**React Native**

**1 – introduction to the course**

What is React Native?

* Framework for develop android and ios apps.
* Written in JavaScript with React Js.
* It will convert code to native apps.
* Current stable version 0.72.7.
* Initial release date: 26th Mar 2015.

Wny to learn React-Native?

* Make Android, IOS, and many more platforms app.
* High demand.
* Easy to learn and develop apps.
* If you know React Js, you can easily learn RN.
* Maintened by Facebok so this is a long term player.

Main Topics of Course..

* Setup an dinstall React-Native
* React-Native basics
* Components
* Hooks and Lifecycle
* Navigation
* Handle List
* Form handling
* REST APIs
* Redux
* Project
* Interview Questions

**2 - Installation and setup**

We will follow official docs for React Native setup ( [Setting up the development environment · React Native](https://reactnative.dev/docs/environment-setup?guide=native&os=windows&platform=android))

* Install Choco, Node, and JDK (11)
* Install Android Studio.
* Download SDK an emulator.
* Add environment variables.
* Create first React Native app.
* Run App

To create a react native application ->

***npx react-native@latest init AwesomeProject***

**Note**: this command will create TypeScript based application. To get JavaScript based application go for the specific version like 0.70

with specific version ->

***npx react-native@X.XX.X init AwesomeProject --version X.XX.X***

**What is Metro?**  
Metro is a JavaScript bundler. It takes in an entry file and various options, and gives you back a single JavaScript file that includes all your code and its dependencies.

Metro has three separate stages in its bundling process:

1. Resolution
2. Transformation
3. Serialization

***npm start*** will do this Metro task.

Let Metro Bundler run in its own terminal. Open a new terminal inside your React Native project folder. Run the following:

***npm run android***

If everything is set up correctly, you should see your new app running in your Android emulator shortly.

* Open **App.tsx** in your text editor of choice and edit some lines.
* Press the **R** key twice or select Reload from the Dev Menu **(Ctrl + M)** to see your changes!

**3 - Making First Program**

* How to start App again.
* make the first change in app.
* Just keep simplest code only.
* Interview Questions

***What is the difference between React Native (Framework) and React Js (Library)?***

**4 - Big Change in React–Native 0.71**

A black background with white text

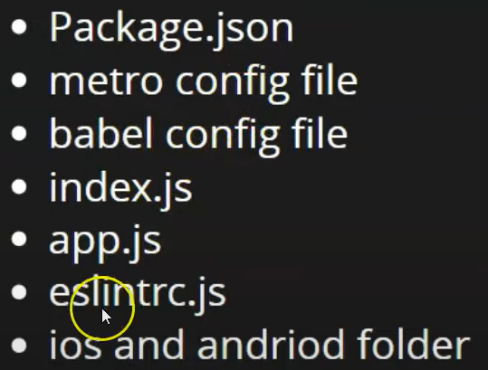
Description automatically generated

***App.tsx***

A close-up of a black box

Description automatically generated

**5 - React Native App File Structure**



* Package.json - Same as React JS.
* Metro.config.js – reloads the application fast.
* Babel.config.js – Sometimes browser or Application can’t understand the newly introduced concepts like ECMA script etc. so Babel helps us there.
* Index.js – Entry point of JavaScript, our components get registered here.
* App.js – 1st component of our file.
* Eslintrc.js – ES stands for ECMA Script, we can do ecma script related configurations here in this file.
* Ios an android folder : used to create build for respective platforms.
* Gemfile – is create for Ruby (In order to debug the application).

**6 – View, Text and Button Components**

View is just a wrapper to enclose multiple text and buttons.



**7 – JSX (Power House of React-Native)**

JavaScript Extension (added some other features in JavaScript).

It helps us to write HTML in JavaScript code.

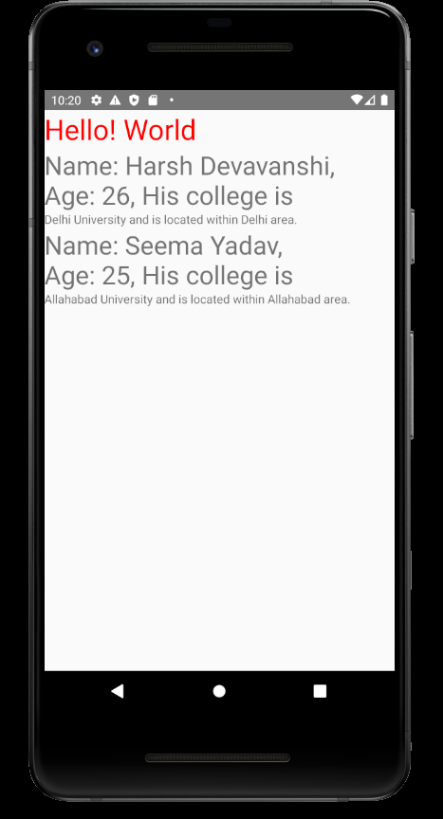
**8 – Components in React-Native**

**A black screen with white text

Description automatically generated**

A screenshot of a computer

Description automatically generated

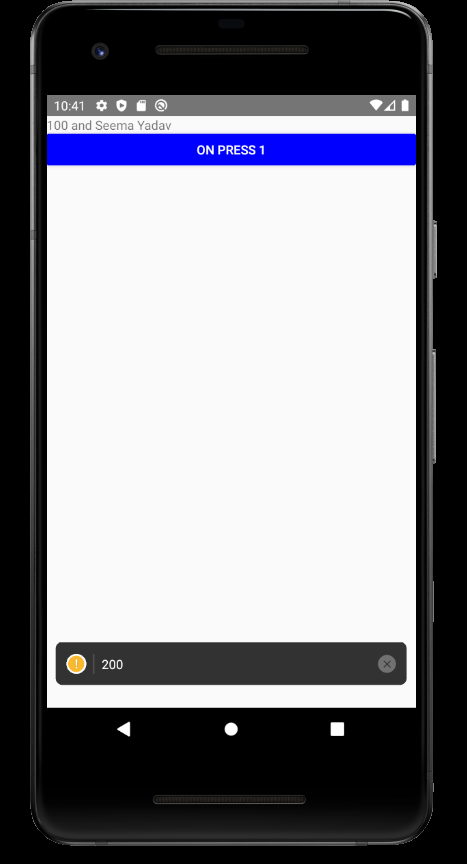
********

**Lesson 8 Buttons**

********

**Lesson – 9 States in React-Native**

****

****

**Lesson – 10 Props**

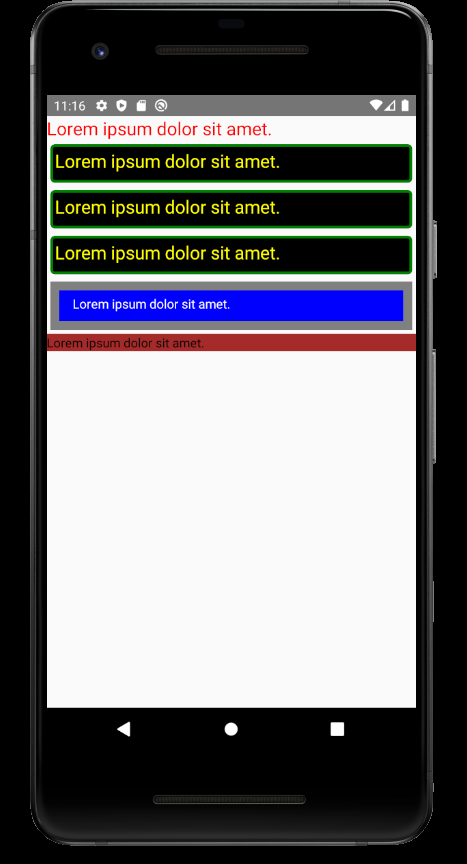
********

**Lesson – 11 styles**

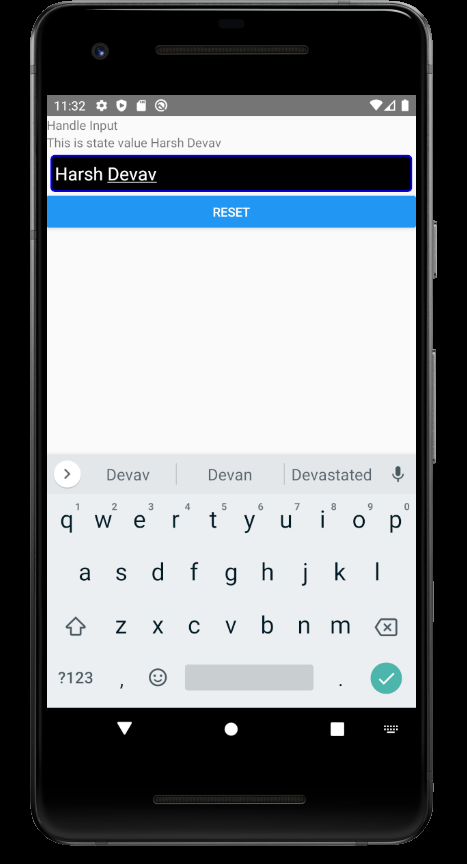
Three ways only

1. Inline
2. Internal
3. External

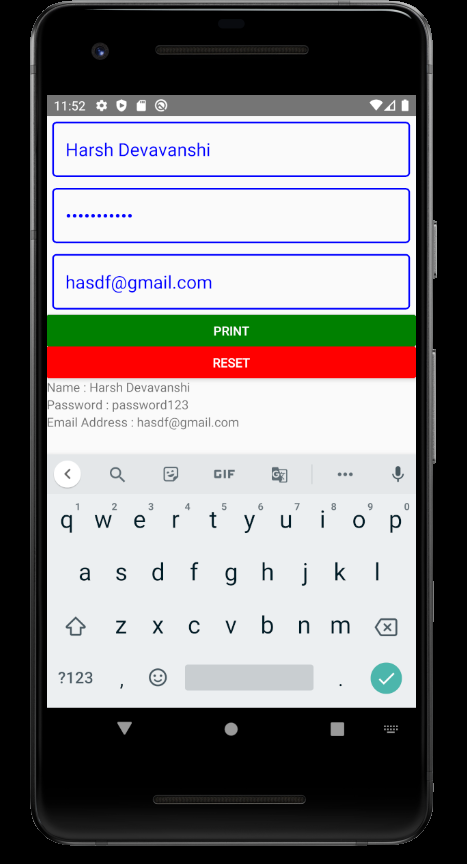




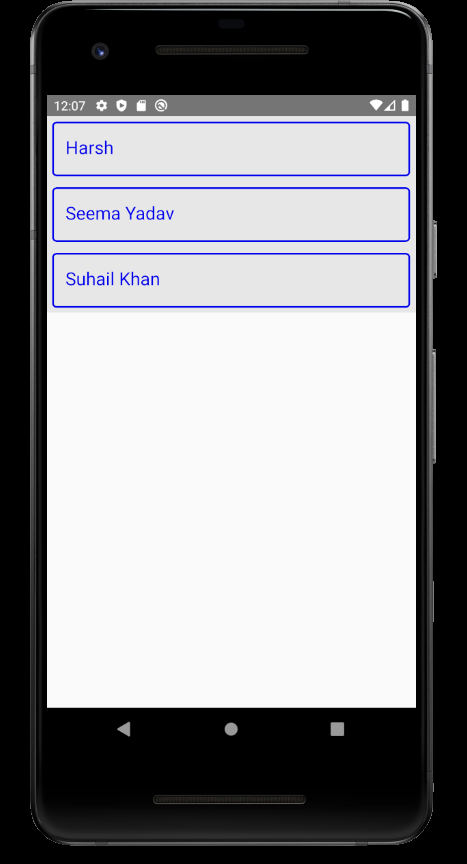
Lesson 12 Input Field



Lesson 13 Form Field

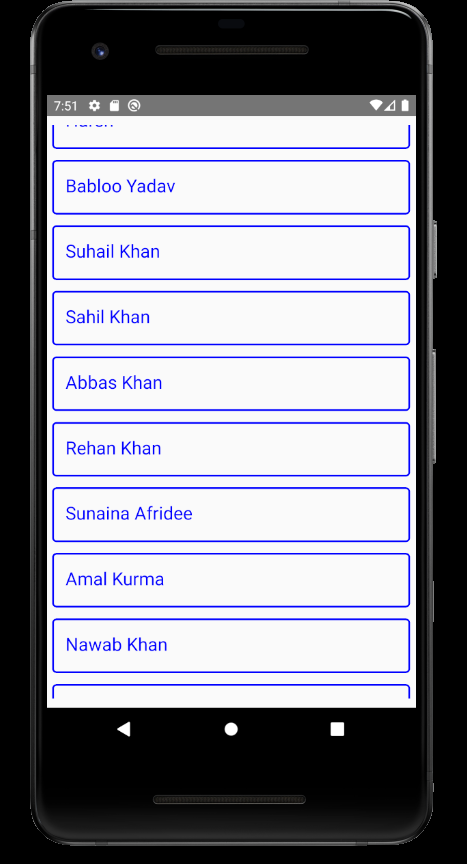


Lesson 14 FlatList



* Flatlists have scrollview by default.
* Flatlists works effectively when list grows.

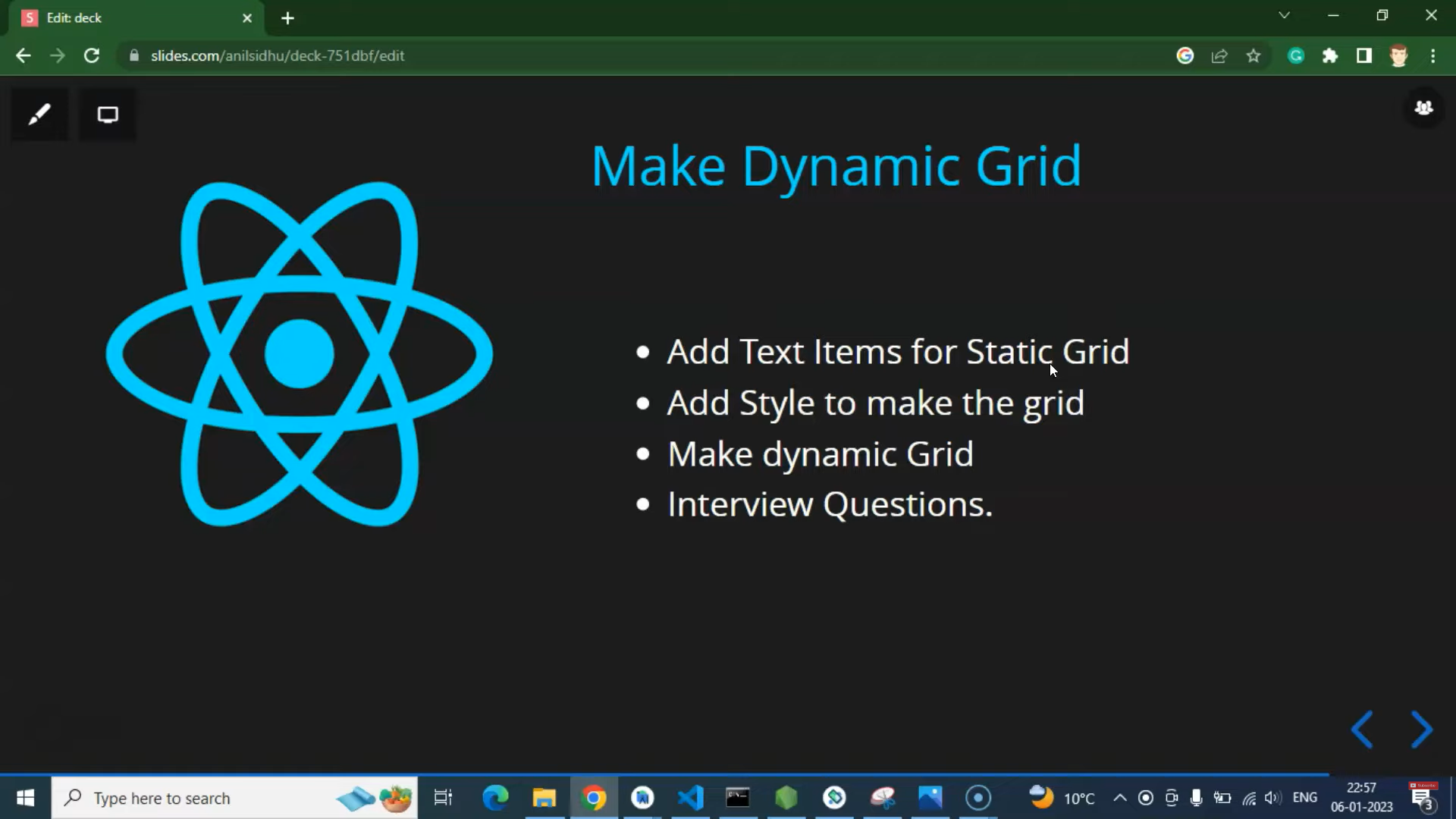
Lesson 15 List With Map Function

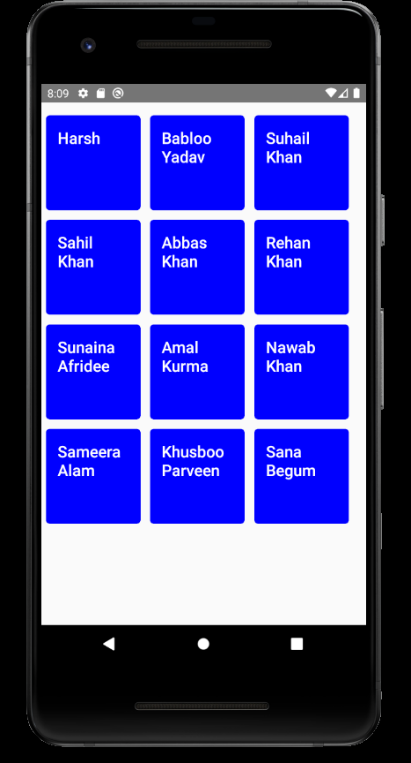


In order to scroll over the list we need to put the list in the ScrollView component of react-native.

When list grows performance of list decreases.

Lesson 16 Dynamic Grid





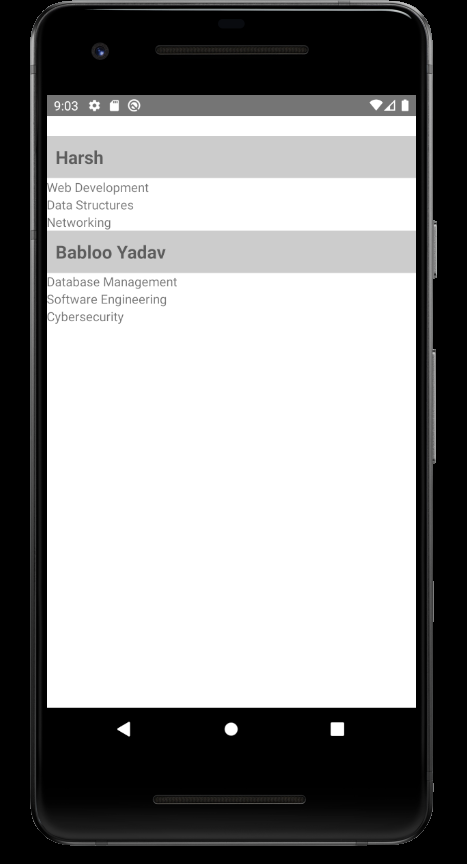
Lesson 17 Component In Loop With FlatList





Lesson18 SectionList

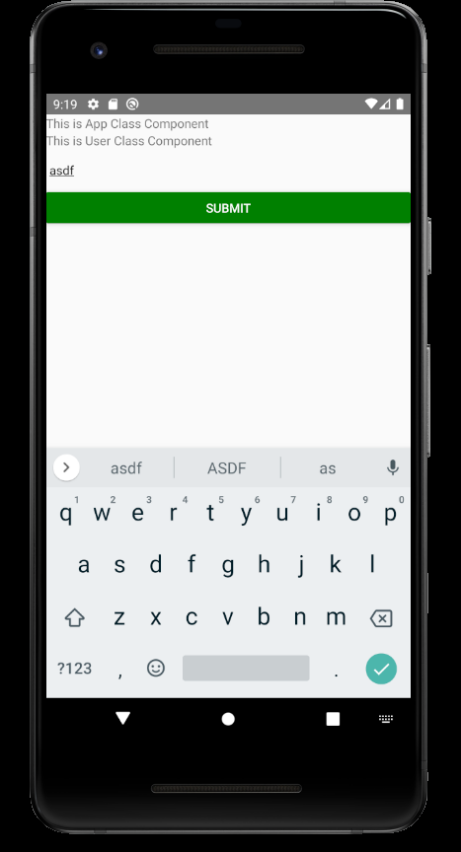
The SectionList component expects the sections prop to be an array of objects where each object represents a section and contains data and title properties.



Lesson 19 Class Component

Why we have render method in class component?

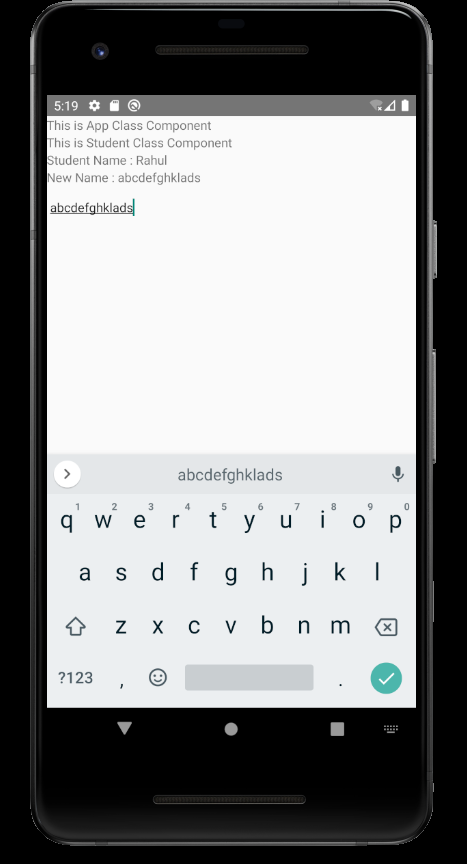
Cause we can’t return anything in class. Return is functionality of functions. That is why we use render from there we can return our jsx.



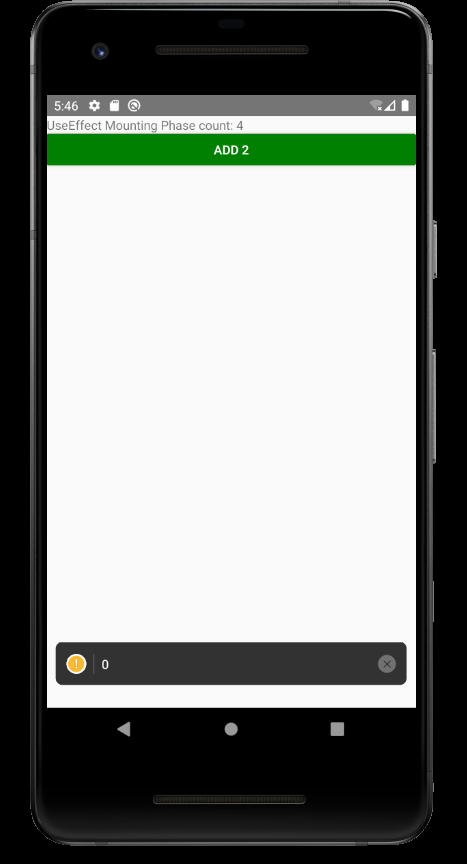


Lesson 20 Props and State in Class component

If we use state without using super() keword in class component we will get error because this.state is the attribute declared in the Component class. Which is parent to inherit and take this property we have to use super().





Lesson 21 LifeCycle Methods

3 Phases

Mount

Update

Unmount

Can be implemented in functional components as well using hooks.

Lesson 22 Hooks in React-Native

Hooks are new addition in React 16.8. They let us use state and other features without writing a class.

Hooks are only used in functional components.

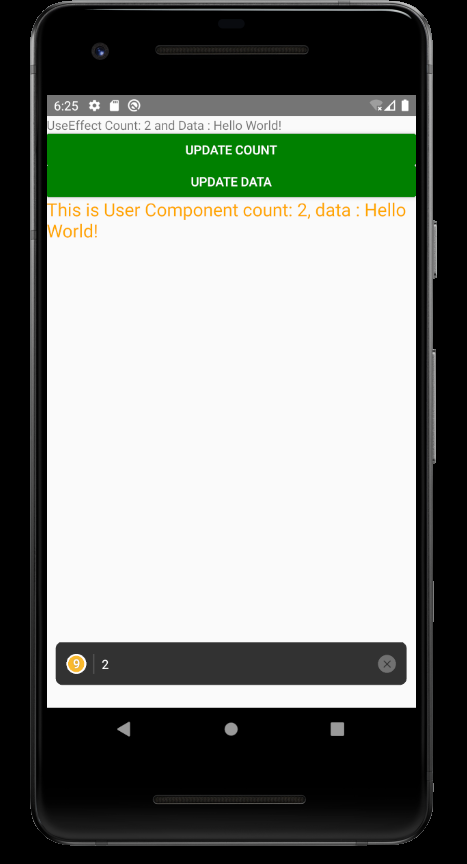
Lesson 23 UseEffect Hook in React Native

This hook is used for lifecycle methods. Takes one callback function and array.

useEffect will get called first on mounting then whenever state/props changes then it will get called.

With blank array it will run only on mounting.



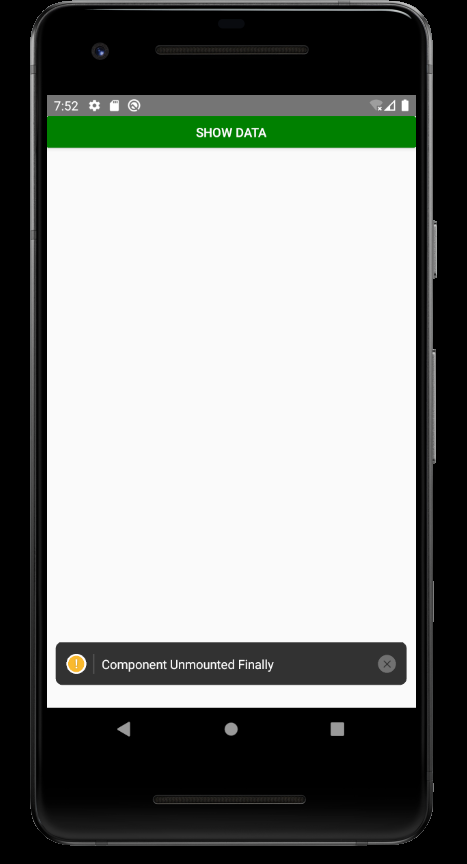
Lesson 24 useEffect as a componentDidUpdate Life Cycle method  


Lesson 25 Show hide component

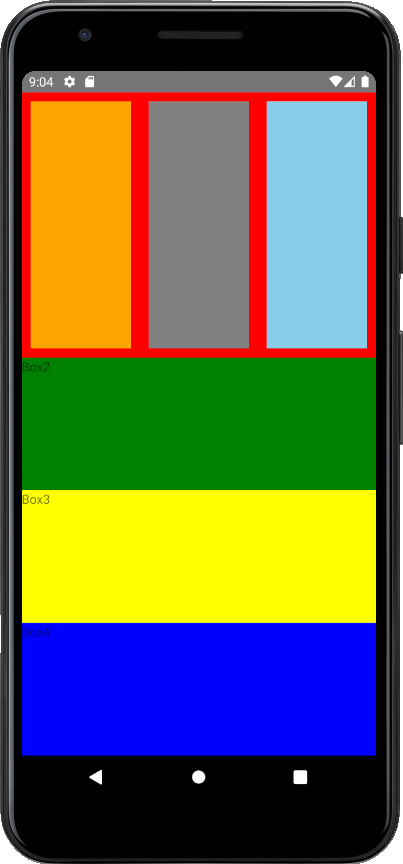




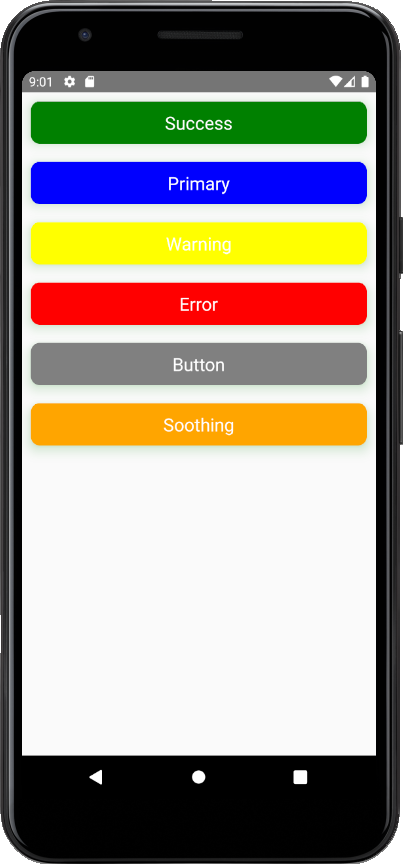
Lesson 26 useEffect with Component Unmound



Lesson 27 Responsive Web Design Using Flex



Lesson 28 TouchableHighlight



Lesson 29 Radio Button

