Pavneet Singh

Getting Started with React Native on Android

Pavneet Singh

Aug 7, 2020 • 13 Min read • 4,930 Views

Aug 7, 2020 • 13 Min read • 4,930 Views

Web Development Front End Web Dev... Client-side Framew... React

Introduction

28

Introduction

React Native is one of the most famous open-source hybrid frameworks to build apps for Android, iOS, and web platforms using JavaScript with a single codebase.

The name “React Native” is composed of two words: React and Native. “React” states the use of the React app development environment, and “Native” represents the use of native

Android/iOS/Web UI widgets to develop apps for better performance. React Native interacts with the Native platform at runtime via JavaScript to construct the native view for React Native UI components. React Native also offers ready-to-use inbuilt APIs for UI development such as `<ScrollView>`, `<FlatList>`, `<ImageView>`, etc. to develop apps.

This guide covers the basics of React Native development on Android using the standard `npx react-native` CLI.

React Native Versus React

React Native is built upon the React framework, but it uses different APIs and technologies to make React apps compatible with the Native mobile platform:

- **Development Tools**: React Native needs an Android studio setup for Android and XCode for iOS development. For React Native project development, an IDE like Atom, Notepad++, Sublime, or Visual Studio Code can be used.
- **UI Widget**: React Native provides a comprehensive set of APIs (UI components) to build UI for a React Native app, so there is no HTML, only predefined core components APIs to build UI interface.

- **Style**: React Native doesn't use CSS or HTML. Instead, it uses StyleSheet API object (like JSON) to define style properties for widgets (views or UI component).
- **UI Structure**: To mimic the Native UI, React Native uses the core UI APIs to build the interface so there is no `DOM` or `window` object.
- **Platform-specific UI**: There are still some APIs that are not supported by React Native core UI components like Map, so React Native offers the support to create a `native module` to create a wrapper to use platform-specific APIs or third party modules.

Expo CLI

As mentioned earlier, the standard React Native `create-react-native-app` CLI requires the Native development tools (Android Studio or XCode) and other npm packages, such as Camera and Map, to build specific features. Expo CLI overcomes the limitations of standard React Native CLI with the following features:

- An expo project can be run directly into the browser without any external dependency (XCode or Android studio).
- Expo offers easy app sharing via link or QR-code, which is quite useful to release and test updates.
- Expo offers inbuilt APIs like Camera, SMS, etc.

Expo is an enhanced version of the standard React Native CLI for quick design, development, and publishing, but despite many pros Expo has some limitations:

- No support for Native modules or external libraries that requires Native code.
- Expo covers a wide variety of APIs but it still does not support all Native APIs.

Expo is an easy way to get started quickly with React Native development in browsers. Expo also supports unimodules to use Expo SDK in React Native apps.

Prerequisite

React Native follows the similar development structure and tools of React, so this guide assumes that you have the basics knowledge of following technologies and tools.

Technologies

- Basics of JavaScript
- HTML, Objects, and EcmaScript 6 (ES6 classes and arrow functions, etc.)

Tools

- Node.js
- Command line interface (CLI)

Development Environment Setup

The `npx react-native init` requires external tools, so follow the below steps to download and install the required tools, as per operating system:

MacOS or Linux

1. Verify Homebrew installation using `brew -v`, and if Homebrew is not installed then install Homebrew using

   sh
   ```sh
   /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/i
   ```

   Note: Homebrew installation requires the `xcode-select` command-line tool to work so it's recommended to install Xcode as well or it will ask to confirm the installation of `xcode-select` tool.
2. Install `node` and `watchman`

   sh
   ```sh
   brew install node watchman
   ```

3. Install JDK using `break cask` to install GUI software installation setups

```sh
brew cask install adoptopenjdk/openjdk/adoptopenjdk8
```

4. Download Android Studio and follow the official installation steps (from video), as per operating system.
5. Add the required android SDK dependencies' paths in environment variables for React Native:

```sh
# can use any editor instead of open
open ~/.bash_profile
```

6. Paste the following text at the end of the file

```sh
# after other pre-configured environment variables
export ANDROID_HOME=$HOME/Library/Android/sdk
export PATH=$PATH:$ANDROID_HOME/emulator
export PATH=$PATH:$ANDROID_HOME/tools
export PATH=$PATH:$ANDROID_HOME/tools/bin
export PATH=$PATH:$ANDROID_HOME/platform-tools
```

MacOS Cataline has a default shell as `zsh` so to fix the warning below, edit/create `~/.zprofile` (under `/users/username/`) file for Cataline.

```sh
chsh -s /bin/zsh
# create the .zprofile under users/your_user_name and copy content from bash_profile
source ~/.zprofile
```

Helpful Commands

- `printenv` prints all environment variables.
- `xcode-select --version` allows you to view the version of xcode-select CLI.

Windows

1. Download Chocolatey package manager and run the below command in command-prompt (run as Administrator):

```sh
choco install -y nodejs.install python2 openjdk8
```

2. Download Android Studio and follow the official installation steps (with video), as per operating system.
3. Add the default android SDK path in environment variables, and make sure to replace `UserName` with your username:

```cmd
setx ANDROID_HOME "C:\Users\UserName\AppData\Local\Android\SDK"
```

and add `platform-tools` to `PATH` variable. Make sure to replace `UserName` with your user name:

```cmd
setx /M PATH "%PATH%;C:\Users\UserName\AppData\Local\Android\SDK\platform-
```

Create and Run a React Native Project

The steps to run the project is the same on all operating systems:

1. Create a React Native project:

```sh
npx react-native init RNClickCounter
```

2. Follow the steps to create an android virtual device (AVD)
3. Run the projects:

```sh
cd RNClickCounter
npx react-native run-android --verbose
```

The above command process may ask to install `CocoaPods`, which is a dependency manager for iOS projects and required to run iOS apps.

In the above command, `--verbose` is optional but useful to view any potential issues, like below.

Known Gradle Issues

- `InvokerHelper Error`: Gradle version `6.1.1` can cause this issue, so make sure to update `distributionUrl` attribute in `RNClickCounter\android\gradle\wrapper\gradle-wrapper.properties` file:

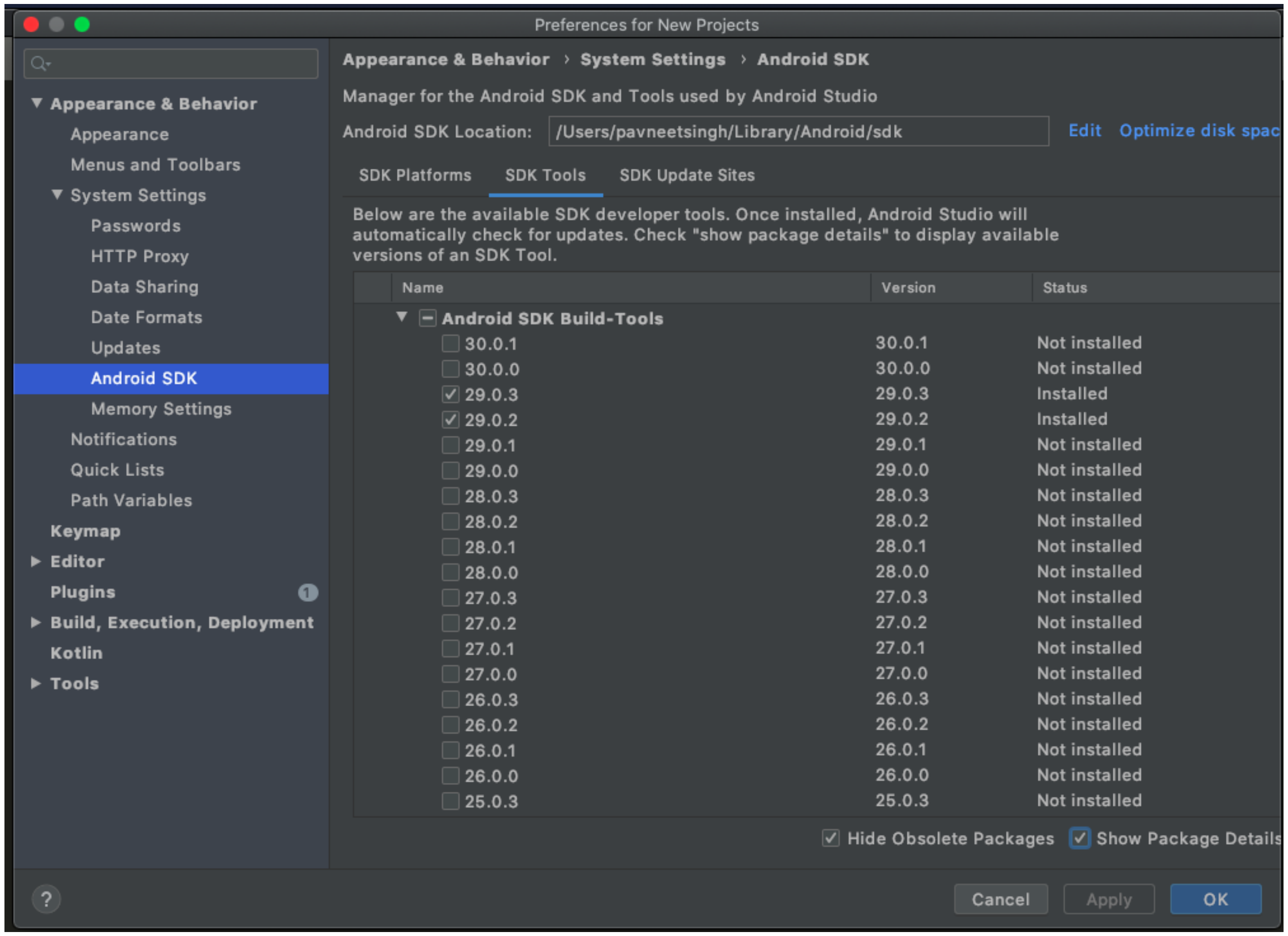
sh

```sh
# To fix the "Could not initialize class org.codehaus.groovy.runtime.Invok
distributionUrl=https\://services.gradle.org/distributions/gradle-6.5.1-al
```

and update classpath in `RNClickCounter\android\build.gradle`:

```
classpath("com.android.tools.build:gradle:4.0.0")
```

Note: React Native may not use the exact `buildToolsVersion` version declared in the `RNClickCounter/android/build.gradle` file, so in case of error the specific version needs to be installed from the Android SDK.


Preferences for New Projects

- Appearance & Behavior
  - Appearance
  - Menus and Toolbars
  - System Settings
    - Passwords
    - HTTP Proxy
    - Data Sharing
    - Date Formats
    - Updates
    - Android SDK
    - Memory Settings
  - Notifications
  - Quick Lists
  - Path Variables
- Keymap
- Editor
- Plugins
- Build, Execution, Deployment
- Kotlin
- Tools

Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /Users/pavneetsingh/Library/Android/sdk Edit Optimize disk spac

SDK Platforms | SDK Tools | SDK Update Sites

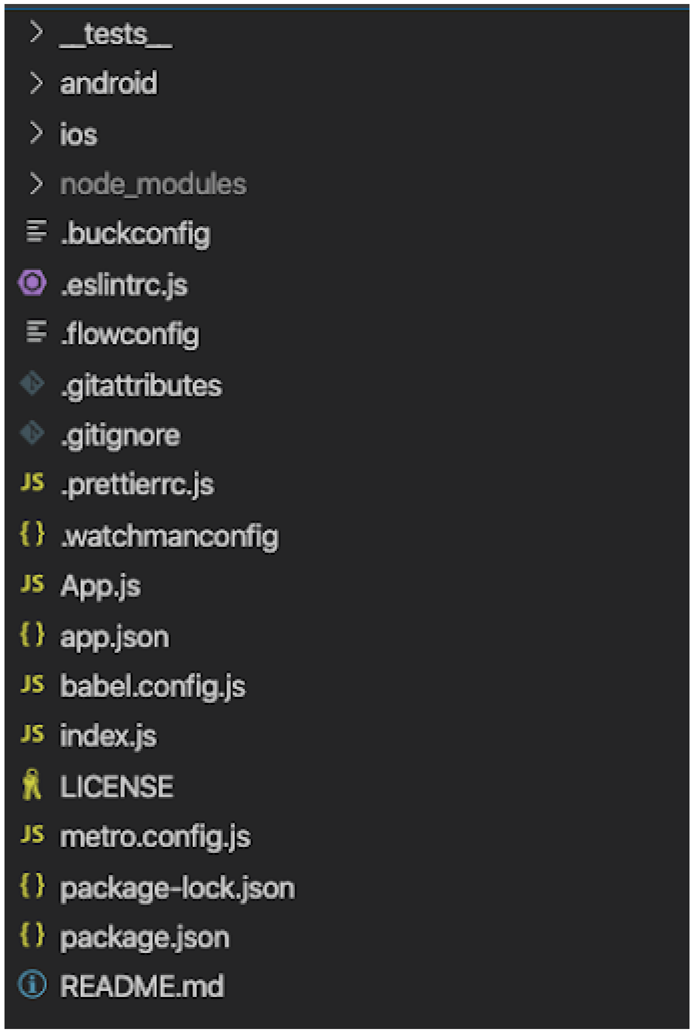
Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
Android SDK Build-Tools		
☐ 30.0.1	30.0.1	Not installed
☐ 30.0.0	30.0.0	Not installed
✓ 29.0.3	29.0.3	Installed
✓ 29.0.2	29.0.2	Installed
☐ 29.0.1	29.0.1	Not installed
☐ 29.0.0	29.0.0	Not installed
☐ 28.0.3	28.0.3	Not installed
☐ 28.0.2	28.0.2	Not installed
☐ 28.0.1	28.0.1	Not installed
☐ 28.0.0	28.0.0	Not installed
☐ 27.0.3	27.0.3	Not installed
☐ 27.0.2	27.0.2	Not installed
☐ 27.0.1	27.0.1	Not installed
☐ 27.0.0	27.0.0	Not installed
☐ 26.0.3	26.0.3	Not installed
☐ 26.0.2	26.0.2	Not installed
☐ 26.0.1	26.0.1	Not installed
☐ 26.0.0	26.0.0	Not installed
☐ 25.0.3	25.0.3	Not installed

✓ Hide Obsolete Packages ✓ Show Package Details

Project Structure

The project structure of a React Native project is similar to React. The underlying Android and iOS project is pre-configured to use React components to build platform-specific apps.



```
> __tests__
> android
> ios
> node_modules
.buckconfig
.eslintrc.js
.flowconfig
.gitattributes
.gitignore
.prettierrc.js
.watchmanconfig
App.js
app.json
babel.config.js
index.js
LICENSE
metro.config.js
package-lock.json
package.json
README.md
```

Let's go through the key elements of a React Native project:

- `Index.js` is an entry point for a React app to register the app and to load required modules for the JS executions environment.
- `App.js` is the first/main screen of the app.
- `android` folder contains all the Native Android development, and build files, which is almost similar to the Native Android project structure.

- `ios` folder contains all the Native iOS development and build files, which is almost similar to the Native iOS project structure.
- `package.json` contains the details about the app (name, version), dependencies, and executable script details.
- `app.json` contains the app name details.

Implement Click Counter

React Native uses some common mobile components like `Button`, `View`, `Text`, etc., along with React Native-specific components like `SafeAreaView` and `StyleSheet`. Let's go though some basic components to build the click counter app:

- `SafeAreaView` adds the required padding for camera-notches/sensor-housing and reflects the area that is not covered by any of the top views like toolbar, navigation, etc.
- `Text` displays text on the screen. It is similar to `UILabel`, `TextView`, or `<p>` tag.
- `View` is a basic UI container element with flexbox layout support. The Native equivalents of view are `UIView`, `View`, or `div` tag.
- `Button` represents the Native platform-specific button with platform-specific style.
- `StyleSheet` is used to define the style attributes for elements that will be mapped to Native-style values.
- `useState` is a React hook that is used to maintain a state (stored values) in a functional component. This is used in the `App.js`

functional component to keep the track of the counter variable's state. The counter variable should be modified by the callback method `setCount`, and returned by `useState`.
- `export default App` is used to allow other components to import the `App` component. There can be only one default export in a file.
- `flex: 1` is used to define the CSS3 flexbox style responsive layout vertically.
- `React$Node` represents a type of React node (from flow type check) whose value can be a ReactChild, ReactFragment, ReactPortal, boolean, null, number, or string.

Steps to Implement Click Counter

Follow the below steps to implement click counter in the `App.js` component:

1. Implement `react-hook` to store the updated value of `count`. The `setCount` method will be used to update the value of `count`:

JSX

```jsx
        const [count, setCount] = useState(0);
```

2. Implement callback functions to increment/decrement the value of `count`:

JSX

```jsx
const counterPlus = () => {
 setCount(count + 1 <= Number.MAX_SAFE_INTEGER ? count + 1 : count)
}

const counterMinus = () => {
  setCount(count - 1 >= Number.MIN_SAFE_INTEGER ? count - 1 : count)
}
```

3. Create the style object to center the views inside container elements, and design details for other views such as `Button` and `Text`:

JSX

```jsx
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#e6e6fa',
  },
  textConter: {
    fontSize: 28,
    color: '#000',
  },
  buttonStyle: {
    width: "80%",
    margin: 10,
  }
});
```

4. Now apply the style to the elements and add the listener on the button to complete the implementation:

JSX

```jsx
import React, { useState } from "react";
import { SafeAreaView, StyleSheet, Text, StatusBar, Button, View} from 'r

const App: () => React$Node = () => {

  const [count, setCount] = useState(0);

  const counterPlus = () => {
    setCount(count + 1 <= Number.MAX_SAFE_INTEGER ? count + 1 : count)
  }

  const counterMinus = () => {
    setCount(count - 1 >= Number.MIN_SAFE_INTEGER ? count - 1 : count)
  }

  return (
    <>
      <StatusBar barStyle="dark-content" />
      <SafeAreaView style={styles.container}>
        <Text style={styles.textConter} >{count}</Text>
        <View style={styles.buttonStyle}>
          <Button
            onPress={counterPlus}
            title='+' />
        </View>
        <View style={styles.buttonStyle}>
          <Button
            onPress={counterMinus}
            title='-' />
        </View>
      </SafeAreaView>
    </>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#e6e6fa',
  },
  textConter: {
    fontSize: 28,
    color: '#000',
  },
  buttonStyle: {
    width: "80%",
    margin: 10,
  }
});

export default App;
```

Conclusion

React Native is a great way to build hybrid apps. You can either use `Expo` or `npx react-native` CLI to get started with React Native development. The optimized codebase is available at RnClickCounter repository. Hopefully, this guide explained the necessary details to get started with React Native on Android. Happy Coding!

28