**Compare REACT Native versus iOS and Android application environment**

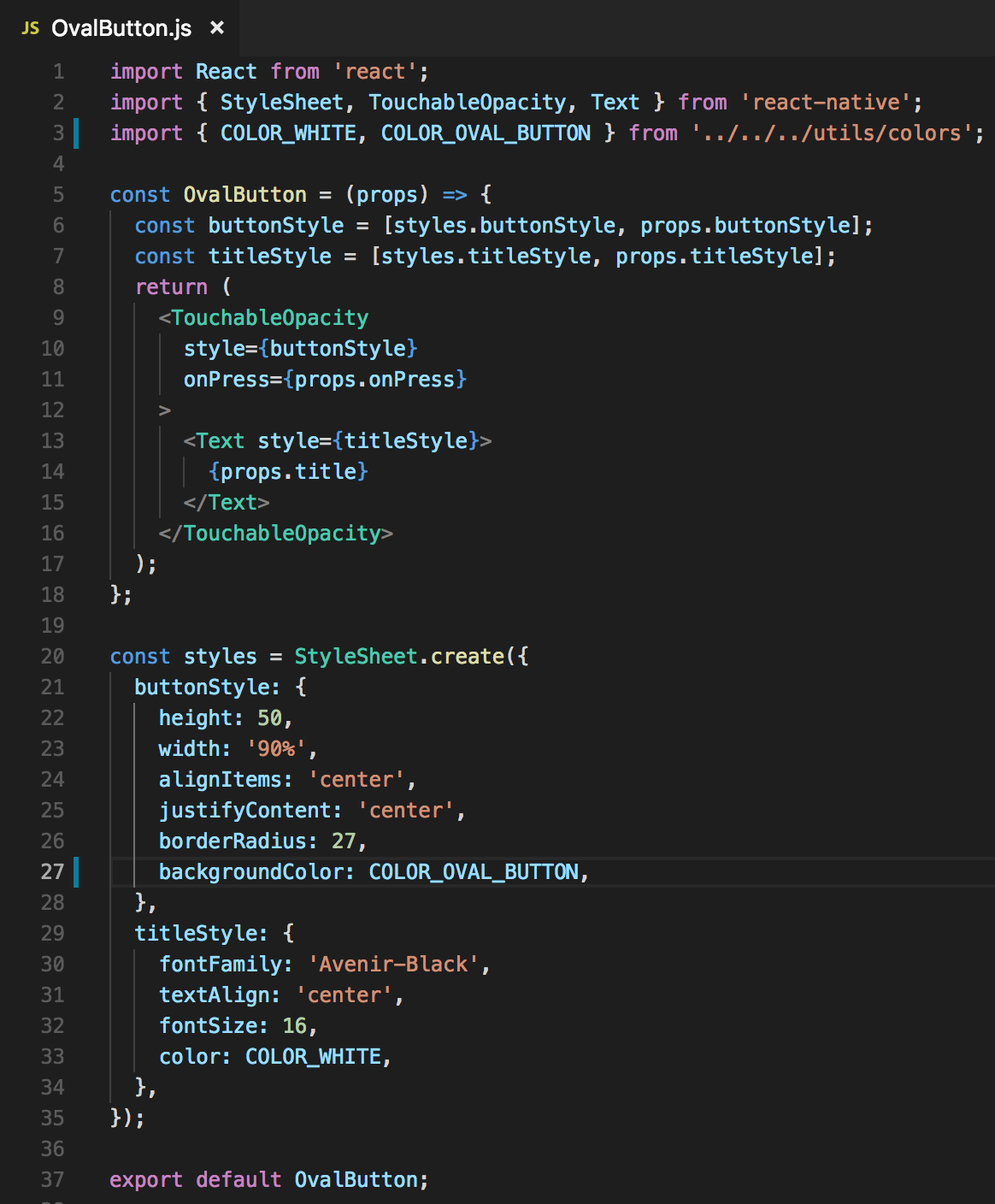
**Dingjun Bian**

**1, introduction:**

React Native is a framework created by Facebook that allows you to develop native mobile apps for iOS and Android with a single JavaScript codebase.

Firstly , let‘s first see how React Native builds a mobile app. React Native framework uses React.js library in order to create a true native mobile app. The important difference between React and React Native is that the latter uses native components instead of web components to create the user interface, along with JSX — a syntax that is used to embed XML with JavaScript.

This basically means that you can write mobile apps similar to how you would write web apps. The picture below is a simple example.



It is pretty clear that there is has a lot similarities to web development coding style. React Native components such as <TouchableOpacity> and <Text> are defined and organised exactly like HTML components such as<button> or <div>. In the meantime , react Native uses a set of components such as <Image>, <View>, <ScrollView> ,  <TextInput> ,  <FlatList> which map to their native iOS and Android counterparts, as well as some platform-specific components such as <DatePickerIOS> and <ProgressViewIOS> for iOS or <ViewPagerAndroid> and <ToastAndroid> for Android.

2. Compare

1)codebase

React Native enables a single JavaScript codebase for 2 different platforms. This means that it is not just easier to maintain the app by having the same development process for both platforms and reusing the same code, but it also requires less resources, because there is no need for separate iOS and Android teams. However, the problem is human Interface Guideline for iOS and Material Design for Android have a big share of differences, so if the project requirements dictate that these specific OS requirements should be followed for each native platform, React Native developer will need to write platform-specific code, which defeats the purpose of the single codebase. This might be a bigger issue when it comes to iOS, because Apple often updates and deprecates their technologies which can be hard to follow, while Android apps generally have more control of the system and are allowed more freedom. However, if it is OK for the app to look the same on both iOS and Android, then using React Native will greatly speed up the development and maintenance processes.

2) The programming language

React Native is based on React.js, which is written in JavaScript. This is a major advantage when it comes to the fact that JavaScript is such an extensive and popular language that it’s very hard to find a programmer who hasn’t used it at some point in their career.

While Java and Swift/Objective-C are strongly-typed, compiled languages, JavaScript is interpreted and often called an untyped language. This means that your variables can be anything at any time and a compiler is not going to help you, so if you’re not extra careful while writing your apps, the floor is paved for JavaScript horrors to ensue. When true native apps have more control of the variables and the logic of the app is more predictable, JavaScript relies on the programmer’s experience, Lint tools and automatic tests. Does this mean that JavaScript programs have more errors by definition? No. But it does mean that you might spend a lot more time searching for that app-breaking error which your compiler would have gotten for you in Java or Swift.

3)Hot reloading and live reloading

Anyone who has ever worked in XCode or Eclipse / Android Studio knows how long and tedious build time can be. This can especially be frustrating if you are working on a feature which is multiple screens away from the launch screen. This makes even trivial tasks such as changing a view color or changing a label text very time consuming.React Native’s solution for this problem is called hot reloading, which keeps the app running and saves the state (the data and the screen you’re on), only injecting the changes that are being made in the code. While being far from perfect, this feature works well for most of the usual use cases, saving a lot of precious development time.Even without using hot reloading, app build time is significantly faster using React Native than on native iOS and Android, and live reloading mechanism can be used to automatically reload the app every time the code is changed, without having to build it manually every single time.

4)User Interface

React Native’s approach to structuring UI is *Flexbox*, which is already very popular in web development and enables the developer to create a responsive web or mobile UI very easily. As such, it is a technology which rivals Android’s *XML / Constraint layout*approach, or iOS’s *Storyboard / XIB / Coding with UI libraries such as Neon* approach. One considerable advantage that native mobile has though is that you have access to all the native APIs such as Camera, Touch ID and GPS, as well as tools for creating animations and complex user interface. There is no middle layer and you are free to take advantage of everything the mobile platform has to offer. React Native does not excel at creating complex UI and animations. It does have the *Animated API* which is a neat solution, but is still far behind the native capabilities.

React Native simply does not provide the amount of user interface power as native mobile does, so for apps that require a highly complex UI or sophisticated animations it might not be the best choice. But for simple-looking apps it is a better option because of the way Flexbox handles responsive UI layout, as well as the simple and intuitive structuring of the XML components.

5) Native modules

While React Native can handle a large amount of cross-platform use cases, it is impossible for it to cover all the native mobile ground. This means that there will always be a need for native modules. Native modules are basically code in native language which handles a specific native feature. Examples of native modules are modules for camera, native image picker, push notifications, deep linking, and third party services like Auth0 or Amplitude. So, while this means that basically any native code can be written in native language if React Native cannot handle it, it also means that React Native developer is forced to dig in native IDEs (XCode / Android Studio) in order to enable the modules. An experienced native mobile developer will usually quickly find a way to link the library, but someone who is purely a React Native developer and has never made a true native mobile app will hit many walls before finally connecting everything together.

In addition, it is worth noting that if the team uses a cloud service for Continuous Integration such as Azure, even more native linking problems might arise, because XCode and Android Studio on the cloud might have a different version and configuration than your local one, so even if everything works well on your machine, it might not build remotely and the developer will need to do more fine-tuning in order for the linking to work on both sides.The same goes for publishing the app. If the team doesn’t have developers that are experienced in native iOS and Android, they may struggle to release the app. This is more of an iOS problem, with Android release workflow being more straightforward. In order to release an iOS app, React Native developer will still need to use XCode in order to configure release-related data such as provisioning profiles and certificates, which can be overwhelming at first.All of this leads to a very important conclusion: **React Native is much more effective if used by developers who already have experience with native mobile development, because it doesn’t completely remove the need of going native.**