REACT NATIVE NOTES - The Practical Guide

- Section 1: Getting Started
- Section 2: React Native Basics (Course Goals App)

1) Intro and Important INFO

1.1) Important Commands

- `npx create-expo-app --template blank project_name` => create new Expo Project (no TypeScript)
- `npm start` => to lauch our App on Expo Go

- Inside the Terminal Window after npm start:
  - Press `a` => to execute app in Android Emulator (Android Studio must be installed)
  - Press `r` => to reload the app

2) React Native Basics

2.1) Basic Components: View, Text and Styling

- `<View></View>` => like div container
- `<Text></Text>` => to include text elements
- `<Button />` => to include a button in the app
- style={{...}} => inline styling
- style={styles.myStyle} => from style-sheet objects
- style properties similar to CSS

> App.js

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

export default function App() {
  return (
    <View style={styles.container}>
      <View>
        <Text>Another text</Text>
      </View>
      <Text
        // inline-style
        style={{ margin: 16, borderWidth: 2, borderColor: 'red', padding: 16 }}
      >
        Hola Sebas
      </Text>
      {/* applying style-sheet object */}
      <Text style={styles.myText}>Hi my friend</Text>
      <Button title='click me' />
    </View>
  );
}

// style-sheet objects
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center'
  },
  myText: {
    margin: 10,
    borderWidth: 3,
    borderColor: 'blue',
    padding: 10
  }
});
```

2.2) Text, View, Button and TextInput

- Basic interaction of core components in React Native
- without proper styling

> App.js

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

export default function App() {
  return (
    <View style={styles.appContainer}>
      <View>
        <TextInput placeholder='Your course goal!' />
        <Button title='Add Goal' />
      </View>
      <View>
        <Text>List of goals...</Text>
      </View>
    </View>
  );
}

// style-sheet objects
const styles = StyleSheet.create({
  appContainer: {
    padding: 50
  }
});
```

2.3) Basic FlexBox and Styling

- Similar to flexbox in CSS
- Properties with camelCase

> App.js

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

export default function App() {
  return (
    <View style={styles.appContainer}>
      <View style={styles.inputContainer}>
        <TextInput style={styles.textInput} placeholder='Your course goal!' />
        <Button title='Add Goal' />
      </View>
      <View>
        <Text>List of goals...</Text>
      </View>
    </View>
  );
}

// style-sheet objects
const styles = StyleSheet.create({
  appContainer: {
    padding: 50
  },
  inputContainer: {
    // flexDirection: 'column', // by default
    flexDirection: 'row',
    justifyContent: 'space-between'
  },
  textInput: {
    borderWidth: 1,
    borderColor: '#cccccc',
    width: '80%', // 80% of available width
    marginRight: 8,
    padding: 8
  }
});
```

2.4) FlexBox deep dive

- **Every `View` by default in React Native uses FlexBox**
- By default the children are organized in a column (from top to bottom)
- In web dev, by default flexbox organizes in a row
- Like CSS, the properties can be overwritten

- Link : FlexBox React Native Info from Docs

> App.js : Starter Code

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

export default function App() {
  return (
    <View style={{ padding: 50 }}>
      <View
        style={{
          backgroundColor: 'red',
          width: 100,
          height: 100,
          justifyContent: 'center',
          alignItems: 'center'
        }}
      >
        <Text>1</Text>
      </View>
      <View
        style={{
          backgroundColor: 'blue',
          width: 100,
          height: 100,
          justifyContent: 'center',
          alignItems: 'center'
        }}
      >
        <Text>2</Text>
      </View>
      <View
        style={{
          backgroundColor: 'green',
          width: 100,
          height: 100,
          justifyContent: 'center',
          alignItems: 'center'
        }}
      >
        <Text>3</Text>
      </View>
    </View>
  );
}
```

App.js : Playing with Flexbox Properties

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

// FLEXBOX
// - allows us to organize content in a page
// - every view by default uses flexbox (column)
export default function App() {
  return (
    <View
      style={{
        padding: 50,
        flexDirection: 'column', // by default
        flexDirection: 'column-reverse',
        flexDirection: 'row-reverse',
        flexDirection: 'row',

        width: '100%',
        height: 500,

        justifyContent: 'center',
        justifyContent: 'space-around',

        alignItems: 'center',
        alignItems: 'stretch'
      }}
    >
      <View
        style={{
          backgroundColor: 'red',
          flex: 1, //? flex elements are distributed according to the number here

          //   width: 100, //? every box can be as big as the text below says
          //   height: 100,

          justifyContent: 'center',
          alignItems: 'center'
        }}
      >
        <Text>1</Text>
      </View>
      <View
        style={{
          backgroundColor: 'blue',
          flex: 2,

          //   width: 100,
          //   height: 100,

          justifyContent: 'center',
          alignItems: 'center'
        }}
      >
        <Text>2</Text>
      </View>
      <View
        style={{
          backgroundColor: 'green',
          flex: 3,

          //   width: 100,
          //   height: 100,

          justifyContent: 'center',
          alignItems: 'center'
        }}
      >
        <Text>3</Text>
      </View>
    </View>
  );
}
```

2.5) Basic Interaction Button, Input and Event Handlers

- `TextInput - onChangetext` => when user types
- `Button - onPress` => when we press there
- Important: don't use 1 state to change other state
- Better Approach: use a function inside the set function state

> App.js : Event Handlers

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

export default function App() {
  const [enteredGoalText, setEnteredGoalText] = useState('');
  const [courseGoals, setCourseGoals] = useState([]);

  function goalInputHandler(enteredText) {
    //console.log(enteredText); //? I can use console.log
    setEnteredGoalText(enteredText);
  }

  function addGoalHandler() {
    //? not a god approach when I use 1 state to modify other
    //setCourseGoals([...courseGoals, enteredGoalText]);

    //? better approach
    setCourseGoals((currentCourseGoals) => [
      ...currentCourseGoals,
      enteredGoalText
    ]);
  }

  return (
    <View style={styles.appContainer}>
      {/* input container */}
      <View style={styles.inputContainer}>
        <TextInput
          style={styles.textInput}
          placeholder='Your course goal!'
          onChangeText={goalInputHandler} //? don't use onChange => returns an Object
        />

        <Button title='Add Goal' onPress={addGoalHandler} />
      </View>

      {/* goals container */}
      <View style={styles.goalsContainer}>
        {courseGoals.map((goal, index) => (
          <Text key={index}>{goal}</Text>
        ))}
      </View>
    </View>
  );
}

// style-sheet objects
const styles = StyleSheet.create({
  appContainer: {
    flex: 1,
    paddingTop: 50,
    paddingHorizontal: 16
  },
  inputContainer: {
    flex: 1,
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
    marginBottom: 24,
    borderBottomWidth: 1,
    borderBottomColor: '#ddd'
  },
  textInput: {
    borderWidth: 1,
    borderColor: '#cccccc',
    width: '70%',
    marginRight: 8,
    padding: 8
  },
  goalsContainer: {
    flex: 9
  }
});
```

2.6) IOS and Android Styling Differences

- Styling here do not cascade!
- Ex: colo applied to `View` does no go to `Text` inside `View`

> App.js : Basic differences between Android and IOS

```
{
  /* goals container */
}
<View style={styles.goalsContainer}>
  {courseGoals.map((goal, index) => (
    //? this styling will work for Android, but not for IOS

    <Text key={index} style={styles.goalItem}>
    {goal}
    </Text>

    //? this styling will work for both

    <View key={index} style={styles.goalItem}>
      <Text style={styles.goalText}>{goal}</Text>
    </View>
  ))}
</View>;

const styles = StyleSheet.create({
  goalItem: {
    margin: 8,
    padding: 8,
    borderRadius: 6,
    backgroundColor: '#5e0acc',
    color: 'white' // styles don't cascade => this is not applied to the text inside View
  },
    goalText: {
    color: 'white'
  }
})
```

2.7) ScrollView Component

- styles like flex cannot be applied to `ScrollView` component
- Instead put `ScrollView` inside a `View` component
- Some props are available for Android, others only for IOS (check docs)
- Some props example:
  - `alwaysBounceVertical` : by default true => vertical animation the moment I want to scroll

- ScrollView in the React Native DOCS

> App.js : ScrollView applied to List with prop alwaysBounceHorizontal

```
import { useState } from 'react';
import {
  StyleSheet,
  Text,
  View,
  Button,
  TextInput,
  ScrollView
} from 'react-native';

export default function App() {
  const [enteredGoalText, setEnteredGoalText] = useState('');
  const [courseGoals, setCourseGoals] = useState([]);

  function goalInputHandler(enteredText) {
    //console.log(enteredText); //? I can use console.log
    setEnteredGoalText(enteredText);
  }

  function addGoalHandler() {
    //? not a god approach when I use 1 state to modify other
    //setCourseGoals([...courseGoals, enteredGoalText]);

    //? better approach
    setCourseGoals((currentCourseGoals) => [
      ...currentCourseGoals,
      enteredGoalText
    ]);
  }

  return (
    <View style={styles.appContainer}>
      {/* input container */}
      <View style={styles.inputContainer}>
        <TextInput
          style={styles.textInput}
          placeholder='Your course goal!'
          onChangeText={goalInputHandler} //? don't use onChange => returns an Object
        />

        <Button title='Add Goal' onPress={addGoalHandler} />
      </View>
      {/*
       * goals container - scrollable
       * syles.goalContainer - cannot be applied to ScrollView
       * better approach - include ScrollView inside a View
       * and apply styles.goalContainer to the parent View
       * check available props for each element, ex: alwaysBounceHorizontal
       */}
      <View style={styles.goalsContainer}>
        <ScrollView alwaysBounceHorizontal={false}>
          {courseGoals.map((goal, index) => (
            //? this styling will work for Android and IOS
            <View key={index} style={styles.goalItem}>
              <Text style={styles.goalText}>{goal}</Text>
            </View>
          ))}
        </ScrollView>
      </View>
    </View>
  );
}

// style-sheet objects
const styles = StyleSheet.create({
  appContainer: {
    flex: 1,
    paddingTop: 50,
    paddingHorizontal: 16
  },
  inputContainer: {
    flex: 1,
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
    marginBottom: 24,
    borderBottomWidth: 1,
    borderBottomColor: '#ddd'
  },
  textInput: {
    borderWidth: 1,
    borderColor: '#cccccc',
    width: '70%',
    marginRight: 8,
    padding: 8
  },
  goalsContainer: {
    flex: 9
  },
  goalItem: {
    margin: 8,
    padding: 8,
    borderRadius: 6,
    backgroundColor: '#5e0acc'
  },
  goalText: {
    color: 'white'
  }
});
```

2.8) FlatList

- For lists with dynamic and scrollable data
- better option than iterating `View + Text`
- IMPORTAT to remember: for long lists => `FlatList` better than `ScrollView`
- 3 important props:
  - `data` : what to render
  - `renderItem` : function with the iteration to render
  - `keyExtractor` : to build a key from the data

> App.js : ScrollView replaced with FlatList

```
import { useState } from 'react';
import {
  StyleSheet,
  Text,
  View,
  Button,
  TextInput,
  ScrollView, // scrolling
  FlatList // scrolling for lists
} from 'react-native';

export default function App() {
  const [enteredGoalText, setEnteredGoalText] = useState('');
  const [courseGoals, setCourseGoals] = useState([]);

  function goalInputHandler(enteredText) {
    //console.log(enteredText); //? I can use console.log
    setEnteredGoalText(enteredText);
  }

  function addGoalHandler() {
    //? not a god approach when I use 1 state to modify other
    //setCourseGoals([...courseGoals, enteredGoalText]);

    //? better approach
    setCourseGoals((currentCourseGoals) => [
      ...currentCourseGoals,
      //{ text: enteredGoalText, key: Math.random().toString() } // I don't need keyExtractor do
      { text: enteredGoalText, id: Math.random().toString() } // I need keyExtractor
    ]);
  }

  return (
    <View style={styles.appContainer}>
      {/* input container */}
      <View style={styles.inputContainer}>
        <TextInput
          style={styles.textInput}
          placeholder='Your course goal!'
          onChangeText={goalInputHandler} //? don't use onChange => returns an Object
        />

        <Button title='Add Goal' onPress={addGoalHandler} />
      </View>

      {/*
       * FlatList
       * scrollable, and work better for lists
       * for lists with dynamic data that are scrollable
       * 3 important props: data, renderItem & keyExtractor
       * data => what to render
       * renderItem => function with the iteration to render
       * keyExtractor => to build a key from the data
       */}
      <View style={styles.goalsContainer}>
        <FlatList
          alwaysBounceHorizontal={false}
          data={courseGoals}
          keyExtractor={(item, index) => {
            return item.id;
          }}
          renderItem={(itemData) => {
            return (
              <View style={styles.goalItem}>
                <Text style={styles.goalText}>{itemData.item.text}</Text>
              </View>
            );
          }}
        />
      </View>
    </View>
  );
}

// style-sheet objects
const styles = StyleSheet.create({
  appContainer: {
    flex: 1,
    paddingTop: 50,
    paddingHorizontal: 16
  },
  inputContainer: {
    flex: 1,
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
    marginBottom: 24,
    borderBottomWidth: 1,
    borderBottomColor: '#ddd'
  },
  textInput: {
    borderWidth: 1,
    borderColor: '#cccccc',
    width: '70%',
    marginRight: 8,
    padding: 8
  },
  goalsContainer: {
    flex: 9
  },
  goalItem: {
    margin: 8,
    padding: 8,
    borderRadius: 6,
    backgroundColor: '#5e0acc'
  },
  goalText: {
    color: 'white'
  }
});
```

2.9) Dividing App into Components + PROPS

- As in a React App

> App.js : main App

```
import { useState } from 'react';
import { StyleSheet, View, FlatList } from 'react-native';
import GoalItem from './components/GoalItem';
import GoalInput from './components/GoalInput';

export default function App() {
  const [courseGoals, setCourseGoals] = useState([]);

  function addGoalHandler(enteredGoalText) {
    setCourseGoals((currentCourseGoals) => [
      ...currentCourseGoals,
      { text: enteredGoalText, id: Math.random().toString() }
    ]);
  }

  return (
    <View style={styles.appContainer}>
      {/* input container */}
      <GoalInput onAddGoal={addGoalHandler} />

      {/* list of goals */}
      <View style={styles.goalsContainer}>
        <FlatList
          alwaysBounceHorizontal={false}
          data={courseGoals}
          keyExtractor={(item, index) => {
            return item.id;
          }}
          renderItem={(itemData) => {
            return <GoalItem text={itemData.item.text} />;
          }}
        />
      </View>
    </View>
  );
}

const styles = StyleSheet.create({
  appContainer: {
    flex: 1,
    paddingTop: 50,
    paddingHorizontal: 16
  },
  goalsContainer: {
    flex: 9
  }
});
```

> components/GoalInput.js

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

function GoalInput(props) {
  const [enteredGoalText, setEnteredGoalText] = useState('');

  function goalInputHandler(enteredText) {
    setEnteredGoalText(enteredText);
  }

  function addGoalHandler() {
    props.onAddGoal(enteredGoalText);
    setEnteredGoalText('');
  }

  return (
    <View style={styles.inputContainer}>
      <TextInput
        style={styles.textInput}
        placeholder='Your course goal!'
        onChangeText={goalInputHandler}
        value={enteredGoalText} // the reset takes place
      />

      <Button title='Add Goal' onPress={addGoalHandler} />
    </View>
  );
}

export default GoalInput;

const styles = StyleSheet.create({
  inputContainer: {
    flex: 1,
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
    marginBottom: 24,
    borderBottomWidth: 1,
    borderBottomColor: '#ddd'
  },
  textInput: {
    borderWidth: 1,
    borderColor: '#cccccc',
    width: '70%',
    marginRight: 8,
    padding: 8
  }
});
```

> components/GoalItem.js

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

function GoalItem(props) {
  return (
    <View style={styles.goalItem}>
      <Text style={styles.goalText}>{props.text}</Text>
    </View>
  );
}

export default GoalItem;

const styles = StyleSheet.create({
  goalItem: {
    margin: 8,
    padding: 8,
    borderRadius: 6,
    backgroundColor: '#5e0acc'
  },
  goalText: {
    color: 'white'
  }
});
```

2.10) Pressable Component and delete items with bind function from JS

- `Pressable` : to make a `View` component pressable. It has the prop `onPress` where we can pass a function to make this happen.
- For the delete we can work with `bind` of JavaScript
- We should pass the `id` of the item as a prop

App.js

```
import { useState } from 'react';
import { StyleSheet, View, FlatList } from 'react-native';
import GoalItem from './components/GoalItem';
import GoalInput from './components/GoalInput';

export default function App() {
  const [courseGoals, setCourseGoals] = useState([]);

  function addGoalHandler(enteredGoalText) {
    setCourseGoals((currentCourseGoals) => [
      ...currentCourseGoals,
      { text: enteredGoalText, id: Math.random().toString() }
    ]);
  }

  function deleteGoalHandler(id) {
    //console.log('DELETE');
    setCourseGoals((currentCourseGoals) => {
      return currentCourseGoals.filter((goal) => goal.id !== id);
    });
  }

  return (
    <View style={styles.appContainer}>
      {/* input container */}
      <GoalInput onAddGoal={addGoalHandler} />

      {/* list of goals */}
      <View style={styles.goalsContainer}>
        <FlatList
          alwaysBounceHorizontal={false}
          data={courseGoals}
          keyExtractor={(item, index) => {
            return item.id;
          }}
          renderItem={(itemData) => {
            return (
              <GoalItem
                text={itemData.item.text}
                id={itemData.item.id}
                onDeleteItem={deleteGoalHandler}
              />
            );
          }}
        />
      </View>
    </View>
  );
}

const styles = StyleSheet.create({
  appContainer: {
    flex: 1,
    paddingTop: 50,
    paddingHorizontal: 16
  },

  goalsContainer: {
    flex: 9
  }
});
```

> GoalInput.js

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

function GoalInput(props) {
  const [enteredGoalText, setEnteredGoalText] = useState('');

  function goalInputHandler(enteredText) {
    setEnteredGoalText(enteredText);
  }

  function addGoalHandler() {
    props.onAddGoal(enteredGoalText);
    setEnteredGoalText('');
  }

  return (
    <View style={styles.inputContainer}>
      <TextInput
        style={styles.textInput}
        placeholder='Your course goal!'
        onChangeText={goalInputHandler}
        value={enteredGoalText} // the reset takes place
      />

      <Button title='Add Goal' onPress={addGoalHandler} />
    </View>
  );
}

export default GoalInput;

const styles = StyleSheet.create({
  inputContainer: {
    flex: 1,
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
    marginBottom: 24,
    borderBottomWidth: 1,
    borderBottomColor: '#ddd'
  },
  textInput: {
    borderWidth: 1,
    borderColor: '#cccccc',
    width: '70%',
    marginRight: 8,
    padding: 8
  }
});
```

> GoalItem.js

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

function GoalItem(props) {
  return (
    <Pressable onPress={props.onDeleteItem.bind(this, props.id)}>
      <View style={styles.goalItem}>
        <Text style={styles.goalText}>{props.text}</Text>
      </View>
    </Pressable>
  );
}

export default GoalItem;

const styles = StyleSheet.create({
  goalItem: {
    margin: 8,
    padding: 8,
    borderRadius: 6,
    backgroundColor: '#5e0acc'
  },
  goalText: {
    color: 'white'
  }
});
```

2.11) Final App

- Final Summary of learned components
  - `StyleSheet` : Provide Styles to the components
  - `View` : like a div component in web apps
  - `Text` : like h1, h2,... p elements

    - `Button` : native button component
    - `ScrollView` : A View component, but scrollable (not recommended for long lists)
    - `FlatList` : Scrollable list, ideal for long dynamic lists
    - `StatusBar` : the status bar of our phone
    - `Pressable` : Pressable component, works with prop `onPress` , where we can pass a function to define what happens when the user press the component
    - `TextInput` : like an input
        - IMPORTANT: `onChangetext` instead of `onChange`
    - `Image` : to include images
        - Important props:
            - `style` : we can include styles
            - `source` : given in the format {require(`path`)}
    - `Modal` : add modal window to the app
        - Important props:
            - `visible` : true or false => can be controlled with a function
            - `animationType` : ex: `slide` => kind of animation, when modal appears
- Important to remember:
    - background color for the whole app can be established in app.json
- Final Code:

> app.json

```json
{
  "expo": {
    "name": "my-app",
    "slug": "my-app",
    "version": "1.0.0",
    "orientation": "portrait",
    "icon": "./assets/icon.png",
    "backgroundColor": "#1e085a",
    "userInterfaceStyle": "light",
    "newArchEnabled": true,
    "splash": {
      "image": "./assets/splash-icon.png",
      "resizeMode": "contain",
      "backgroundColor": "#ffffff"
    },
    "ios": {
      "supportsTablet": true
    },
    "android": {
      "adaptiveIcon": {
        "foregroundImage": "./assets/adaptive-icon.png",
        "backgroundColor": "#ffffff"
      },
      "edgeToEdgeEnabled": true
    },
    "web": {
      "favicon": "./assets/favicon.png"
    }
  }
}
```

> App.js

```
import { useState } from 'react';

import { StyleSheet, View, FlatList, Button } from 'react-native';
import { StatusBar } from 'expo-status-bar';

import GoalItem from './components/GoalItem';
import GoalInput from './components/GoalInput';

export default function App() {
  const [modalIsVisible, setModalIsVisible] = useState(false);
  const [courseGoals, setCourseGoals] = useState([]);

  function startAddGoalHandler() {
    setModalIsVisible(true);
  }

  function endAddGoalHandler() {
    setModalIsVisible(false);
  }

  function addGoalHandler(enteredGoalText) {
    setCourseGoals((currentCourseGoals) => [
      ...currentCourseGoals,
      { text: enteredGoalText, id: Math.random().toString() }
    ]);
    endAddGoalHandler();
  }

  function deleteGoalHandler(id) {
    //console.log('DELETE');
    setCourseGoals((currentCourseGoals) => {
      return currentCourseGoals.filter((goal) => goal.id !== id);
    });
  }

  return (
    <>
      {/* status bar of the phone */}
      <StatusBar style='light' />

      {/* main container of the app */}
      <View style={styles.appContainer}>
        <Button
```

```
          title='Add New Goal'
          color='#3ab11d'
          onPress={startAddGoalHandler}
        />

        {/* input container */}
        {modalIsVisible && (
          <GoalInput
            visible={modalIsVisible}
            onAddGoal={addGoalHandler}
            onCancel={endAddGoalHandler}
          />
        )}

        {/* list of goals */}
        <View style={styles.goalsContainer}>
          <FlatList
            alwaysBounceHorizontal={false}
            data={courseGoals}
            keyExtractor={(item, index) => {
              return item.id;
            }}
            renderItem={(itemData) => {
              return (
                <GoalItem
                  text={itemData.item.text}
                  id={itemData.item.id}
                  onDeleteItem={deleteGoalHandler}
                />
              );
            }}
          />
        </View>
      </View>
    </>
  );
}

const styles = StyleSheet.create({
  appContainer: {
    flex: 1,
    paddingTop: 50,
    paddingHorizontal: 16
    //backgroundColor: '#1e085a' //? defined in app.json
  },

  goalsContainer: {
    marginTop: 20,
    flex: 9
  }
});
```

> GoalInput.js

```
import { useState } from 'react';
import {
  StyleSheet,
  View,
  TextInput,
  Button,
  Modal,
  Image
} from 'react-native';

function GoalInput(props) {
  const [enteredGoalText, setEnteredGoalText] = useState('');

  function goalInputHandler(enteredText) {
    setEnteredGoalText(enteredText);
  }

  function addGoalHandler() {
    props.onAddGoal(enteredGoalText);
    setEnteredGoalText('');
  }

  return (
    <Modal visible={props.visible} animationType='slide'>
      <View style={styles.inputContainer}>
        {/* Adding an Image */}
        <Image
          style={styles.image}
          source={require('../assets/images/sebas_apps.png')}
        />
        <TextInput
          style={styles.textInput}
          placeholder='Your course goal!'
          placeholderTextColor='#120438'
          onChangeText={goalInputHandler}
          value={enteredGoalText} // the reset takes place
        />

        <View style={styles.buttonContainer}>
          <View style={styles.button}>
            <Button title='Add Goal' onPress={addGoalHandler} color='#5e0acc' />
          </View>
          <View style={styles.button}>
          <Button title='Cancel' onPress={props.onCancel} color='#f31282' />
        </View>
      </View>
    </View>
  </Modal>
  );
}

export default GoalInput;

const styles = StyleSheet.create({
  inputContainer: {
    flex: 1,
    flexDirection: 'column',
    justifyContent: 'center',
    alignItems: 'center',
    padding: 40,
    backgroundColor: '#311b6b'
  },
  textInput: {
    borderWidth: 1,
    borderColor: '#e4d0ff',
    backgroundColor: '#e4d0ff',
    color: '#120438',
    borderRadius: 6,
    width: '100%',
    padding: 16
  },
  buttonContainer: {
    flexDirection: 'row',
    marginTop: 20
  },
  button: {
    width: '30%',
    marginHorizontal: 8 // margin left and right
  },
  image: {
    width: 200,
    height: 250,
    margin: 20,
    borderRadius: 30,
    shadowColor: 'white',
    elevation: 10
  }
});
```

> GoalItem.js

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

function GoalItem(props) {
  return (
    <View style={styles.goalItem}>
      <Pressable
        //android_ripple={{ color: '#dddddd' }} //? it does not work anymore
        onPress={props.onDeleteItem.bind(this, props.id)}
        style={({ pressed }) => pressed && styles.pressedItem} //? solution for Android and IOS
      >
        <Text style={styles.goalText}>{props.text}</Text>
      </Pressable>
    </View>
  );
}

export default GoalItem;

const styles = StyleSheet.create({
  goalItem: {
    margin: 8,
    borderRadius: 6,
    backgroundColor: '#5e0acc',

    // to give box shadow
    shadowColor: 'yellow',
    elevation: 10
  },
  pressedItem: {
    opacity: 0.5
  },
  goalText: {
    color: 'white',
    padding: 8,
    textAlign: 'center',
    textTransform: 'capitalize'
  }
});
```