A Whitepaper on React Native

A whitepaper on React Native

Rajan Twanabashu

Copyright Information

Table of Content

1 Introduction to React Native

React Native is the next generation of React - a JavaScript code library developed by Facebook and Instagram, which was released on Github in 2013. React Native let you build native iOS and Android application by using JavaScript. So react native utilize both the advantages of Native app and Hybrid app functionality.

React Native helps developers reuse code across the web and on mobile. Engineers won't have to build the same app for iOS and for Android from scratch - reusing the code across each operating system. Thus the focus of React Native is on developer efficiency across all the platform. They even uses the phrase 'Learn Once, write any where', which is absolutely true.

More information on react native can be found at:
https://facebook.github.io/**react-native**/

2 What's wrong with traditional approach of mobile application development

1. No portability

Since each native application only runs on one platform, businesses building native apps must make a choice–build for one platform or build for multiple platforms? Unfortunately, there's no easy answer. Support of all the platform requires significant amount of time and resource.

2. Platform instability

The mobile platform is notoriously unstable. A popular platform today may disappear in just a few years. For example, both Blackberry and Palm dominated the mobile industry just 5 short years ago. Today, Blackberry is struggling and Palm doesn't exist. The fact is, nobody knows what the mobile platform landscape will look like in another 5 years. Companies that choose the native approach always run the risk of wasting time and money building for a platform that might not last.

3. Development cost

While native app development cost varies depending on the app's complexity, it's easily the most expensive and time-consuming approach. For example, Forrester Research estimates that most native apps require at least 6 months of full-time work, and cost between $20,000 and $150,000, depending on complexity.

It's important to note that those estimates apply to single-platform native app

development. The cost rises exponentially when developing cross-platform native applications, as every platform requires a separate application built with a different programming language.

4. Development time

As mentioned above, Forrester Research estimates that a single native app requires 6 months of development time. If building native apps for more than one platform, the time requirements raise depending on the number of developers needed and application complexity. For example, using just one developer for cross-platform smartphone app development brings the development time up to 2 years (4 apps x 6 months each). However, development time estimates become increasingly complex when using multiple developers. For instance, if a business uses four different developers for cross-platform smartphone app development, they will receive four different app designs. As any project manager knows, ensuring that multiple apps created by multiple developers look and function identically is a very time- consuming task.

5. Maintenance cost

While all apps require regular updates and maintenance, native apps require the most future maintenance when compared with the other two mobile app options. Beyond regular app maintenance, native apps must also be updated with every new platform release. Additionally, businesses that build native apps for multiple platforms must maintain multiple applications, duplicating every change or update across all applications. For instance, a minor change to a cross platform smartphone and tablet application requires changes to eight

separate applications.

6. Limited control

When placed in an app store, a native application is completely controlled by the app store's owner (like Apple or Google). For instance, if Apple rejects or bans a company's app from their app store, the company has no recourse. If Apple decides an app doesn't meet their terms of service, the app is removed. If another company claims copyright over an element in the app, the app is removed. Or, if Apple decides the app isn't right for their store, the app is removed. The app store model puts companies at the mercy of a third party. All of the resources put into their application are wasted if that app store's owner decides the app isn't right for their store.

7. Learning Curve

For developing multi-platform application, once should go through the very different programming language specific to that platform. Apple has Objective C (and Swift) to develop iOS application. Google uses Java to develop android application and Microsoft deploys C# or Visual Basic for their Windows Phones.

3 What react native has to offer

1. Truly Native

Most mobile apps built with JavaScript use Cordova (https://cordova.apache.org/), or a framework built on top of it, such as the popular Ionic (http://ionicframework.com/) or Sencha Touch (https://www.sencha.com/products/touch/). With Cordova, you have the ability to make calls to native APIs, but the bulk of your app will be HTML and JavaScript inside a Web View. While you can approximate native components – and it is certainly possible to build a great UI with HTML and JS , but none of such app will match the look and feel of a real native app. The little things – like scrolling acceleration, keyboard behavior, and navigation – all add up and can create frustrating experiences for your customers when they don't behave as expected.

Although you still write JavaScript with React Native, the components you define will end up rendering as native platform widgets. React Native component definitions look and behave pretty much just like react for web components, but target native UI widgets instead of HTML (which on iOS gets rendered to a native UIView, and on android, android. View). When your components' data changes, React Native will calculate what in your view needs to be altered, and make the needed calls to whatever native UI widgets are displayed.

So react native application have "NO HTML, NO Browser and No Web view" that makes it more responsive giving native UI and user experience.

2. Responsive

Is React Native really fast ? JavaScript isn't as fast as native code can be, but for most tasks, JavaScript and React Native are more than capable of keeping your app running at 60 frames per second. Under the hood, your JavaScript code is run on its own thread, separate from the main UI thread. So even when your app is running complex logic, your UI can still be smoothly animating or scrolling at 60fps, so long as the UI isn't blocked by the JS thread.

3. Ease of Learning

One of react's greatest strengths is how readable it is, even to those unfamiliar with it. Many frameworks require that you learn a long list of concepts that are only useful within that framework, while ignoring language fundamentals. React does its absolute best to do the opposite.

5. Vibrant Ecosystem

Since the majority of your React Native code is just plain JavaScript, it reaps the benefits of all the advancements in the language and its ecosystem. For e.g. React Native can easily integrate with other power JavaScript library available out there.

6. One Design Tool

React Native's support for flexbox means you can use the exact same layout code for Android, iOS and web, instead of learning three different engines.

7. Developer Experience

Happy developers are productive developers, and React Native has a great development environment. Instead of repeatedly waiting for your code to compile and your app to restart while making tiny edits, changes to a React Native codebase are made to your running app without the need to restart.

And if you've written any JavaScript before, you're probably already familiar with the Chrome developer tools. When running React Native in development mode, you can attach to your desktop Chrome browser, so you'll be right at home with its debugger and profiling tools. Attaching to Chrome works either in the simulator or connected to a physical device.

Code sharing React Native can share code between Android and iOS. 9. Live Update

Anyone who has shipped an iOS app has experienced the frustration of waiting for App Store approval. With React Native, it is possible to do live updates to your app without going through the App Store – much like for a web app. Since the bulk of your app will be JavaScript, you can fetch updates on the fly over the network. There are already services to help with this

like AppHub (https://apphub.io/) .

10. A robust set of native components

Xcode provides Swift and Objective C developers with a plethora of components to build native apps. *Many* of these components have been built into React Native, and are immediately at your disposal when getting started.

Basic components like Images, ListViews, ScrollViews, Sliders, TabBars, etc. are able to be added into your app, using an easy to understand component-driven implementation. The full list (https://facebook.github.io/react-native/docs/getting-started.html#content) is far too expansive to list here, but take a look for yourself.

Beyond basic components, various iOS APIs have been incorporated as well. AsyncStorage, Camera Roll, PushNotifications, Vibration, etc. All APIs the developer has access to when building their native apps.

11. A robust set of third-party components

In addition to the large amount of native components the RN team provides, the RN developer community is creating additional components available to developers.

Sites like React Components (http://react-components.com/) and React Parts (https://react.parts/native) contain dozens of additional components for React, implementing other iOS specific design patterns, or in some cases additional functionality. In our most recent React Native project, we used Mapbox's React Native Component (https://github.com/mapbox/react-native-mapbox-gl) to create a custom stylized map in our app.

12. CSS-like Stylesheets

To provide aesthetics for the app, React Native utilizes a simplified version of CSS, to assist with styling your components. Most standard color, type, and layout style properties can be used in React Native, and the process is very familiar to a HTML/CSS workflow; albeit all of your styles need to be inlined within a components javascript file.

To provide positioning of components within the screen, React Native uses a slightly modified version of Flexbox, allowing the user to create a layout scaling between orientations and between device sizes. Yet another example of how smooth RN makes the transition from web to native.

13. Native Modules

Sometimes an app needs access to platform API, and React Native doesn't have a corresponding module yet. Maybe you want to reuse some existing Objective-C, Swift or Java code without having to implement it in JavaScript . In React Native it is possible for you to write real native code and have access to the full power of the platform.

14. Native UI Components

There are tons of native UI widgets out there ready to be used in the latest apps – some of them are part of the platform, others are available as third-party libraries, and still more might be in use in your very own portfolio. React Native has several of the most critical platform components already wrapped,

like ScrollView and TextInput, but not all of them, and certainly not ones you might have written yourself for a previous app. Fortunately, it's quite easy to wrap up these existing components for seamless integration with your React Native application.

15. Integrate with existing Application

Since React makes no assumptions about the rest of your technology stack – it's commonly noted as simply the V in MVC – it's easily embeddable within an existing non-React Native app.

16. Communication between React Native and Native

In Integrating with Existing Apps guide and Native UI Components guide we learn how to embed React Native in a native component and vice versa. When we mix native and React Native components, we'll eventually find a need to communicate between these two worlds. Some ways to achieve that have been already mentioned in other guides. This article summarizes available techniques.

4 Application build with React Native

These are some of the most well crafted React Native apps that we have come across.

Facebook Group Sound Cloud Discovery VR

More info can be found at showcase (https://facebook.github.io/react-native/showcase.html) or http://www.reactnative.com

5 Negative Side of React Native.

One of the major negative points about the react native is the availability proper of documentation. Facebook have recently open source this framework. Since then lot's and lot's of contributor are putting there effort in react native which is actually very good for this community. But with the increase of contributor the available document are getting scattered across the git hub and it's really tough task to find those api from one place.

Similarly react use a coding pattern called JSX (JavaScript XML) where both JavaScript and XML code are placed in single place, which was consider very bad programming pattern. This will make lot's of developer especially web developer uncomfortable in witting code in react.

6 Conclusion

Coming to final thought , I have never been so excited in developing mobile application using react native because of how ease it is to develop multi platform mobile application with one single tool. Till now react native support for iOS and Android mobile application development. But Facebook is putting lot of effort to make it available for other platform as well. Hopefully next year they will be able to achieve windows support as well.