Structuring React Native Code— Wireless Buzzer App

What we did:

- Used Stylesheet to better name and use CSS styles.
- Exported and imported custom React Native components.
- Wrote structured code for each custom React Native component differently.

How we did it:

In this class, our task was to create a screen which would allow users to pick their team and then navigate to the buzzer button. We used a sheet to style the code as we do in HTML and CSS.

In react native, a good practice is to create a separate styles object using 'StyleSheet.create()' and use the declared styles inside the style prop of the component when rendering.

- First, we import the 'StyleSheet' component defined in React Native.
- Second, we use 'StyleSheet.create()' to create the different styles. The function expects a JSON object with different styles defined on them.
- Note: We use "const" to create a variable which we do not want to change while the program is running. Typically we do not want the style object to change during the program.
- We can call the styles.<name of the style> inside the style prop of a component to give a particular style to the React Native component.

```
import * as React from 'react';
import { Text, View, Button, TouchableOpacity, StyleSheet } from 'react-native';
import {Audio} from 'expo-av';

class SoundButton extends React.Component {
  playSound = async () => {
    await Audio.Sound.createAsync(
      { uri: 'http://soundbible.com/mp3/Buzzer-SoundBible.com-188422102.mp3' },
      { shouldPlay: true }
    );
  }

  render() {
    return (
      <TouchableOpacity
        style={{
          marginLeft: 180,
          borderWidth: 1,
          borderColor: 'rgba(0,0,0,0.2)',
          alignItems: 'center',
          justifyContent: 'center',
          width: 200,
          height: 200,
          backgroundColor: 'red',
          borderRadius: 100,
        }}
        onPress={this.playSound}>
        <Text
          style={{
```

Prettier {} Editor

```
export default class App extends React.Component {
  render() {
    return (
      <View>
        <SoundButton />
      </View>
    );
  }
}

const styles = StyleSheet.create({
  button: {
    marginTop: 200,
    marginLeft: 100,
    borderWidth: 1,
    borderColor: 'rgba(0,0,0,0.2)',
    alignItems: 'center',
    justifyContent: 'center',
    width: 200,
    height: 200,
    backgroundColor: 'red',
    borderRadius: 100,
  },
  buttonText: {
    fontWeight: 'bold',
    fontSize: 20,
  }
});
```

iOS

Press Me

```
class SoundButton extends React.Component {
  playSound = async () => {
    await Audio.Sound.createAsync(
      { uri: 'http://soundbible.com/mp3/Buzzer-SoundBible.com-188422102.mp3' },
      { shouldPlay: true }
    );
  }

  render() {
    return (
      <TouchableOpacity
        style={styles.button}
        onPress={this.playSound}>
        <Text
          style={styles.buttonText}>
          Press Me
        </Text>
      </TouchableOpacity>
    );
  }
}

export default class App extends React.Component {
  render() {
    return (
      <View>
        <SoundButton />
      </View>
```

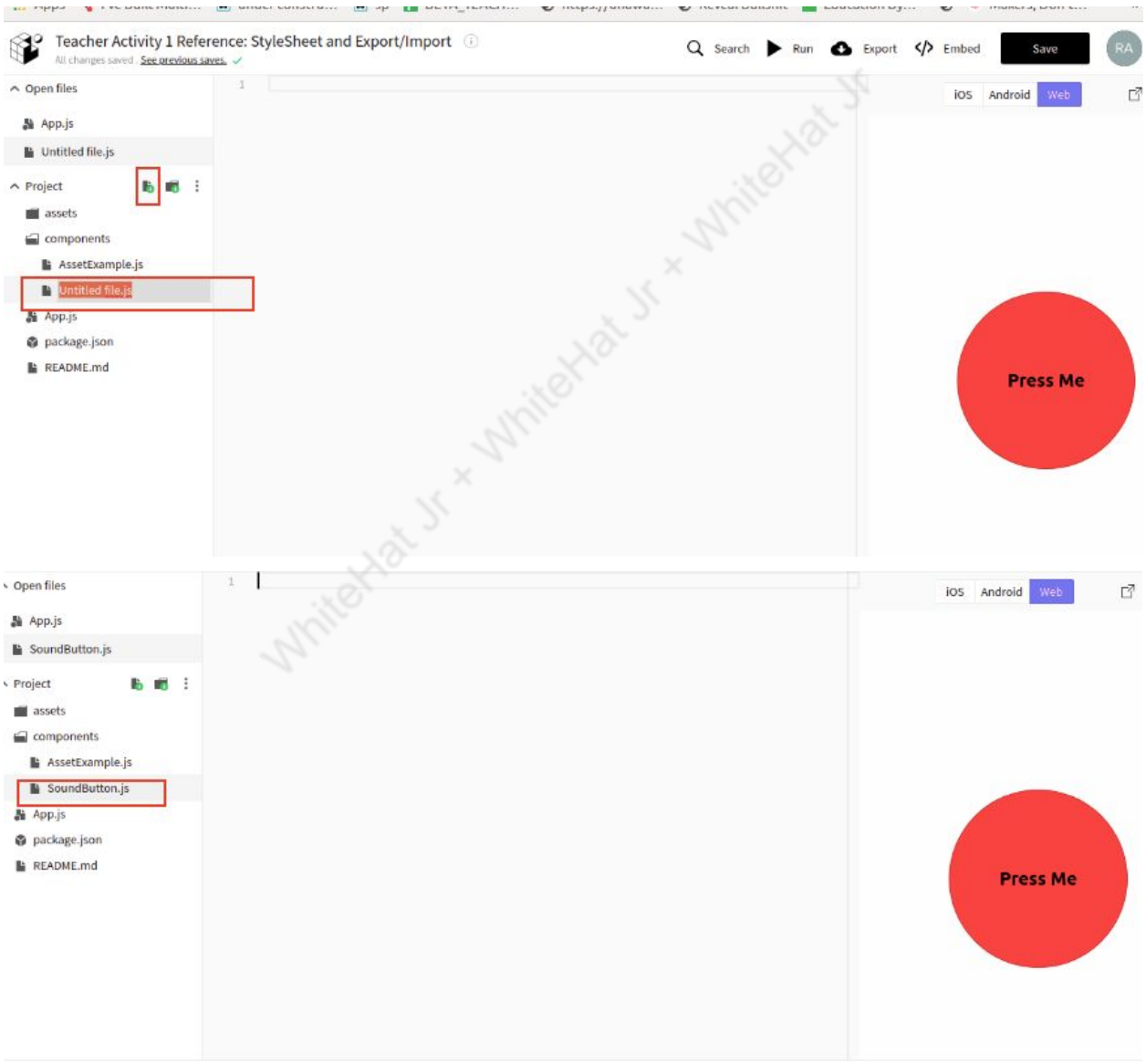
iOS Android Web

Press Me

Prettier { } Editor Expo v36.0.0 Devices 1 Preview

We created different components in different files to keep things more organized and then import all the components in App.js to use them.

- Create a new file inside the components folder and call it ‘SoundButton.js’ (since it defines the SoundButton Component).
- Copy all the code for the ‘SoundButton’ component into this file.
- Import components from the libraries used for this component including - react, expo-av , StyleSheet etc.
- Add an export statement - export default SoundButton.
- This instruction allows the SoundButton component to be imported by default when we write ‘import’ statement in another file.



WhiteHat Jr
Live Online Coding for Kids

Teacher Activity 1 Reference: StyleSheet and Export/Import

All changes saved 2 minutes ago. See previous saves.

Search Run

Open files
- App.js
- SoundButton.js

Project
- assets
- components
  - AssetExample.js
  - SoundButton.js
- App.js
- package.json
- README.md

```
import * as React from 'react';
import { Text, View, Button, TouchableOpacity, StyleSheet } from 'react-native';
import {Audio} from 'expo-av';

class SoundButton extends React.Component {
  playSound = async () => {
   await Audio.Sound.createAsync(
     { uri: 'http://soundbible.com/mp3/Buzzer-SoundBible.com-188422102.mp3' },
     { shouldPlay: true }
   );
  }

  render() {
   return (
     <TouchableOpacity
       style={styles.button}
       onPress={this.playSound}>
       <Text
         style={styles.buttonText}>
         Press Me
       </Text>
     </TouchableOpacity>
   );
  }
}

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

No errors

Prettier {} Editor

Teacher Activity 1 Reference: StyleSheet and Export/Import

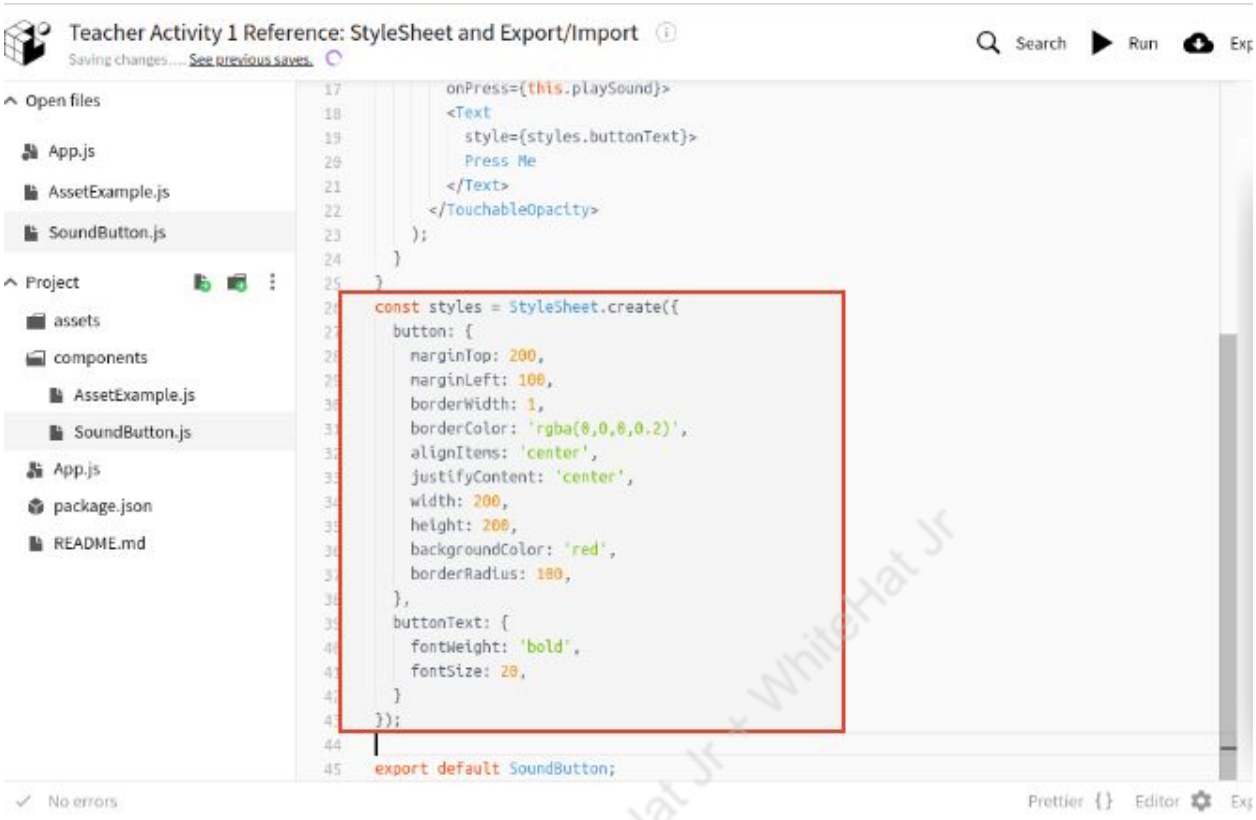
All changes saved See previous saves.

Search Run

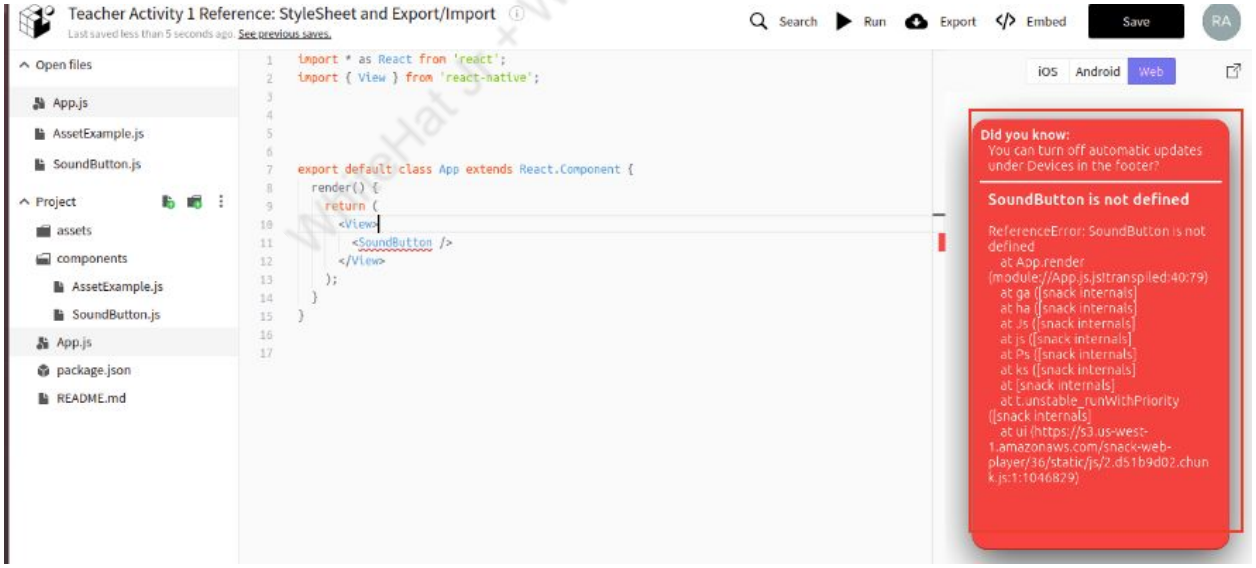
Open files
- App.js
- AssetExample.js
- SoundButton.js

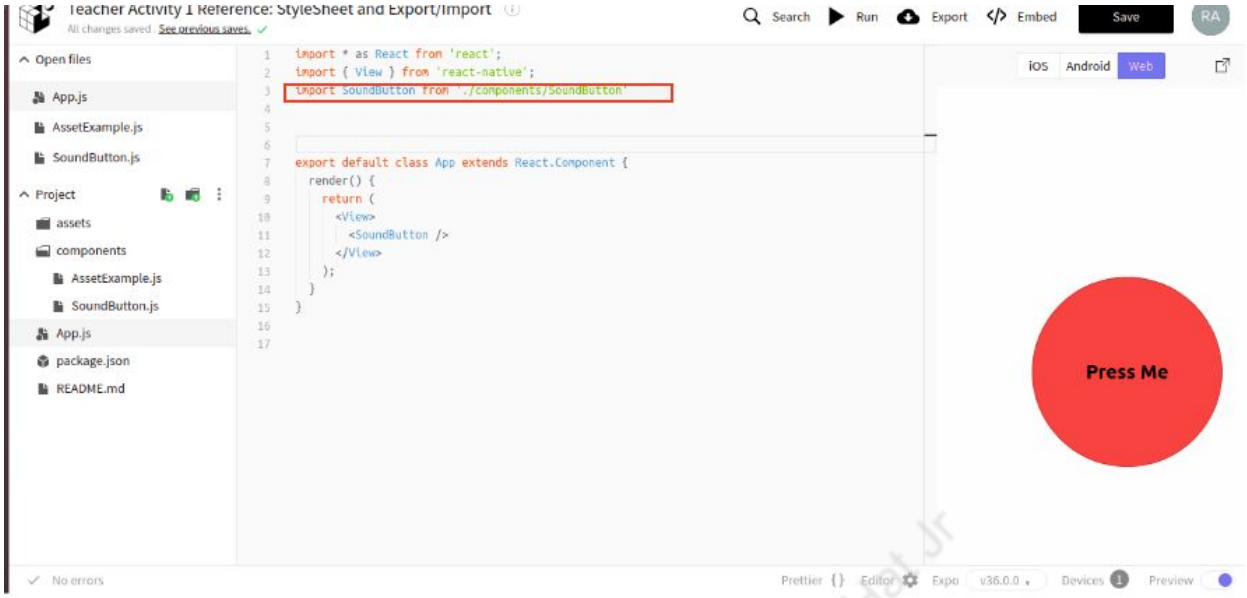
Project
- assets
- components
  - AssetExample.js
  - SoundButton.js
- App.js
- package.json
- README.md

```
import * as React from 'react';
import { Text, View, TouchableOpacity, StyleSheet } from 'react-native';
import {Audio} from 'expo-av';

class SoundButton extends React.Component {
  playSound = async () => {
   await Audio.Sound.createAsync(
     { uri: 'http://soundbible.com/mp3/Buzzer-SoundBible.com-188422102.mp3' },
     { shouldPlay: true }
   );
  }

  render() {
   return (
     <TouchableOpacity
       style={styles.button}
       onPress={this.playSound}>
       <Text
         style={styles.buttonText}>
         Press Me
       </Text>
     </TouchableOpacity>
   );
  }
}

export default SoundButton;
```



Import SoundButton from the components folder.

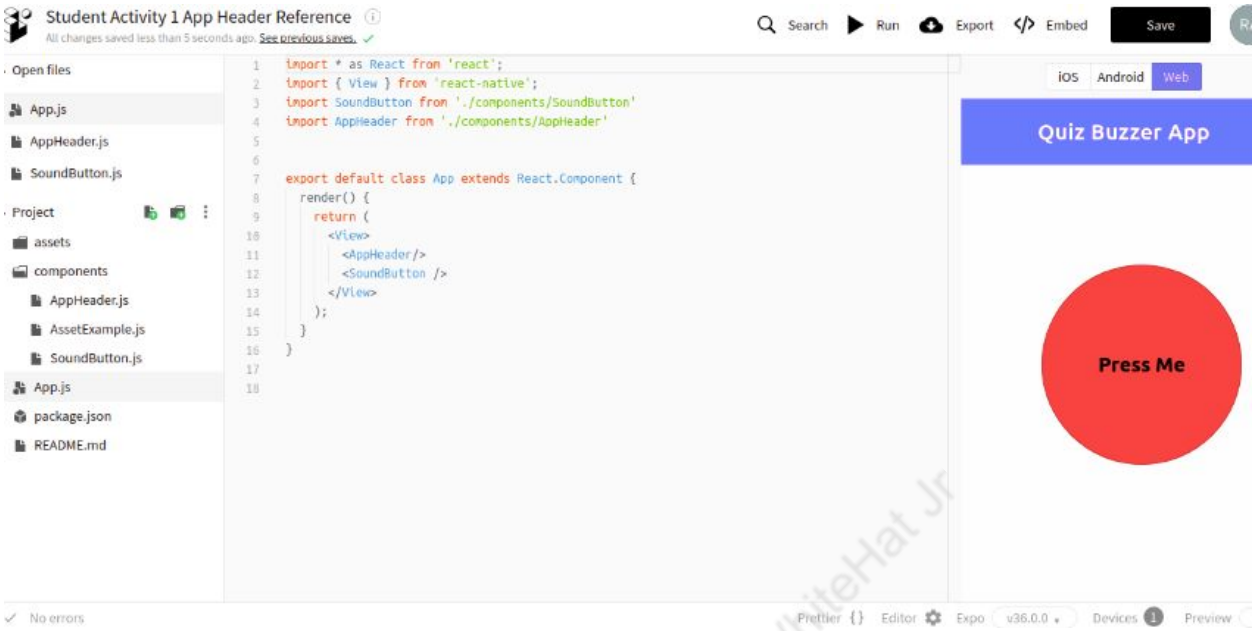




Create a component called AppHeader which gives a header to your app (with app name):

Write a component 'AppHeader.js'.
Import necessary components at the top.
Render the component inside the render() function.
Create appropriate styling for it using StyleSheet
Export the class as the default export.

Use the ‘AppHeader’ component you created inside ‘App.js’ file.



What’s next?:

In the next class, we will continue to work on the rest of the assigned tasks towards completing the Wireless Buzzer App. We will learn how to convert our single screen app to a two-screen app and how to navigate and pass information from one screen to another.