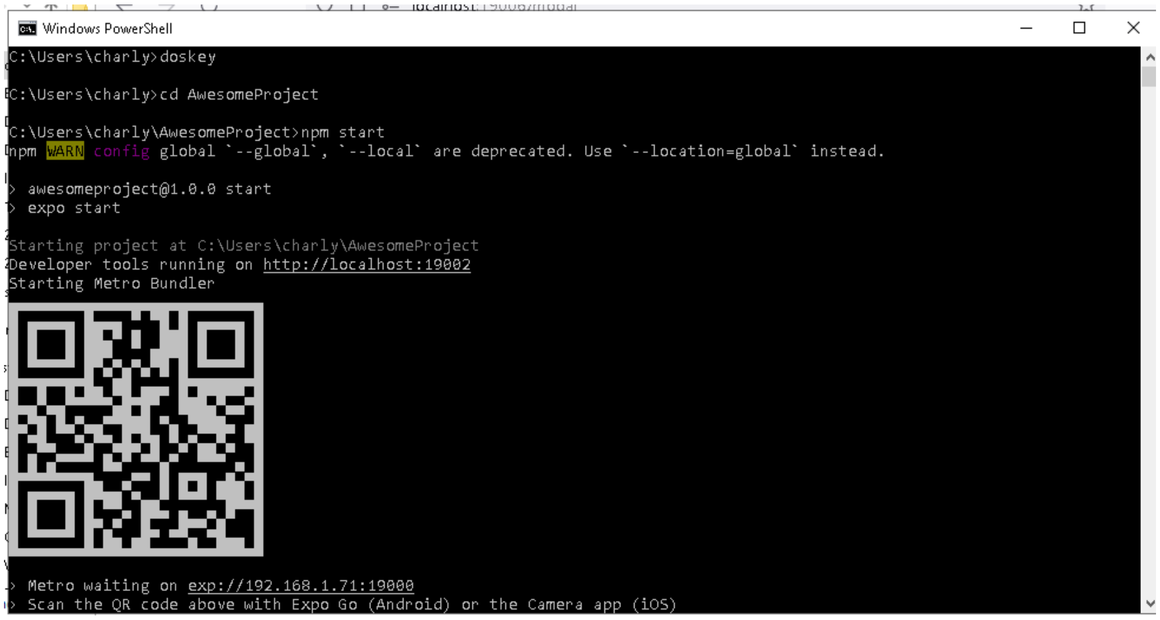
Day 2 Exercises

Exercise 1

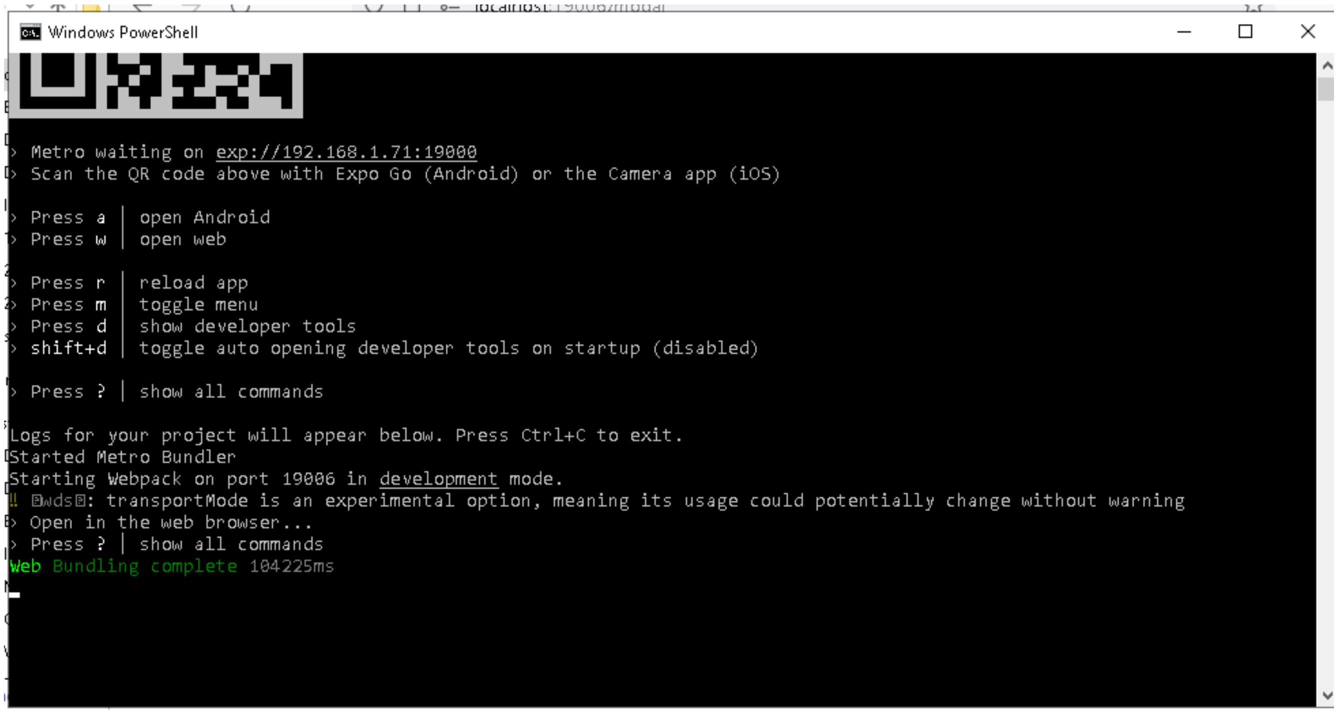
Time: 15 mins.

1. Download and install Nodejs on your computer.
   I. Goto https://nodejs.org/es/download/ and download the correct file for you OS (Note on Linux Ubuntuu users: use your snap store instead the site)
   II. Run the installer (in the case of linux users beware on the instructions when unpacking the compresed file)
2. Install expo application
   I. use the npm install command from your Command Line Interface: **npm install -g expo-cli**
3. Create a project named AwesomeProject using expo
   I. on the command line: **`expo init Awesome Project`**
4. Start the nodejs server
   I. On the command line change to the "AwesomeProject" directory
   II. start the server using npm start

   You should see the following:



```
C:\Users\charly>doskey

C:\Users\charly>cd AwesomeProject

C:\Users\charly\AwesomeProject>npm start
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.

> awesomeproject@1.0.0 start
> expo start

Starting project at C:\Users\charly\AwesomeProject
Developer tools running on http://localhost:19002
Starting Metro Bundler

› Metro waiting on exp://192.168.1.71:19000
› Scan the QR code above with Expo Go (Android) or the Camera app (iOS)
```



```
› Metro waiting on exp://192.168.1.71:19000
› Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

› Press a │ open Android
› Press w │ open web

› Press r │ reload app
› Press m │ toggle menu
› Press d │ show developer tools
› shift+d │ toggle auto opening developer tools on startup (disabled)

› Press ? │ show all commands

Logs for your project will appear below. Press Ctrl+C to exit.
Started Metro Bundler
Starting Webpack on port 19006 in development mode.
‼ wds: transportMode is an experimental option, meaning its usage could potentially change without warning
› Open in the web browser...
› Press ? │ show all commands
Web Bundling complete 104225ms
```

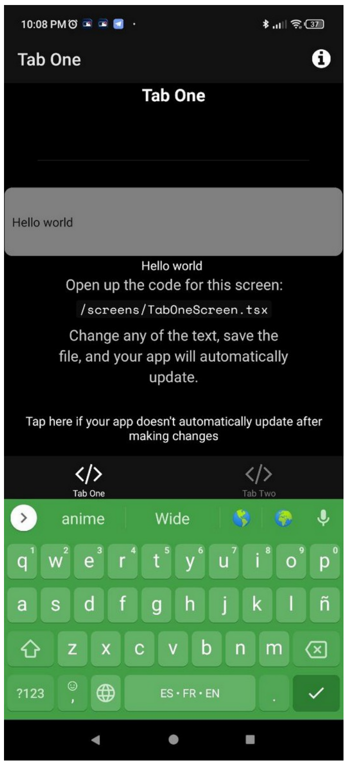
5. Review the application on your browser and your phone

    I. Remember to use the letter w in the console and the propper one for your telephone type

Exercise 2

Time: 15mins.

On the tab1 object create a text field to capture the name of the user. Create also a text tag to display the name the user typed on the text field.



Tab One

Hello world

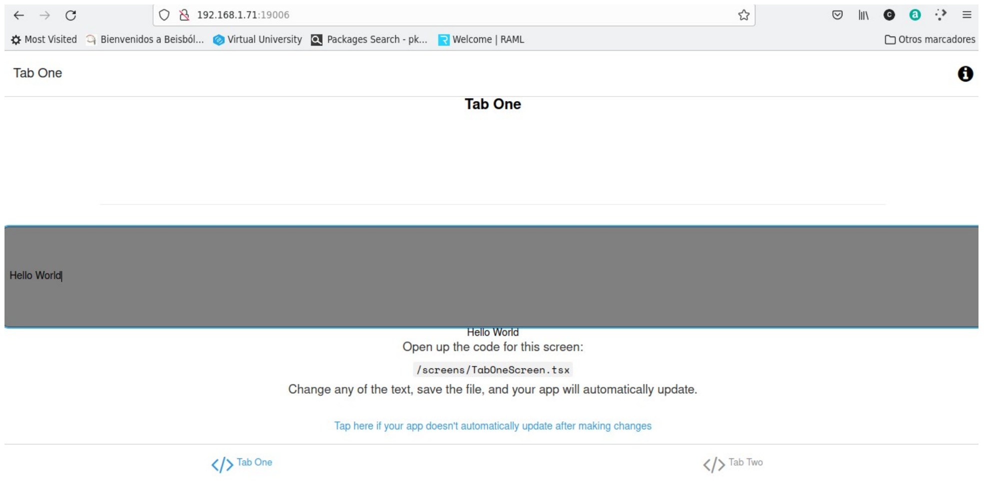
Hello world

Open up the code for this screen:

/screens/TabOneScreen.tsx

Change any of the text, save the file, and your app will automatically update.

Tap here if your app doesn't automatically update after making changes



Tab One

Hello World

Hello World

Open up the code for this screen:

/screens/TabOneScreen.tsx

Change any of the text, save the file, and your app will automatically update.

Tap here if your app doesn't automatically update after making changes

Play with styling all the components.

Answer:

TabOneScreen.tsx

```
import { StyleSheet, TextInput } from 'react-native';
import {useState} from 'react';

import EditScreenInfo from '../components/EditScreenInfo';
import { Text, View } from '../components/Themed';
import { RootTabScreenProps } from '../types';

export default function TabOneScreen({ navigation }: RootTabScreenProps<'TabOne'>)
{
const [myText, setMyText] = useState("My Original Text");
  return (

    <View style={styles.container} >
      <Text style={styles.title}>Tab One</Text>
      <View style={styles.separator} lightColor="#eee"
darkColor="rgba(255,255,255,0.1)" />
        <TextInput style={styles.textInput} placeholder="Your name here"
onChangeText={(newText)=>{setMyText(newText);}} />
        <Text>{myText}</Text>
      <EditScreenInfo style={{flex:1}} path="/screens/TabOneScreen.tsx" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
  },
  title: {
    fontSize: 20,
      flex: 1,
    fontWeight: 'bold',
  },
  separator: {
    marginVertical: 30,
    height: 1,
    width: '80%',
  },
  textInput : {
     backgroundColor:'gray',
       width: "100%",
    borderWidth: 1,
    borderRadius: 10,
    padding: 10,
       border: '1px solid rgb(0.5,0.5,0.5,0.5)',
       flex: 1
  }
});
```

Exercise 3

Time 10 mins.

Remove all the items from the second tab, put an **ImageBackground** component with an image of your preference. Add a couple of buttons in the bottom of the tab page.

When a button is pressed change the image on the background.

Exercise 4

Time 15 mins.

Build Search filter in React.

React code to build a simple search filter functionality to display a filtered list based on the search query entered by the user.

Answer:

1. Declare React states for search input values.
2. Create HTML input text for entering search term and update state in **onChange** function.
3. Add **Array.filter()** on list of items with search term value

App.js

```
import React, { useState } from "react";

import "./styles.css";
```

```
function App() {

  const list = [
    "New York",
    "New Orleans",
    "Merida",
    "Tulum",
    "Miami",
    "Toronto"
  ];

  const [filterList, setFilterList] = useState(list);

  const handleSearch = (event) => {

    if (event.target.value === "") {
      setFilterList(list);
      return;
    }
    const filteredValues = list.filter(
      (item) =>
        item.toLowerCase().indexOf(event.target.value.toLowerCase()) !== -1
    );
    setFilterList(filteredValues);
  };
  return (
    <div className="app">
      <div>
        Search: <input name="query" type="text" onChange={handleSearch} />
      </div>
      {filterList &&
        filterList.map((item, index) => (
          <div key={index}>{item}</div> //Display each item
        ))}
    </div>
  );
}

export default App;
```

Style.css

```
.app {
  font-family: sans-serif;
  display: flex;
  align-items: center;
  justify-content: center;
  flex-direction: column;
  gap: 20px;
  height: 100vh;
  font-family: Cambria, Cochin, Georgia, Times, "Times New Roman", serif;
}
```

Note: This exercise answer is in Reactive JS, a good second exercise should be disscuss and rewrite it to have all React Native properties and advantages.

Exercise 5

Time 5 minutes

Create a grid of 3x3 boxes (<View>) using flex. Every box should have a different color

Answer:

```
import React from 'react';
import {useState,useCallback} from 'react';
import {FlatList,StyleSheet,Text,View,Switch} from 'react-native';
const styles = StyleSheet.create({
  container: {
   flex: 1,
   paddingTop: 22,
   backgroundColor : 'red',
  },
  container2: {
   flex: 1,
   paddingTop: 22,
   backgroundColor : 'lightgray',
  },
  container3: {
   flex: 1,
   paddingTop: 22,
   backgroundColor : 'dodgerblue',
  },
});

const Squares=()=>{
  return(
    <View style={({flex:1, flexDirection:"column"})}>
            <View style={({flex:1, flexDirection:"row"})}>
                  <View style={styles.container}/>
                  <View style={styles.container2}/>
                  <View style={styles.container3}/>
            </View>
            <View style={({flex:1, flexDirection:"row"})}>
                  <View style={styles.container2}/>
                  <View style={styles.container3}/>
                  <View style={styles.container}/>
            </View>
            <View style={({flex:1, flexDirection:"row"})}>
                  <View style={styles.container3}/>
                  <View style={styles.container}/>
                  <View style={styles.container2}/>
            </View>
      </View>
  )
};


export default Squares;
```

Exercise 6

Time 40 minutes

Create a list of countries with their flags and show them in a FlatList with a switch to select them.

```
import React from 'react';
import {useState, useCallback} from 'react';
import { FlatList, StyleSheet, Text, View, Switch} from 'react-native';

const styles = StyleSheet.create({
  container: {
   flex: 1,
   paddingTop: 22
  },
  item: {
    padding: 10,
    fontSize: 18,
    height: 44,
  },
});
const countries = [
          {name: 'Mexico'},
          {name: 'Holland'},
          {name: 'Dominican Republic'},
          {name: 'United States'},
          {name: 'Brazil'},
          {name: 'Argentina'},
          {name: 'Canada'},
          {name: 'Peru'},
          {name: 'Spain'},
          {name: 'Germany'},
        ];

const FlatListBasics = () => {
    const [selectedCountries, setSelectedCountries] = useState([]);
    const handleValueChange = useCallback((value, country) => {
                if(value === true){
                  setSelectedCountries(countries => [...countries, country]);
                }else{
                  setSelectedCountries(countries => countries.filter(selectedCountry => country.na
me !== selectedCountry.name, )
,);}},[]);
  return (

    <View style={styles.container}>

      <FlatList
        data={countries}
        keyExtractor = {item => item.name}
         renderItem ={({item}) => (
            <View styles = {styles.item}>
            <Text> {item.name} </Text>
            <Switch value={!!selectedCountries.find(country => country.name === item.nam
e)}
                  onValueChange={selected => {handleValueChange(selected, item)}}/>
          </View>
        )
      } />

    </View>
  )
};

export default FlatListBasics;
```

Note: When copying and pasting some invalid characters can be set instead of the space char, for Android rendering this could lead to a trouble. So don't copy/paste instead of making your own solution (or at least typing yourself the proposed one). Web rendering seems not to be affected.

Exercise 7

Time 60 minutes

From the previous exercise modify it and create two screens with a button on each one to navigate between them, on the other screen (the one that is not the FlatList one) show two lists, one with the countries that have the switch turned on and the list of all countries on different view, each one of the half of the screen.

```
import React from 'react';
import {useState, useCallback} from 'react';
import { Alert, FlatList, StyleSheet, Text, View, Switch, SafeAreaView, TouchableOpacity} from
'react-native';
import {NavigationContainer } from '@react-navigation/native';
import {createStackNavigator} from '@react-navigation/stack';

const Stack = createStackNavigator();
const styles = StyleSheet.create({
  container: {
   flex: 1,
   paddingTop: 22
  },
  item: {
    padding: 10,
    fontSize: 18,
    height: 44,
  },
  button : {
     pading : 20,
    height : 30,
    textAlign: 'center',
    textAlignVertical : 'center',
    backgroundColor :'dodgerblue'
  }
});

const countries = [
        {name: 'Mexico'},
        {name: 'Holland'},
        {name: 'Dominican Republic'},
        {name: 'United States'},
        {name: 'Brazil'},
        {name: 'Argentina'},
        {name: 'Canada'},
        {name: 'Peru'},
        {name: 'Spain'},
        {name: 'Germany'},
      ];

const SecondScreen = ({navigation, route}) => {
  const [country, setCountry] = useState('');
  const [isRefreshing, setIsRefreshing] = useState(false);
  const newCountry = route.params ? route.params.country : null;
  const [selectedCountries, setSelectedCountries] = useState([]);
  const handleUpdate = useCallback(
    (country, newValue) => {
      if (newValue === true) {
        setSelectedCountries(current => [country]);
      }
      },
    [ setSelectedCountries],
  );

  const handleSubmit = useCallback(() => {
    if (!country) {
      Alert.alert('Please add a contry');
    } else if (selectedCountries.length < 3) {
      Alert.alert('Please choose at least 3 colors');
    } else {
      navigation.navigate('FlatListBasics', {
        newCountry: { countryName: country, countries: selectedCountries },
      });
    }
  }, [country, navigation, selectedCountries]);
  return(
    <View>
```

```
   <TouchableOpacity
     style={styles.button}
     onPress ={() =>{
         navigation.navigate("FlatListBasics");
       }
     }>
        <Text>
    Go to select countries
        </Text>
   </TouchableOpacity>
   <FlatList
      data={selectedCountries}
      keyExtractor = {item => item.name}
       renderItem ={({item}) => (
          <View styles >
             <Text> {item.name} </Text>
          </View>
          )
         }
   />
   </View>

  )

}

const FlatListBasics = ({navigation}) => {
   const [selectedCountries, setSelectedCountries] = useState([]);
   const handleValueChange = useCallback((value, country) => {
             if(value === true){
              setSelectedCountries(countries => [...countries, country]);
             }else{
              setSelectedCountries(countries => countries.filter(selectedCountry => country.na
me !== selectedCountry.name, )
,);}},[]);
  return (
   <View style={styles.container}>
     <FlatList
      data={countries}
      keyExtractor = {item => item.name}
       renderItem ={({item}) => (
          <View styles = {styles.item}>
             <Text> {item.name} </Text>
             <Switch value={!!selectedCountries.find(country => country.name === item.nam
e)}
                         onValueChange={selected => {handleValueChange(selected, item)}}/>
         </View>
        )
      } />
    <TouchableOpacity onPress = {()=>{
      navigation.navigate("SecondScreen", selectedCountries)
    }}>
      <Text>
        Submit
      </Text>
    </TouchableOpacity>
   </View>
  )
};

const MainNavigator = () => {
  return(
  <NavigationContainer>
    <Stack.Navigator>
      <Stack.Screen name='Second' component={SecondScreen}/>
      <Stack.Screen name="FlatListBasics" component={FlatListBasics} />
    </Stack.Navigator>
  </NavigationContainer>
  );
}
export default MainNavigator;
```

Exercise 8

Time: 1:30

Create a simple calculator.

Answer:

Operator.tsx

```
import React, {Component} from 'react';
import { StyleSheet, Text, View, TouchableOpacity } from 'react-native';

export default class OperatorKeys extends Component{
  render(){
    return(
        <View style={styles.calcKey}>
            <TouchableOpacity onPress={()=>{this.props.onClick()}}>
                <Text style={styles.textDisplay}>{this.props.displayKey}</Text>
            </TouchableOpacity>
        </View>
    );
  }
}
const styles = StyleSheet.create({
  calcKey:{
    backgroundColor:"grey",
    flex:.2
  },
  textDisplay:{
    color:"goldenrod",
    textAlign:"center",
    fontSize:36,
  }
});
```

CalcKeys.tsx

```
import React, {Component} from 'react';
import { StyleSheet, Text, View, TouchableOpacity } from 'react-native';

export default class CalcKeys extends Component{
  render(){
    return(
        <View style={styles.calcKey}>
            <TouchableOpacity onPress={()=>{this.props.onClick()}}>
                <Text style={styles.textDisplay}>{this.props.displayKey}</Text>
            </TouchableOpacity>
        </View>
    );
  }
}

const styles = StyleSheet.create({
  calcKey:{
    backgroundColor:"grey",
    flex:.25
  },

  textDisplay:{
    color:"goldenrod",
    textAlign:"center",
    fontSize:36,
  }
});
```

App.tsx

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

import CalcKeys from "./components/CalcKeys.tsx";
import OperatorKeys from "./components/OperatorKeys.tsx";

export default class App extends React.Component {
  constructor(props){
    super(props);
    this.state = {
      display:"",
      numerator:"",
      denominator:"",
      operator:"",
      switchFractionSection:false
      }
  }

  clear(){
    this.setState((state, props) => ({ display:""}));
  }

  evalutate(x,y,operator){
    if(operator == "+" ){
      this.setState((state, props) => ({ display: parseInt(x) + parseInt(y) }))
      this.setState((state, props) => ({ switchFractionSection: false }))
    }else if(operator == "-"){
      this.setState((state, props) => ({ display: parseInt(x) - parseInt(y) }))
      this.setState((state, props) => ({ switchFractionSection: false }))
    }else if(operator == "x"){
      this.setState((state, props) => ({ display: parseInt(x) * parseInt(y) }))
      this.setState((state, props) => ({ switchFractionSection: false }))
    }else {
      this.setState((state, props) => ({ display: parseInt(x) / parseInt(y) }))
      this.setState((state, props) => ({ switchFractionSection: false }))
    }

    //Clear state for next operation
    this.setState((prevState) =>({denominator:""}))
    this.setState((prevState) =>({numerator:""}))
  }

  addNumber(x){
    //show the number clicked on the display. IF this is the first number saved it
is save as the denominator If this is the second number enter, it is saved as the
numerator.
    this.setState((state, props) => ({ display: state.display + x }))
    if(this.state.switchFractionSection ==true){
      this.setState((state, props) =>({denominator:state.denominator + x}))
    }else{
      this.setState((state, props) => ({numerator:state.numerator + x}))
    }
  }

  operatorSymbol(x){

    //If display already have a number and the user press a operator button, then
the current display number is save as the numerator
    if(this.state.numerator == "" && this.state.switchFractionSection == false){
        this.setState((state, props) => ({numerator:this.state.display}))
    }

    this.setState((state, props)=>({display:state.display + x}))
    this.setState((state, props) => ({ operator: x }))
    this.setState((state, props) => ({switchFractionSection:true }))
  }
  render() {
    return (
      <View style={styles.container}>
        <View >
            <Text style={styles.title}>Calculator</Text>
        </View>
        <View style={styles.display}>
            <Text style={styles.title}>{this.state.display}</Text>
        </View>
        <View style={styles.calcKeyRow}>
            <CalcKeys displayKey="1" onClick={()=> this.addNumber("1")} />
            <CalcKeys displayKey="2" onClick={()=> this.addNumber("2")} />
            <CalcKeys displayKey="3" onClick={()=> this.addNumber("3")} />
        </View>
        <View style={styles.calcKeyRow}>
            <CalcKeys displayKey="4" onClick={()=> this.addNumber("4")} />
            <CalcKeys displayKey="5" onClick={()=> this.addNumber("5")} />
            <CalcKeys displayKey="6" onClick={()=> this.addNumber("6")} />
        </View>
        <View style={styles.calcKeyRow}>
            <CalcKeys displayKey="7" onClick={()=> this.addNumber("7")} />
            <CalcKeys displayKey="8" onClick={()=> this.addNumber("8")} />
            <CalcKeys displayKey="9" onClick={()=> this.addNumber("9")} />
        </View>
        <View style={styles.calcKeyRow}>
            <CalcKeys displayKey="0" onClick={()=> this.addNumber("0")} />
            <CalcKeys onClick={()=> this.clear()} displayKey="C" />
            <CalcKeys displayKey="=" onClick={()=>
this.evalutate(this.state.numerator, this.state.denominator,
this.state.operator)} />
        </View>
        <View style={styles.calcKeyRow}>
            <OperatorKeys displayKey="+" onClick={()=> this.operatorSymbol("+")} />
            <OperatorKeys displayKey="-" onClick={()=> this.operatorSymbol("-")} />
            <OperatorKeys displayKey="*" onClick={()=> this.operatorSymbol("x")} />
            <OperatorKeys displayKey="/" onClick={()=> this.operatorSymbol("/")} />
        </View>
      </View>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#696969',
    alignItems: 'center',
    justifyContent:"space-around",
  },

  display:{
    display:"flex",
    justifyContent:"center",
    alignContent:"center",
    backgroundColor:"white",
    width:"70%",
    height:"10%",
    //textAlign:"center",
  },

  title: {
    color:"goldenrod",
    textAlign:"center",
    fontSize:36,
  },

  calcKeyRow:{
    display:"flex",
    flexDirection:"row",
    justifyContent:"space-around",
    alignItems:"center",
    width:"100%",
  }
});
```