Monkey-Chunky App - A Case Study

What is our GOAL for this MODULE?

We explored a case study of an app designed for students with reading difficulties.

What did we ACHIEVE in the class TODAY?

- Explored a case study of an app designed for students with reading difficulties.
- Designed wireframe for the app (including UI/UX).
- Collected input from the user and displayed it on the screen.

Which CONCEPTS/CODING BLOCKS did we cover today?

- Designing Wireframe
- TextInput Component

How did we DO the activities?

Case Study: The case study is about a student named Zaara. Zaara is a Grade 6 kid who finds it hard to read even grade 1 text. Zaara has a reading disorder which makes it hard for her to read. She needs repeated practice for breaking a word into smaller chunks and joining them to read the complete word.

Information about Learning Disabilities:

There are 44 different kinds of sounds in English - called phonemes. Each word is made up of a combination of these sounds. For example:
The word cat is made up of sounds \c\ \a\ \t\

A phoneme could be made up of a combination of one or more than one letter.

Through minimal practice, most of us have learned to identify these patterns of letters which make up a phoneme, identify them in words and combine them to make words.

Pattern recognition is so natural for us that we have learned to do it almost intuitively.

However, students with reading difficulties find it very hard to identify these patterns. They need a repeated practice of breaking down the words into different chunks, identifying the phoneme sound associated with each chunk, combining the phoneme sounds to pronounce a word.

Often these students need special teachers who would be patient and not get irritated with the students' failed attempts to recognize the chunks and phonemes. With enough practice, these students can read as well as anyone else.

Making the app:

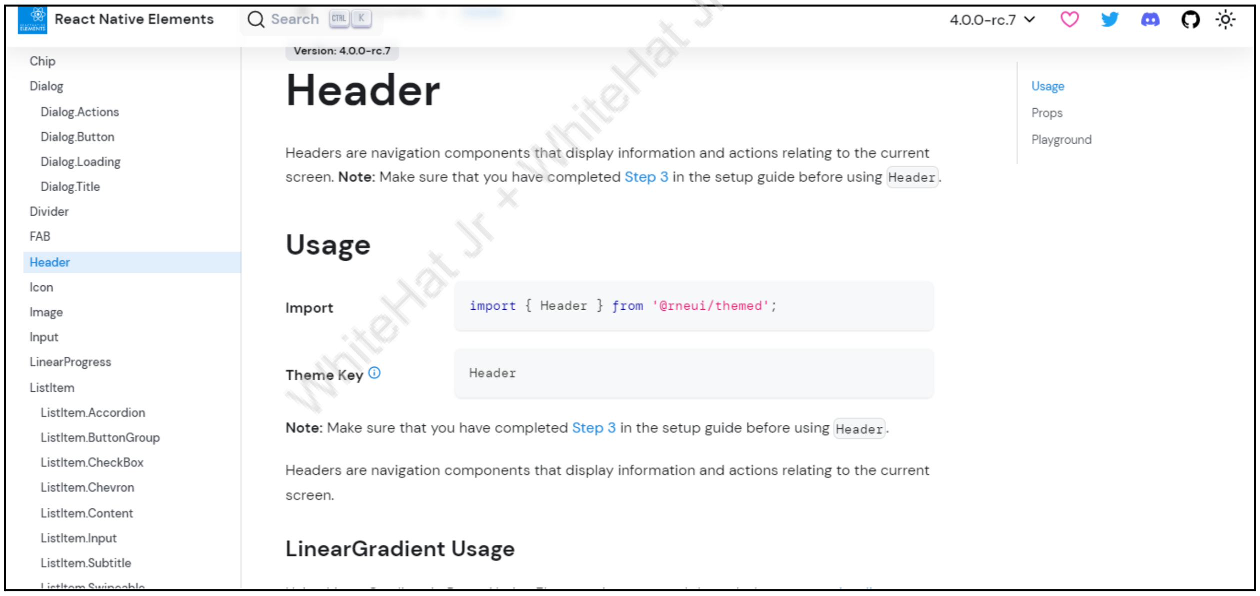
Building a wireframe is also called designing the User Interface (UI) or User Experience (UX). It is a crucial part of any app. It defines how your app will be used by the user and how they will experience it.

Like coding, designing the user interface/user experience (UI/UX) is also an iterative process.

We design a user interface and implement it in the app. After implementation and testing, we might realize that the user experience could be made better by tweaking a few features in our app and we make those changes to the UI/UX.

*Note: We can start our project either on Expo snack online as we were doing for previous projects OR we can do so locally using expo-cli tools we installed in the last class.

- To start the project locally - you need to type: expo init <project name>
- Choose a blank project and let expo install all the expo libraries for the project.
- A project directory with your project name will be created. Then change directory (cd) and open the folder in any code editor.
- You can also run expo start to look at the output of your code.
- Alternatively, you can also use the online snack expo.



Version: 4.0.0-rc.7

Header

Headers are navigation components that display information and actions relating to the current screen. **Note:** Make sure that you have completed Step 3 in the setup guide before using `Header`.

Usage

Import

```
import { Header } from '@rneui/themed';
```

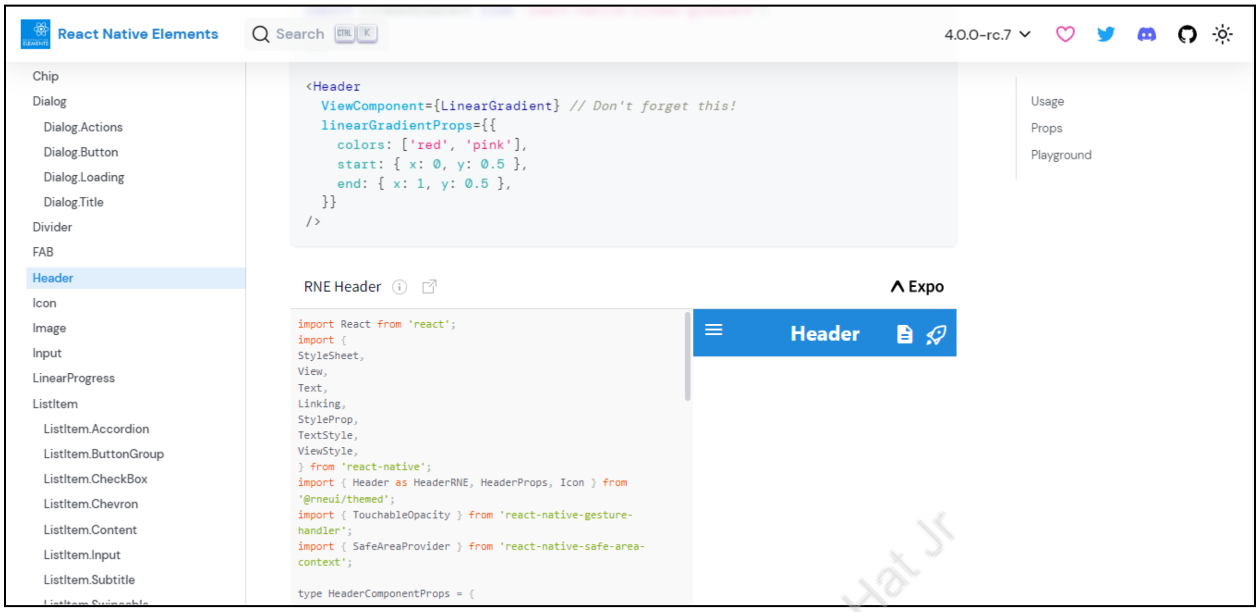
Theme Key

```
Header
```

Note: Make sure that you have completed Step 3 in the setup guide before using `Header`.

Headers are navigation components that display information and actions relating to the current screen.

LinearGradient Usage

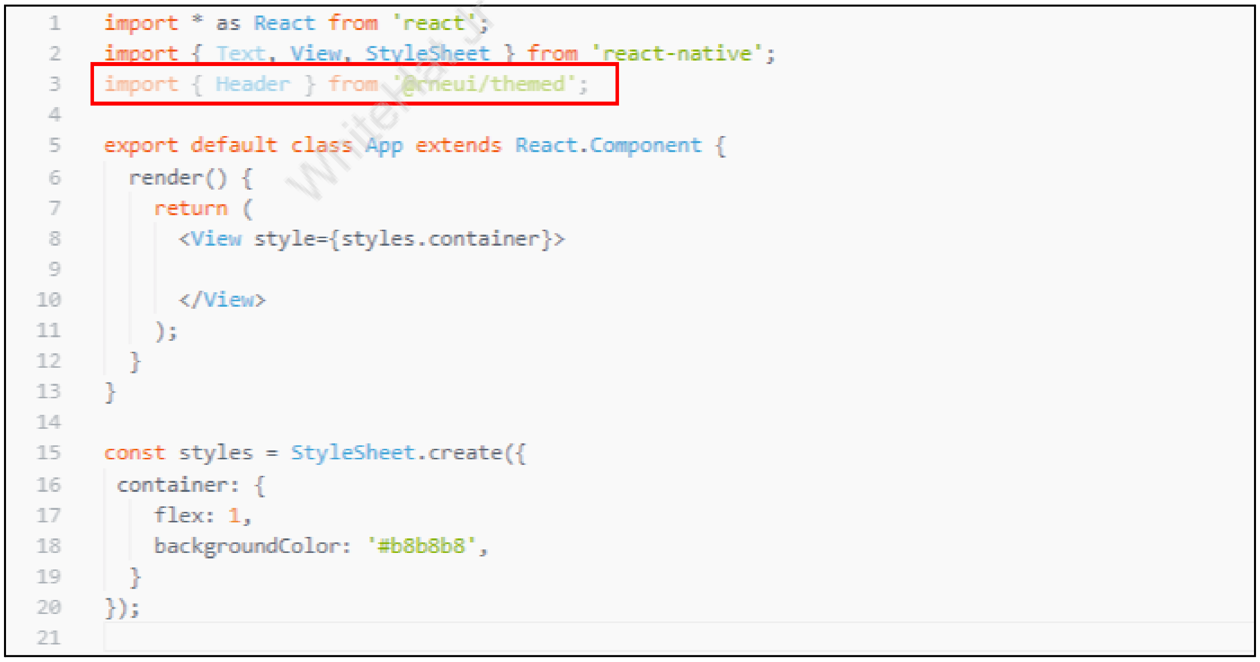


```
<Header
  ViewComponent={LinearGradient} // Don't forget this!
  linearGradientProps={{
    colors: ['red', 'pink'],
    start: { x: 0, y: 0.5 },
    end: { x: 1, y: 0.5 },
  }}
/>
```

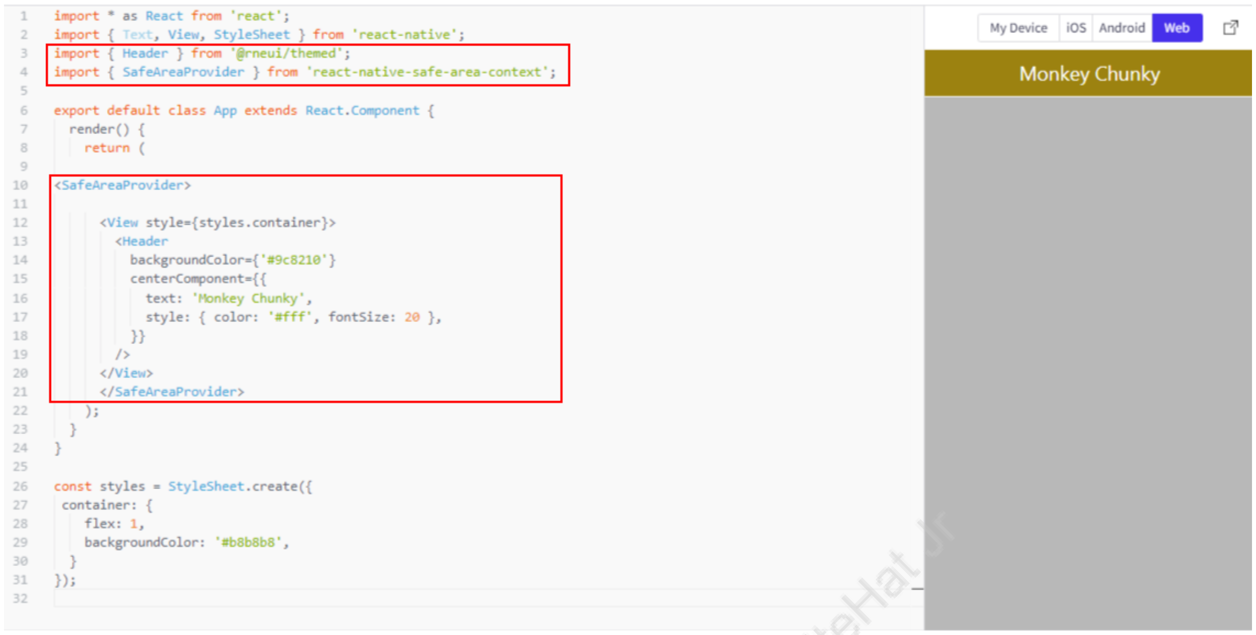
```
import React from 'react';
import {
StyleSheet,
View,
Text,
Linking,
StyleProp,
TextStyle,
ViewStyle,
} from 'react-native';
import { Header as HeaderRNE, HeaderProps, Icon } from
'@rneui/themed';
import { TouchableOpacity } from 'react-native-gesture-
handler';
import { SafeAreaProvider } from 'react-native-safe-area-
context';

type HeaderComponentProps = {
```

*Note: One of the popular React Native UI libraries which developers like to use is 'React Native Elements'. You can learn about the different components available and their props, examples on how to use them through the documentation available.

- Let's use one of their components, 'Header'.

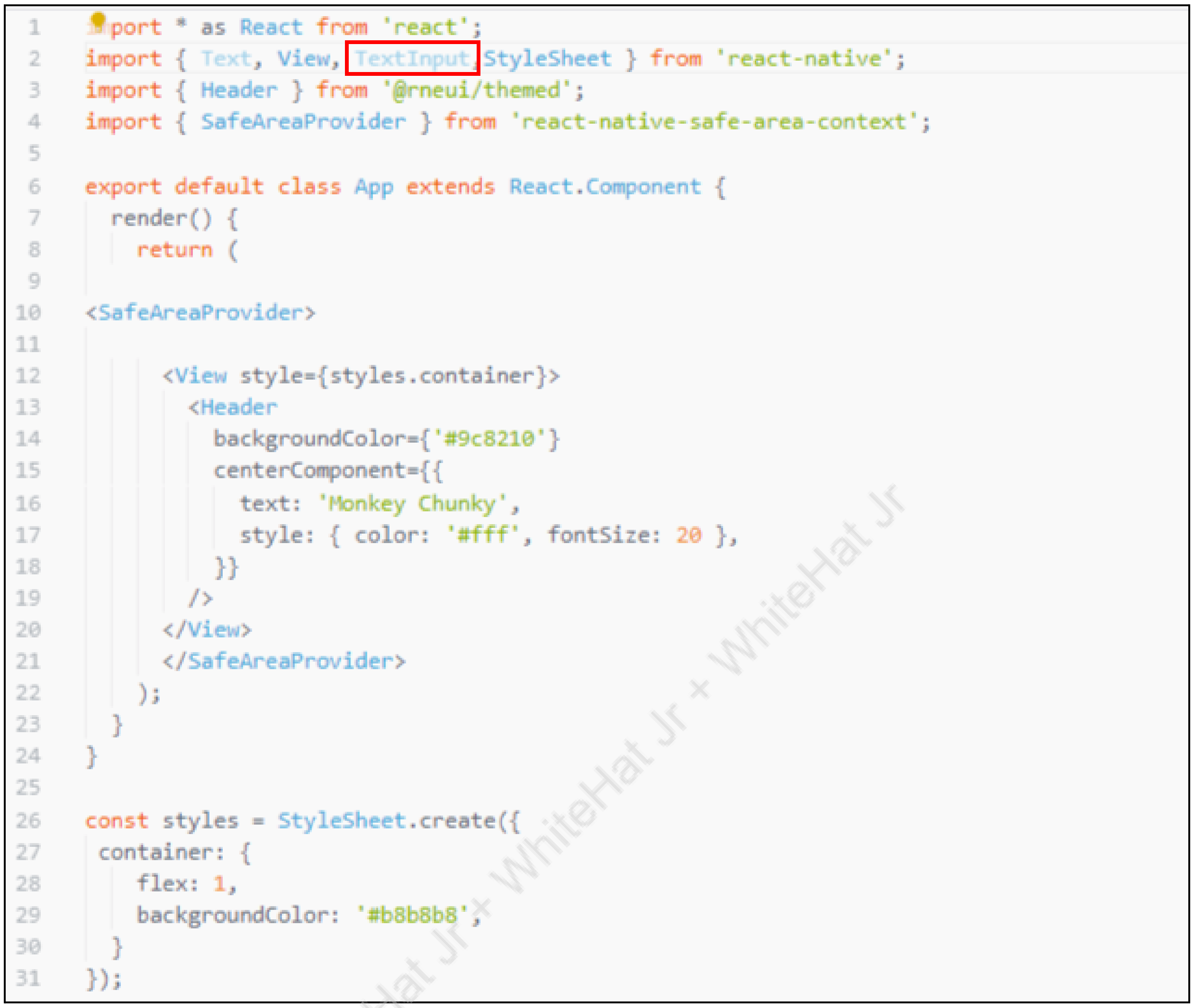


```
import * as React from 'react';
import { Text, View, StyleSheet } from 'react-native';
import { Header } from '@rneui/themed';

export default class App extends React.Component {
  render() {
    return (
      <View style={styles.container}>

      </View>
    );
  }
}

const styles = StyleSheet.create({
 container: {
    flex: 1,
    backgroundColor: '#b8b8b8',
  }
});
```



```
import * as React from 'react';
import { Text, View, StyleSheet } from 'react-native';
import { Header } from '@rneui/themed';
import { SafeAreaProvider } from 'react-native-safe-area-context';

export default class App extends React.Component {
  render() {
    return (

<SafeAreaProvider>

      <View style={styles.container}>
        <Header
          backgroundColor={'#9c8210'}
          centerComponent={{
            text: 'Monkey Chunky',
            style: { color: '#fff', fontSize: 20 },
          }}
        />
      </View>
      </SafeAreaProvider>
    );
  }
}

const styles = StyleSheet.create({
 container: {
    flex: 1,
    backgroundColor: '#b8b8b8',
  }
});
```

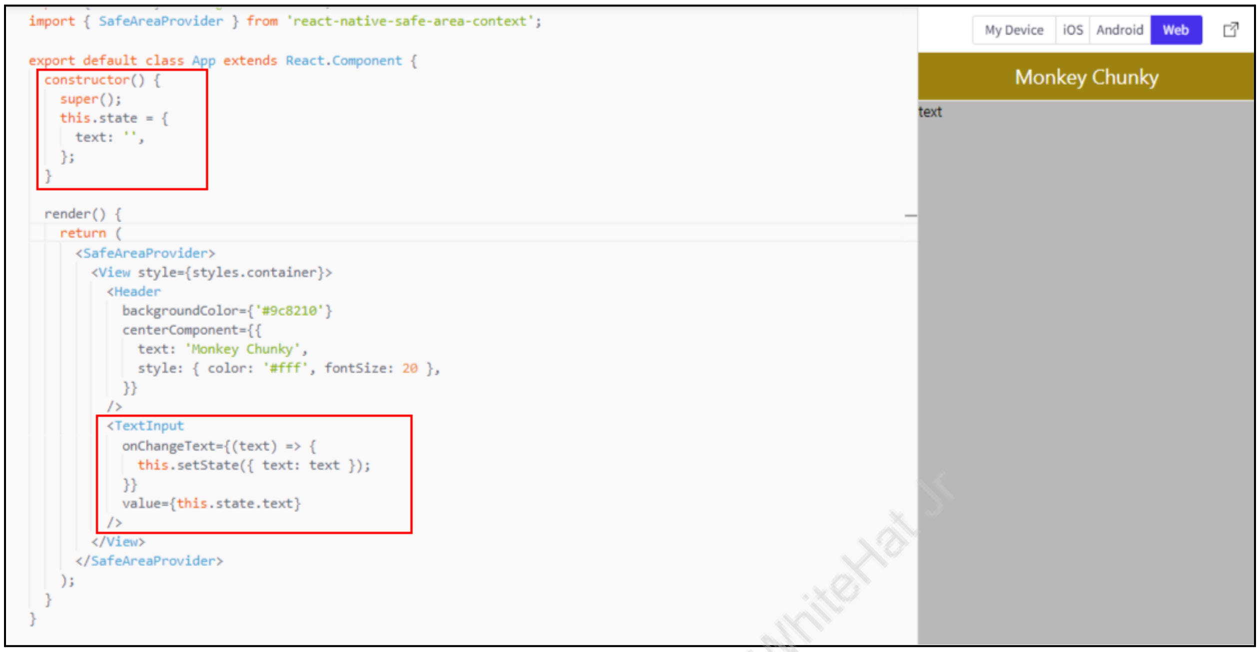
- Now, we need to use a component which will take input from the user.
- There is a React native component called 'TextInput' for this. Using it is slightly tricky.
- First, we import it.



```
import * as React from 'react';
import { Text, View, TextInput, StyleSheet } from 'react-native';
import { Header } from '@rneui/themed';
import { SafeAreaProvider } from 'react-native-safe-area-context';

export default class App extends React.Component {
  render() {
    return (

<SafeAreaProvider>

      <View style={styles.container}>
        <Header
          backgroundColor={'#9c8210'}
          centerComponent={{
            text: 'Monkey Chunky',
            style: { color: '#fff', fontSize: 20 },
          }}
        />
      </View>
      </SafeAreaProvider>
    );
  }
}

const styles = StyleSheet.create({
 container: {
   flex: 1,
   backgroundColor: '#b8b8b8',
 }
});
```

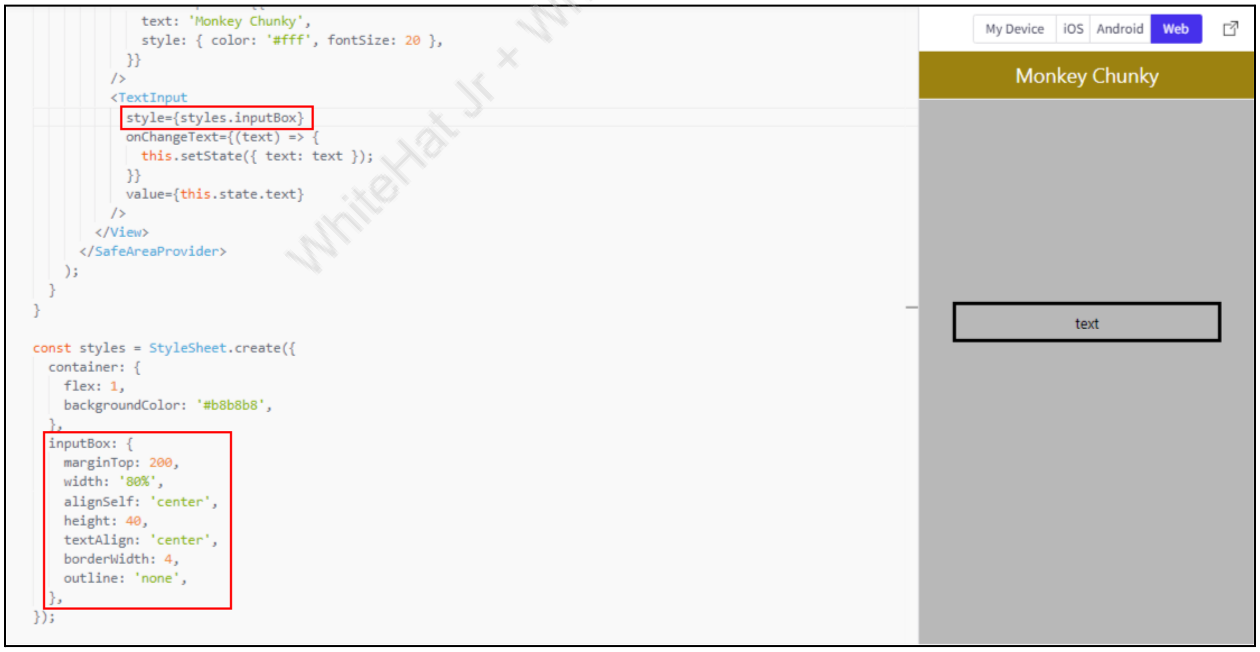
- Let's look at the TextInput documentation, the props that are available and how to use them.

- TextInput takes the value which is passed on to the value prop.
- We have to create a state which constantly updates when a user types text as input.
- The value of 'TextInput' should be the same as 'text state'.



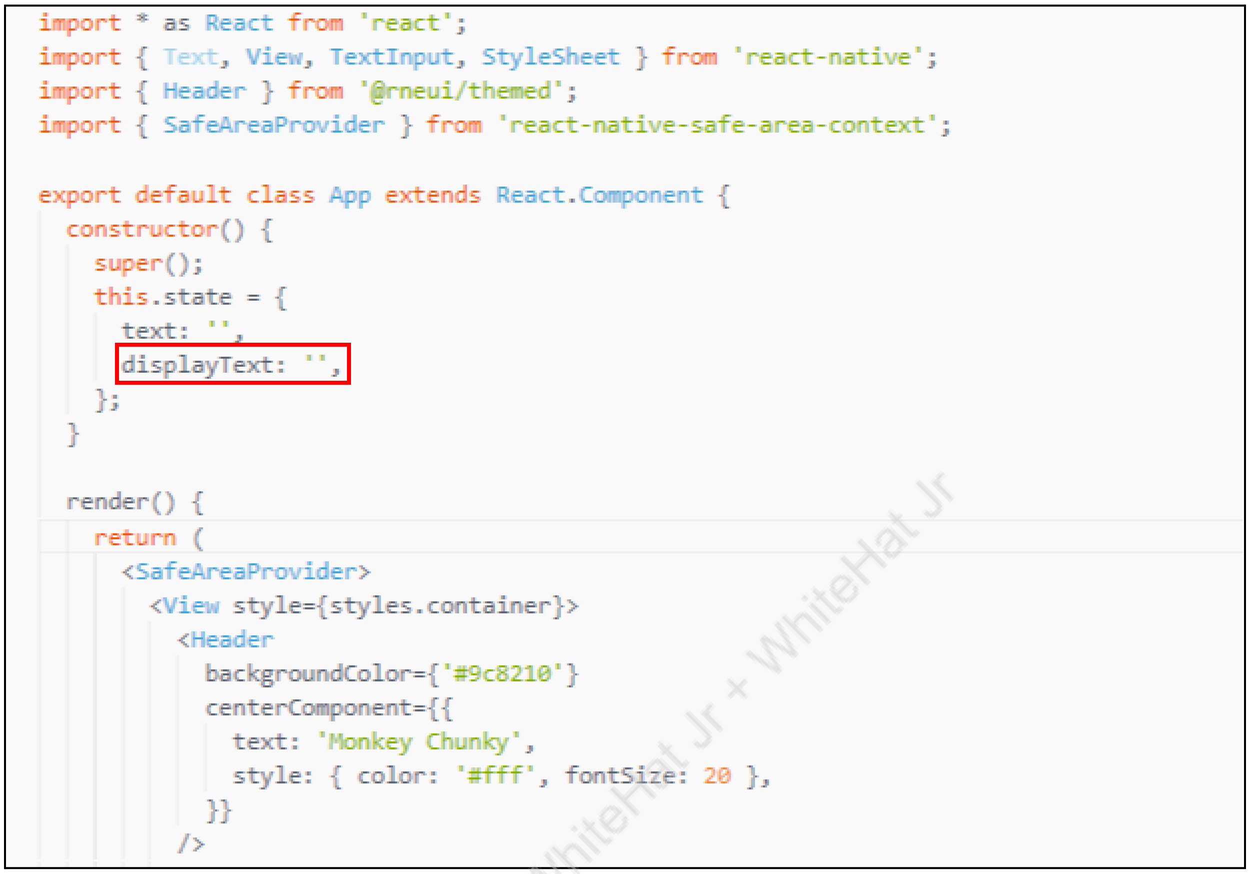
```
import { SafeAreaProvider } from 'react-native-safe-area-context';

export default class App extends React.Component {
  constructor() {
    super();
    this.state = {
      text: '',
    };
  }

  render() {
    return (
      <SafeAreaProvider>
        <View style={styles.container}>
          <Header
            backgroundColor={'#9c8210'}
            centerComponent={{
              text: 'Monkey Chunky',
              style: { color: '#fff', fontSize: 20 },
            }}
          />
          <TextInput
            onChangeText={(text) => {
              this.setState({ text: text });
            }}
            value={this.state.text}
          />
        </View>
      </SafeAreaProvider>
    );
  }
}
```

- Let's add some styling to make it look like an Input Text Box.
- Try to render the typed text in 'TextInput' as a normal displayed Text when a button is pressed.

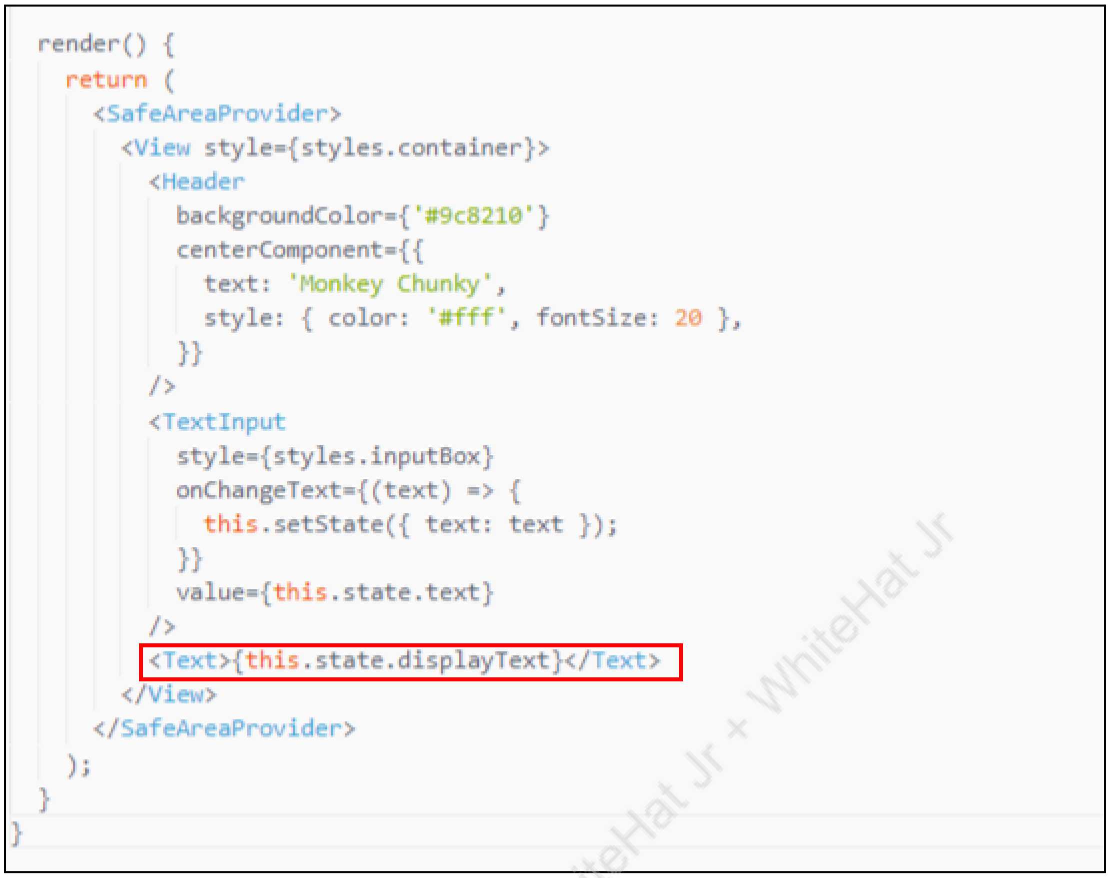


```
              text: 'Monkey Chunky',
              style: { color: '#fff', fontSize: 20 },
            }}
          />
          <TextInput
            style={styles.inputBox}
            onChangeText={(text) => {
              this.setState({ text: text });
            }}
            value={this.state.text}
          />
        </View>
      </SafeAreaProvider>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#b8b8b8',
  },
  inputBox: {
    marginTop: 200,
    width: '80%',
    alignSelf: 'center',
    height: 40,
    textAlign: 'center',
    borderWidth: 4,
    outline: 'none',
  },
});
```

- Let's create another state called 'displayText' which is to be displayed here.

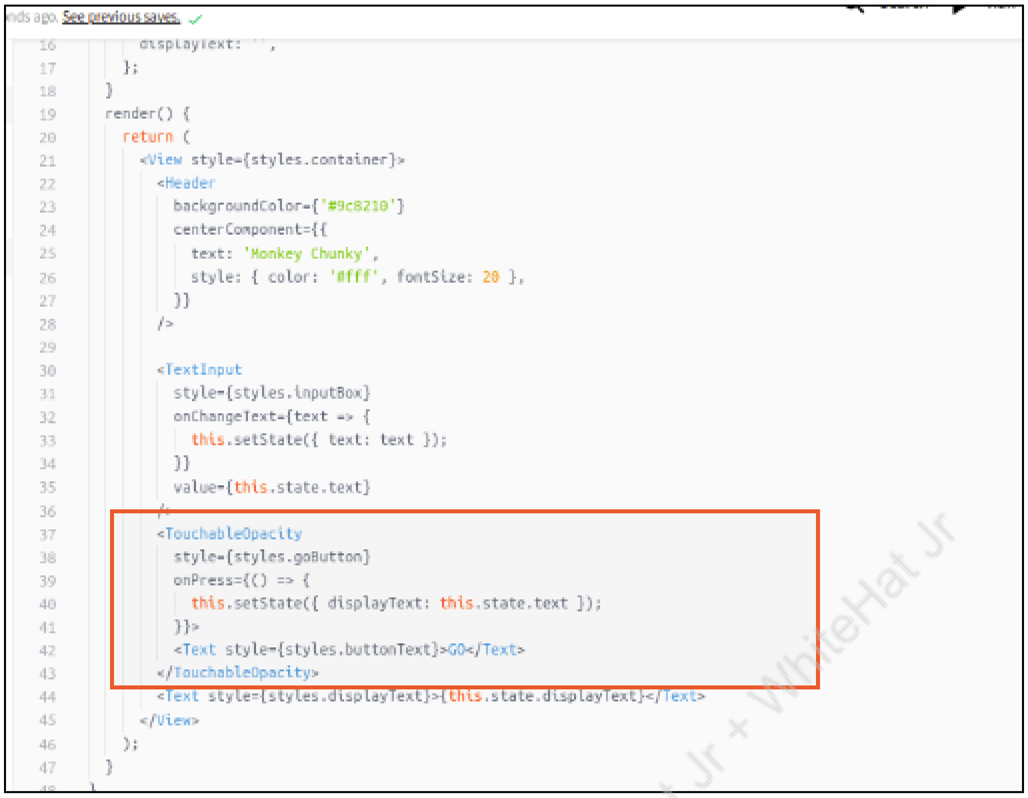


```
import * as React from 'react';
import { Text, View, TextInput, StyleSheet } from 'react-native';
import { Header } from '@rneui/themed';
import { SafeAreaProvider } from 'react-native-safe-area-context';

export default class App extends React.Component {
  constructor() {
    super();
    this.state = {
      text: '',
      displayText: '',
    };
  }

  render() {
    return (
      <SafeAreaProvider>
        <View style={styles.container}>
          <Header
            backgroundColor={'#9c8210'}
            centerComponent={{
              text: 'Monkey Chunky',
              style: { color: '#fff', fontSize: 20 },
            }}
          />
```



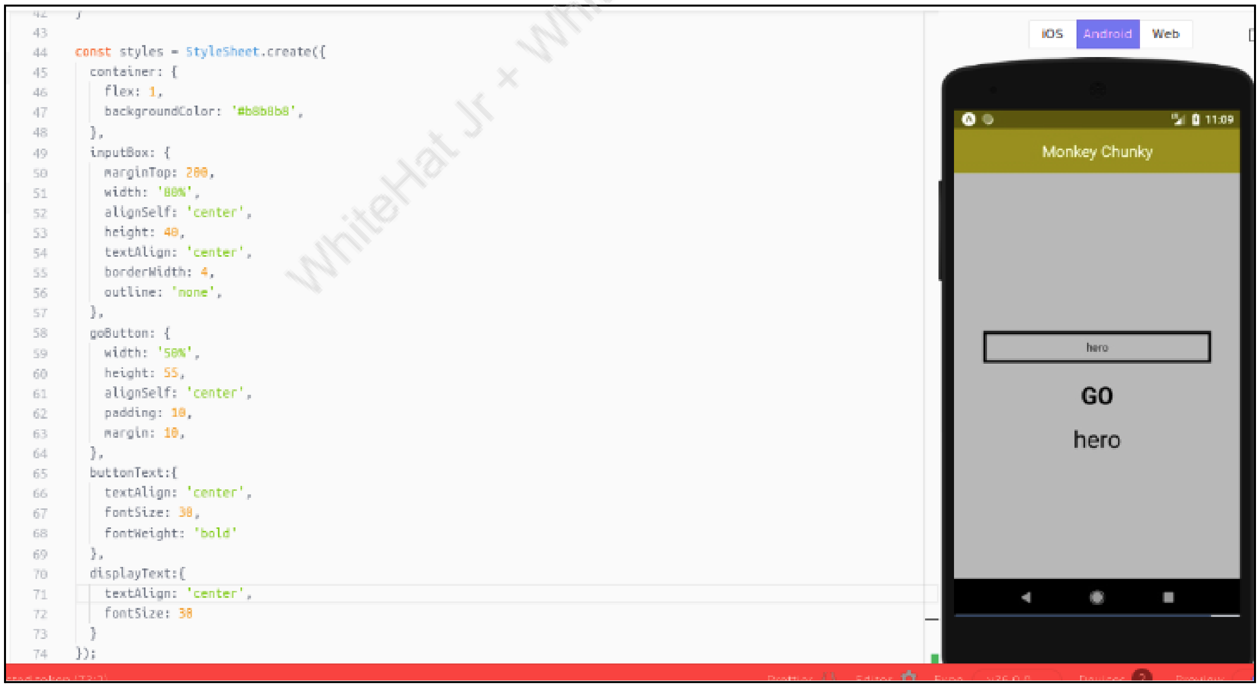
```
render() {
  return (
    <SafeAreaProvider>
      <View style={styles.container}>
        <Header
          backgroundColor={'#9c8210'}
          centerComponent={{
            text: 'Monkey Chunky',
            style: { color: '#fff', fontSize: 20 },
          }}
        />
        <TextInput
          style={styles.inputBox}
          onChangeText={(text) => {
            this.setState({ text: text });
          }}
          value={this.state.text}
        />
        <Text>{this.state.displayText}</Text>
      </View>
    </SafeAreaProvider>
  );
 }
}
```

- Create a button called 'Go' button which updates 'displayText' to the same value as text when the user presses this button.



```
      displayText: '',
    };
  }
  render() {
    return (
      <View style={styles.container}>
        <Header
          backgroundColor={'#9c8210'}
          centerComponent={{
            text: 'Monkey Chunky',
            style: { color: '#fff', fontSize: 20 },
          }}
        />

        <TextInput
          style={styles.inputBox}
          onChangeText={text => {
            this.setState({ text: text });
          }}
          value={this.state.text}
        />
        <TouchableOpacity
          style={styles.goButton}
          onPress={() => {
            this.setState({ displayText: this.state.text });
          }}>
          <Text style={styles.buttonText}>GO</Text>
        </TouchableOpacity>
        <Text style={styles.displayText}>{this.state.displayText}</Text>
      </View>
    );
  }
```

- Add some styling to our buttons and 'displayText'.



```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#b8b8b8',
  },
  inputBox: {
    marginTop: 200,
    width: '80%',
    alignSelf: 'center',
    height: 40,
    textAlign: 'center',
    borderWidth: 4,
    outline: 'none',
  },
  goButton: {
    width: '50%',
    height: 55,
    alignSelf: 'center',
    padding: 10,
    margin: 10,
  },
  buttonText:{
    textAlign: 'center',
    fontSize: 30,
    fontWeight: 'bold'
  },
  displayText:{
    textAlign: 'center',
    fontSize: 30
  }
});
```

What's NEXT?

In the next class, we will learn how to break the word into chunks and display it to the user.

EXTEND YOUR KNOWLEDGE

1. TextInput: https://reactnative.dev/docs/textinput