Introduction to React Native

What is React Native?

React Native is a library built with the same API as React. It uses the same declarative UI paradigm but renders directly to native components. You can write your app once in React and run it on multiple native platforms, like Android and iOS.
While “native” is part of the name, it’s not pure native app development: Expo and React Native still use JavaScript to run your app. Instead of rendering this JavaScript with a web engine, they use the actual native components from the platform.

Expo and React Native

Expo is a platform to make universal React apps that helps you develop, build, deploy, and quickly iterate on mobile apps. It provides a range of tooling to make development with React Native even easier. Although Expo isn’t required to create React Native apps, it helps developers by removing the need for basic native knowledge.

- *Expo Go* is an app you can download on your phone to “view” your app in development.
- *Expo CLI* is a tool to create, manage, and develop your apps.
- *Expo SDK* is a modular set of packages that provide access to native APIs, like Camera or Notifications.
- *Expo Snack* is a web-based playground where you can write React Native snippets and run them in the browser.

Native Rendering

There are a few components which ship with React Native; we call these core components. React Native knows how to render these on a specific platform because they are tied to a native component counterpart.

By creating components and rendering them, you tell React Native what to render. The JavaScript with the components is bundled in your app and executed in a separate thread from the native UI. This JS thread instructs React Native what it needs to render. Splitting this into a JS thread and a UI thread allows the platform to understand what needs to be rendered without blocking the actual interface components.

To visualize these two threads, consider a highway with only a single lane for traffic. With both slow and fast traffic combined on this lane, some traffic might slow down others. Adding another lane allows the faster traffic to run independently of the slower traffic. While JS isn’t necessarily slow traffic, it can still block the UI thread and cause stuttering in visible animations.

Besides the visible UI components, the native UI thread is also handling native API requests. Some functionality, like GPS location, needs to be requested from the native APIs. If your JS code uses this kind of functionality, it interacts with the native API using native code. The data from this native code is sent back to the JS code and handled in your app.

Cross-Platform Differences

A lot of Expo and React Native code can be reused across multiple native platforms, but there are some differences you should be aware of. Native components may look different because of the platform-provided design guidelines. On iOS, Apple implemented their Human Interface guidelines while Android provides their Material Design guidelines. Some of the core components might look different on other platforms because of this.

There are also differences in the functionality provided by each platform. Apple wants app developers to use their “Authenticate with Apple” feature, but this is not available on Android. Because of that, it isn’t always possible to provide a unified API for native functionality.

Comparing Cross Platform Development & Native Development

For most applications, cross platform tools like Expo & React-Native are a good choice.

- Apps built with Expo or React Native can run on multiple platforms. That means faster development and less code to maintain while sharing most of the code.
- It provides direct access to native functionality, allowing developers to make the app as performance as pure native apps.
- Getting started with Expo and React Native only requires basic web development and basic native platform understanding.

However, there are some areas where Cross Platform Tools like React Native don't perform quite well

- Pure native apps have a higher performance ceiling compared to Expo and React Native apps.
- Expo and React Native are abstractions on top of the native platform. They need to follow the latest changes and functionality from the native platforms.
- Complex apps often require you to optimize and customize native code— that requires a good understanding of every platform you need to support.

Setting Up React-Native & Expo locally

Text Editor

You can use any text editor of your choice, we recommend Visual Studio Code (https://code.visualstudio.com).

NodeJS

React Native requires that we have NodeJS set up. To download and set up the latest version of NodeJS, check out this link (https://nodejs.org/en/).

To check the version of NodeJS, run the following command

```
node -v
```

To check the version of npm, run the following command

```
npm -v
```

Expo CLI

We will be making use of the *expo-cli* to manage our React Native Application.

To install the expo cli, run the following command

```
npm install -g expo-cli
```

Expo Mobile App

While it is possible for us to test expo applications on android or iOS emulators on the device, we may alternatively use the expo app to test our applications on a real device, by simply installing the expo app on our device of choice. To do this, head over to the AppStore or Google Play Store and download the expo app. It's a free application.

Documentaion and Pre-requistes

- React Native Documentation (https://youtu.be/W6NZfCO5SIk)
- JavaScript Course (https://reactnative.dev/docs/getting-started)