**React Native Notes**

**1) Installation - Windows**

1. Download & install Java SDK - For any android apps

2. Download & install Node JS - To generate application bundle & package management

3. Download & install Python2

4. Android Studio

5. React Native Command Line Tool - npm install -g react-native-cli

6. Initialize a project - react-native init [projectName]

7. Open android studio and open the existing project.

8. Tools - Android - AVD Manager

9. Set Environment Variables

JAVA\_HOME - C:\Program Files\Java\jdk-version

Path - C:\Users\[username]\AppData\Local\Android\sdk\platform-tools

10. Open cmd from Project directory. Run react-native run-android

11. Set up ESLint

1. npm install -g eslint

2. Install package Control - packagecontrol.io/installation

3. ESLint config - npm install --save-dev eslint-config-rallycoding

4. Create a .eslintrc file in root project directory

{

"extends" : "rallycoding"

}

5. Install sublime linter - CTRL+Shift+P in sublime - Install Package - sublimelinter, sublimelinter-contrib-eslint

**2) Getting started with the Project**

1. index.android.js and index.ios.js are the entry files to our applications. When the app first starts, it will execute all of the code in these files.

index.android.js

//Import a library to help create a component

import React from 'react';

import ReactNative from 'react-native';

//Create a component

//Render it to the device

**3) Differences between React and React Native**

**React** - Knows how a component should behave. Knows how to take a bunch of components and make them work together.

**React Native** - Knows how to take the output from a component and place it on a screen. Provides default core components (image, text)

**4) Create a component with JSX - babeljs.io**

//Create a component

const App = () => {

return (

<Text>Some Text</Text>

);

};

**5) Registering a Component**

//Render it to the device

ReactNative.AppRegistry.registerComponent('[projectname]', () => App);

\*[projectname] should match with the name of our project directory name.

\*For every React application that we create, we must register atleast one component.

**6) Destructuring Imports**

import { Text, AppRegistry } from 'react-native';

const App = () => (

<Text>Some Text</Text>

);

AppRegistry.registerComponent('[projectname]', () => App);

**7) The Header Component**

\* Always create one component per file.

\* Create a new folder - src - components

\* Create a new file. Eg - header.js

\* Only the root component uses 'AppRegistry'

//Import libraries for making a component

import React from 'react';

import { Text } from 'react-native';

//Make a component

const Header = () => {

return <Text> Albums! </Text>

};

//Make the component availabe to other parts of the app

export default Header;

**8) Consuming File Exports**

\*Component Nesting - Add Header component to App Component

import Header from './src/components/header';

const App = () => (

<Header />

);

**9) Styling with React Native**

const Header = () => {

const { textSyle } = styles;

return <Text style={textStyle}> Albums! </Text>

};

const styles = {

textStyle : {

fontSize : 20

}

};

\*View Tag - More like a wrapper

import { Text, View } from 'react-native';

const Header = () => {

const { textStyle, viewStyle } = styles;

return (

<View style={viewStyle}>

<Text style={textStyle}> Albums! </Text>

</View>

);

};

const styles = {

viewStyle : {

backgroundColor : '#F8F8F8'

},

textStyle : {

fontSize : 20

}

};

**10) Introduction to Flexbox**

**\*justifyContent** - Position elements in the vertical direction

**\*alignItems** - Position elements in the horizontal direction

justifyContent : 'flex-end', 'center', 'flex-start'

alignItems : 'flex-end', 'center', 'flex-start'

const styles = {

viewStyle : {

backgroundColor : '#F8F8F8',

justifyContent : 'center',

alignItems : 'center',

height : 60,

paddingTop : 15,

shadowColor : '#000',

shadowOffset : { width: 0, height: 2},

shadowOpacity : 0.2,

elevation : 2,

position : 'relative'

},

textStyle : {

fontSize : 20

}

};

**11) Reusing Components - props**

const Header = (props) => {

const { textStyle, viewStyle } = styles;

return (

<View style={viewStyle}>

<Text style={textStyle}> {props.headerText} </Text>

</View>

);

};

//index.android.js

const App = () => (

<Header headerText={'Albums'} />

);

**12) Sourcing Data - Album Data here**

\* First we need an API or so, from where we could retrieve the data - http://rallycoding.herokuapp.com/api/music\_albums

\* Let's have two components - Album List and Album Detail

Album List will render several Album Details components

**13) List Component Boilerplate**

\* Create a new file for AlbumList Component - AlbumList.js

import React from 'react';

import {View, Text} from 'react-native';

const AlbumList = () => {

return (

<View>

<Text>Album List!!!</Text>

</View>

);

};

export default AlbumList;

\* We cannot return two components in App. It's like

return 1;

return 2;

\* When we have multiple tags to be returned, we can tag them inside a single View Tag.

//index.android.js

import { AppRegistry, View } from 'react-native';

import AlbumList from './src/components/AlbumList';

const App = () => (

<View>

<Header headerText={'Albums'} />

<AlbumList />

</View>

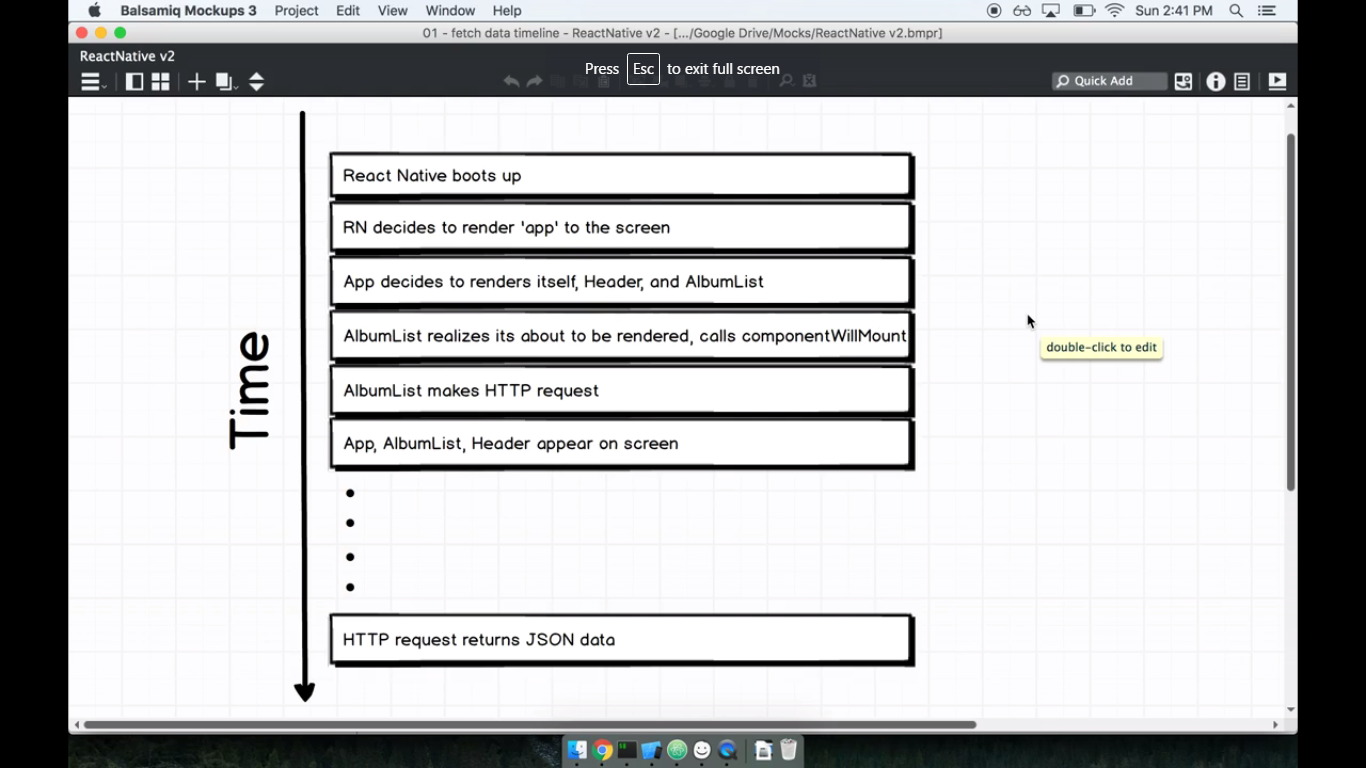
);

14) Class Based Components - Making an HTTP request

\* All the components we made so far are functional components. They are functions that return some amount of JSX to be displayed.

* Differences between Functional and Class Based Component

|  |  |
| --- | --- |
| **Functional Component** | **Class Component** |
| Used for presenting static data | Used for dynamic sources of data |
| Can’t handle fetching data | Handles any data that might change (fetching data, user events, etc) |
| Easy to write | Knows when it gets rendered to the device (useful for data fetching)  More code to write |
| const Header = ( ) => {  return <Text>Hi There!</Text>  } | class Header extends Component {  render () {  return <Text>Hi There!</Text>  }  } |

****

