React Native lets you build mobile apps using only JavaScript. It uses the same design as React, letting you compose a rich mobile UI from declarative components. With React Native, you don't build a mobile web app, an HTML5 app, or a hybrid app; you build a real mobile app that's indistinguishable from an app built using Objective-C or Java. React Native uses the same fundamental UI building blocks as regular iOS and Android apps. You just put those building blocks together using JavaScript and React.

features of React Native −

* **React** − This is a Framework for building web and mobile apps using JavaScript.
* **Native** − You can use native components controlled by JavaScript.
* **Platforms** − React Native supports IOS and Android platform.

React Native Advantages

Follow are the advantages of React Native −

* **JavaScript** − You can use the existing JavaScript knowledge to build native mobile apps.
* **Code sharing** − You can share most of your code on different platforms.
* **Community** − The community around React and React Native is large, and you will be able to find any answer you need.

limitations of React Native −

* **Native Components** − If you want to create native functionality which is not created yet, you will need to write some platform specific code.

Setup:

Install Node and NPM

Step 1: Install create-react-native-app

After installing NodeJS and NPM successfully in your system you can proceed with installation of create-react-native-app (globally as shown below).

C:\Users> npm install -g create-react-native-app

## Step 2: Create project

Browse through required folder and create a new react native project as shown below.

C:\Users >cd Desktop

C:\Users\Desktop>create-react-native-app MyReactNative

## Step 3: NodeJS Python Jdk8

Make sure you have Python NodeJS and jdk8 installed in your system if not, install them. In addition to these it is recommended to install latest version of yarn to avoid certain issues.

## Step 4: Install React Native CLI

You can install react native command line interface on npm, using the install -g react-native-cli command as shown below.

npm install -g react-native-cli

Start react native

To verify the installation browse through the project folder and try starting the project using the start command.

C:\Users\Tutorialspoint\Desktop>cd MyReactNative

C:\Users\Tutorialspoint\Desktop\MyReactNative>npm start

If everything went well you will get a QR code as shown below.



As instructed, one way to run react native apps on your android devise is to using expo. Install expo client in your android devise and scan the above obtained QR code.

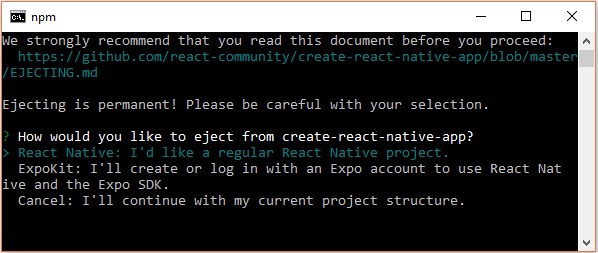
Step 6: Eject the project

If you want to run android emulator using android studio, come out of the current command line by pressing **ctrl+c**.

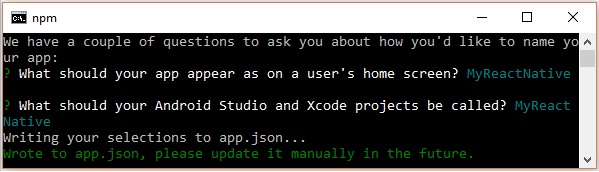
Then, execute run **eject command** as

npm run eject

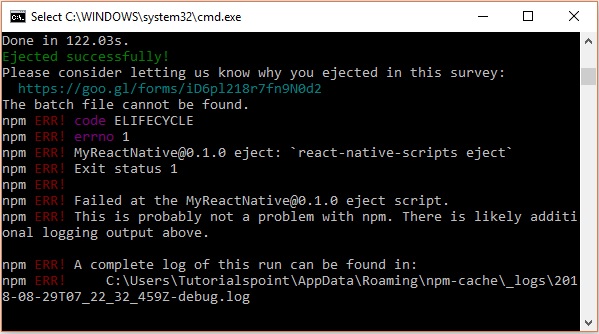
This prompts you options to eject, select the first one using arrows and press enter.



Then, you should suggest the name of the app on home screen and project name of the Android studio and Xcode projects.



Though your project ejected successfully, you may get an error as −



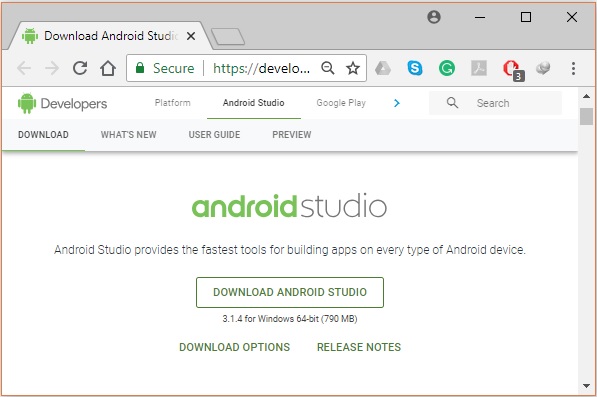
Ignore this error and run react native for android using the following command −

react-native run-android

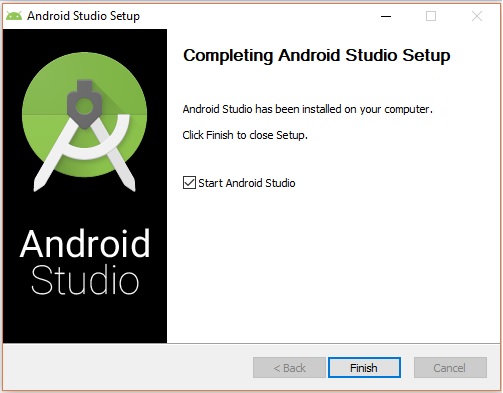
But, before that you need to install android studio.

Step 7: Installing Android Studio

Visit the web page <https://developer.android.com/studio/> and download android studio.

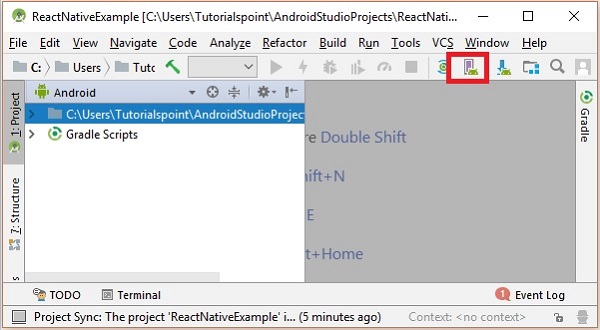


After downloading the installation file of it, double click on it and proceed with the installation.



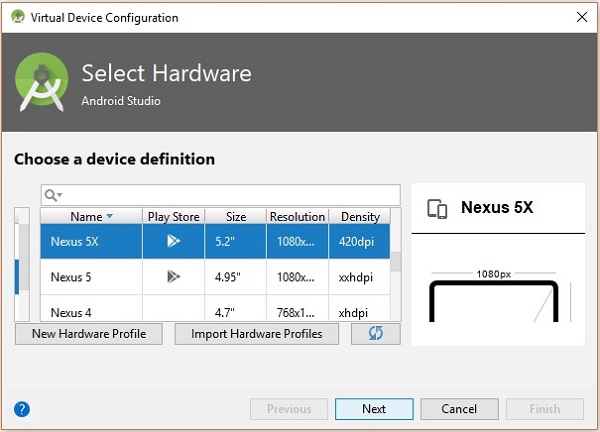
Step 8: Configuring AVD Manager

To configure the AVD Manager click on the respective icon in the menu bar.

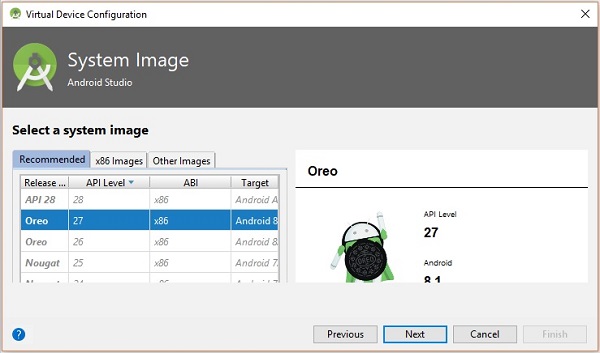


Step 9: Configuring AVD Manager

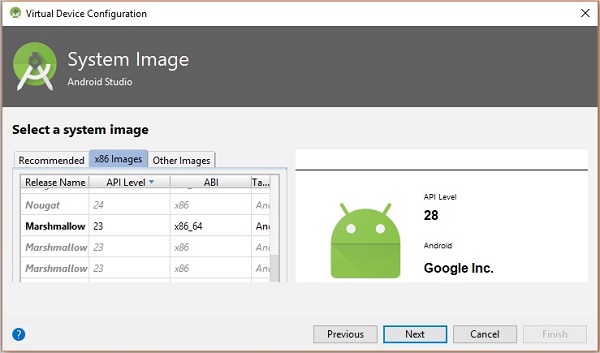
Choose a device definition, Nexus 5X is suggestable.



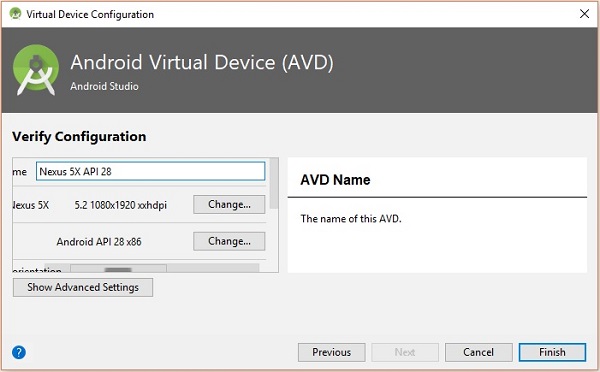
Click on the Next button you will see a System Image window. Select the **x86 Images** tab.



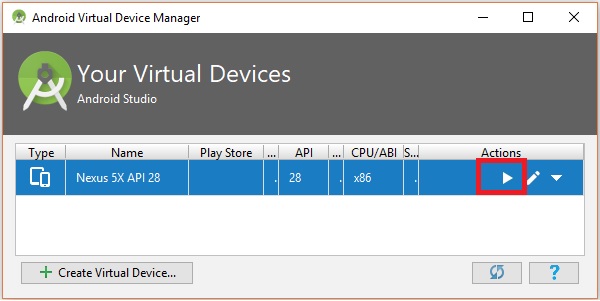
Then, select Marshmallow and click on next.



Finally, click on the Finish button to finish the AVD configuration.

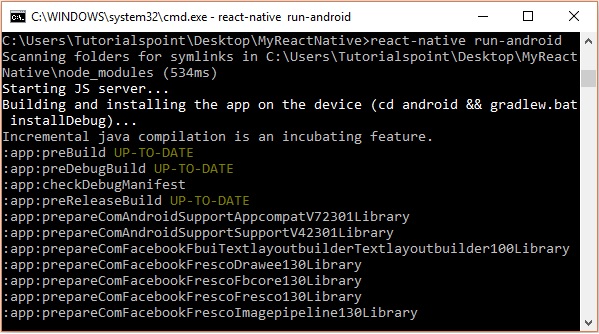


After configuring your virtual device click on the play button under the Actions column to start your android emulator.

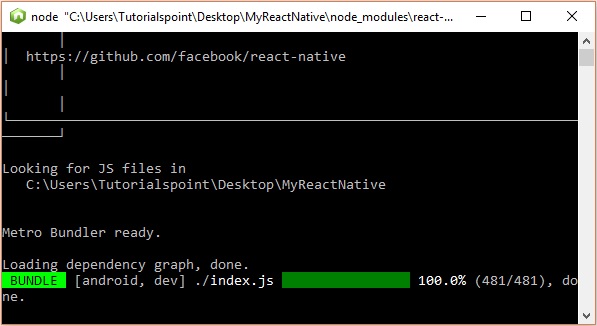


Step 10: Running android

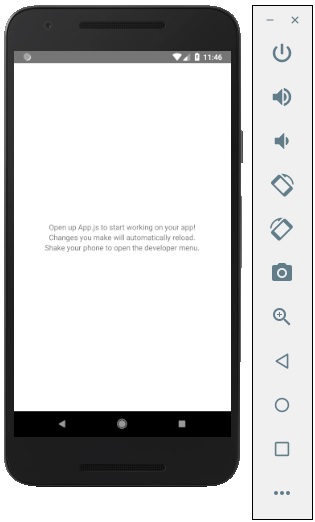
Open command prompt, browse through your project folder and, execute the **react-native run-android** command.



Then, your app execution begins in another prompt you can see its status.



In your android emulator you can see the execution of the default app as −



Step 11: local.properties

Open the **android** folder in your project folder ***SampleReactNative/android*** (in this case). Create a file with named **local.properties** and add the following path in it.

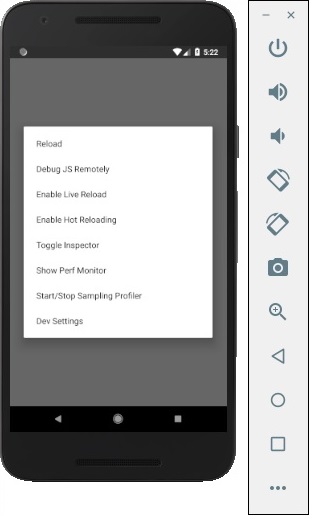
sdk.dir = /C:\\Users\\Tutorialspoint\\AppData\\Local\\Android\\Sdk

here, replace **Tutorialspoint** with your user name.

Step 12: Hot Reloading

And to build application modify the App.js and the changes will be automatically updated on the android emulator.

If not, click on the android emulator press **ctrl+m** then, select **Enable Hot Reloading** option.



## Difference between State and Props

The **state** is mutable while **props** are immutable. This means that **state** can be updated in the future while props cannot be updated

We will use the same code that we used in our **React Native - Styling** chapter. We will only change the **PresentationalComponent**.

Layout

To achieve the desired layout, flexbox offers three main properties − **flexDirection justifyContent** and **alignItems**.

The following table shows the possible options.

|  |  |  |
| --- | --- | --- |
| **Property** | **Values** | **Description** |
| flexDirection | 'column', 'row' | Used to specify if elements will be aligned vertically or horizontally. |
| justifyContent | 'center', 'flex-start', 'flex-end', 'space-around', 'space-between' | Used to determine how should elements be distributed inside the container. |
| alignItems | 'center', 'flex-start', 'flex-end', 'stretched' | Used to determine how should elements be distributed inside the container along the secondary axis (opposite of flexDirection) |

If you want to align the items vertically and centralize them, then you can use the following code.

**App.js**

import React, { Component } from 'react'

import { View, StyleSheet } from 'react-native'

const Home = (props) => {

return (

<View style = {styles.container}>

<View style = {styles.redbox} />

<View style = {styles.bluebox} />

<View style = {styles.blackbox} />

</View>

)

}

export default Home

const styles = StyleSheet.create ({

container: {

flexDirection: 'column',

justifyContent: 'center',

alignItems: 'center',

backgroundColor: 'grey',

height: 600

},

redbox: {

width: 100,

height: 100,

backgroundColor: 'red'

},

bluebox: {

width: 100,

height: 100,

backgroundColor: 'blue'

},

blackbox: {

width: 100,

height: 100,

backgroundColor: 'black'

},

})