ReactJS, on the other hand, from a simple troubleshooting idea has been transformed into a solution that can significantly optimize efforts of web development and increase its efficiency.

Just recently Facebook announced the release of React v.16 under codename React Fiber. The whole library was completely rewritten from scratch and, as developers say, saved everything innovative in the system and added new. You can check if Fiber is ready and help developers complete unit tests. You can also visit the library page on GitHub.

This article is part of our "The Good and the Bad" series. If you want to know more about cross-platform mobile development, take a look at our previous publication on the pros and cons of using Node.js.

2 Comments

The Good and The Bad of Xamarin Mobile Development