Contents

[Basics 1](#_Toc69112271)

[Play sound: 1](#_Toc69112272)

[Stylesheet: 2](#_Toc69112273)

[createSwitchNavigator: 2](#_Toc69112274)

[In App.js 2](#_Toc69112275)

[In Homescreen component write goToBuzzerScreen 2](#_Toc69112276)

[create a prop called color in our SoundButton. We can pass the data from HomeScreen to this prop. 2](#_Toc69112277)

[Header Component 3](#_Toc69112278)

[TextInput: 3](#_Toc69112279)

[Image 4](#_Toc69112280)

# Basics

A React Native App is made up of several components.

All components are inherited from the base class call Component (defined in react library.)

App is the main component inside which all other components should live. Text is a component defined in React Native library which allows us to show text on screen.

render() function displays the returned components on the screen. Components can have properties defined on them - like Text has a property called style. We can write javascript inside JSX tags, inside { }.

We learned how React Native uses components to make app development declarative –

Everything inside React native is made up of components - including the app itself. –

We learned how components are created using class and extending the Component class already defined in React library –

We learned about some of the React Native components –

Text, View, Button and their properties - We learned how to create our own custom React Native component.

View is a component defined in React native. It is empty container which can be contain other components.

The render function displays/renders the component returned by the functon. render() can return only one component. We return more than one component in react native by nesting them inside a View Component.

The title and color are called the properties of the Button component. Title is used to display the text inside the button. color is used to add color to the button.

## Play sound:

Audio Class library defined in expo-av package.

  playSound = async () => {

    await Audio.Sound.createAsync(

      { uri: 'http://soundbible.com/mp3/Buzzer-SoundBible.com-188422102.mp3' },

      { shouldPlay: true }

    );

  }

## Stylesheet:

import {StyleSheet } from 'react-native';

 <View style= {styles.textContainer}>

const styles = StyleSheet.create({

  textContainer:{

    backgroundColor: 'blue',

  },

## createSwitchNavigator:

### In App.js

import { createAppContainer, createSwitchNavigator } from 'react-navigation';

var AppNavigator = createSwitchNavigator({

  Screen1: HomeScreen,

  Screen2: BuzzerScreen,

});

const AppContainer = createAppContainer(AppNavigator);

### In Homescreen component write goToBuzzerScreen

 goToBuzzerScreen = (buzzercolor) => {

    this.props.navigation.navigate('Screen2', { color: buzzercolor });

  };

 <TouchableOpacity

          style={[styles.button, { backgroundColor: 'red' }]}

          onPress={() => {

            this.goToBuzzerScreen('red');

          }}>

          <Text style={styles.buttonText}>Team 1</Text>

        </TouchableOpacity>

### create a prop called color in our SoundButton. We can pass the data from HomeScreen to this prop.

#### In buzzer screen:

<SoundButton color={this.props.navigation.getParam('color')}/>

#### In home screen:

goToBuzzerScreen = (buzzercolor) => {

    this.props.navigation.navigate('Screen2', { color: buzzercolor });

  };

#### In sound button:

 style={[styles.button, { backgroundColor: this.props.color }]}

# Header Component

import { Header } from 'react-native-elements';

<Header

          backgroundColor={'#9c8210'}

          centerComponent={{

            text: 'Monkey Chunky',

            style: { color: '#fff', fontSize: 20 },

          }}

# TextInput:

import {Text,} from 'react-native';

  <TextInput

          style={styles.inputBox}

          onChangeText={text => {

            this.setState({ text: text });

          }}

          value={this.state.text}

           placeholder="write your query"

            keyboardType="default"

        />

# Image

import {

  Image,

} from 'react-native';

 <Image

          style={styles.imageIcon}

          source={{

            uri:

              'https://www.shareicon.net/data/128x128/2015/08/06/80805\_face\_512x512.png',

          }}

        />

## Bar Code Scanner- Wireless library app

import \* as Permissions from 'expo-permissions';

import { BarCodeScanner } from 'expo-barcode-scanner';

 getCameraPermissions = async (id) =>{

      const {status} = await Permissions.askAsync(Permissions.CAMERA);

      this.setState({

        /\*status === "granted" is true when user has granted permission

          status === "granted" is false when user has not granted the permission

        \*/

        hasCameraPermissions: status === "granted",

        buttonState: id,

        scanned: false

      });

    }

define three states in our application: hasCameraPermissions: null => this will tell if the user has granted camera permission to the application scanned: false => this will tell if the scanning has completed or not scannedData: '' => this will hold the scanned data that we get after scanning

Then write function

getCameraPermission -which can request for camera permission. Note that this function needs to be asynchronous because it takes time for the user to give camera permission to the application.

The Permission component which we imported has a pre-defined function called askAsync() which can request for various permissions. We will use this to request for camera permission inside our function and change the state of hasCameraPermissions.

Note that askAsync() returns an object with 'status' key containing the status of the permission granted by the user. If the user grants permission, status changes to 'granted' .

Note: {status} automatically extracts the value from the object with key 'status' .

 getCameraPermissions = async (id) =>{

      const {status} = await Permissions.askAsync(Permissions.CAMERA);

      this.setState({

        /\*status === "granted" is true when user has granted permission

          status === "granted" is false when user has not granted the permission

        \*/

        hasCameraPermissions: status === "granted",

        buttonState: id,

        scanned: false

      });

    }

### Bar Code Scanner

Now, we want to return a BarCodeScanner component when the button is clicked and user has given camera permissions.

Create state called buttonState: ‘normal’

BarCodeScanner component automatically starts scanning using the Camera. It has a prop called onBarCodeScanned which can call a function to handle data received after scanning.

We want to call this function only when scanned is false.

write a function called handleBarCodeScanned which is called when scan is completed. This function automatically receives the type of bar code scanned and the data inside the barcode. We can set the scannedData here to be equal to the data received after scanning. Once the scan has been completed, we also want to set the scanned state to true. We also want to change state for the button to make it back to normal when the scan is completed.

handleBarCodeScanned = async({type, data})=>{

      const {buttonState} = this.state

      if(buttonState==="BookId"){

        this.setState({

          scanned: true,

          scannedBookId: data,

          buttonState: 'normal'

        });

      }

      else if(buttonState==="StudentId"){

        this.setState({

          scanned: true,

          scannedStudentId: data,

          buttonState: 'normal'

        });

      }

    }

### defaultNavigationOptions

defaultNavigationOptions field returns some navigation options which is used in the app. One of the navigation options is tabBarIcon. tabBarIcon returns components which you want to use as icon in your bottom tab navigation.

Inside createBottomNavigator, there is an object as a second argument. This object contains a field called defaultNavigationOptions - which currently returns an empty object. Note that the function gets passed navigation object by default.

We set the tabBarIcon field inside defaultNavigationOption. The tabBarIcon will return a component (depending on the tab which is selected).

Navigation object which is passed down during function call, contains routeName property inside its state. routeName property changes depending on which screen is active in the app. If Transaction Screen is active, routeName equals "Transaction" If Search Screen is active, routeName equals "Search".

we also add empty Image object when each tab is active.

const TabNavigator = createBottomTabNavigator({

  Transaction: {screen: TransactionScreen},

  Search: {screen: SearchScreen},

},

{

  defaultNavigationOptions: ({navigation})=>({

    tabBarIcon: ()=>{

      const routeName = navigation.state.routeName;

      console.log(routeName)

      if(routeName === "Transaction"){

        return(

          <Image

          source={require("./assets/book.png")}

          style={{width:40, height:40}}

        />

        )

      }

      else if(routeName === "Search"){

        return(

          <Image

          source={require("./assets/searchingbook.png")}

          style={{width:40, height:40}}

        />)

      }

    }

  })

}

);