

Adding Phonic Sounds



What is our GOAL for this MODULE?

We reviewed the learnings so far in React Native - react-native philosophy, props, states, using pre-designed react components, designing custom react components, importing and exporting components from the react-native library etc.

What did we ACHIEVE in the class TODAY?

- Reviewed learnings in React Native.
- Designed 'phonicButton' component which takes the word chunk and plays the sound of phonic chunk corresponding to it.

Which CONCEPTS/ CODING BLOCKS did we cover today?

- Review React native - props, states, react native philosophy
- Design custom React Native Components

What did we REVISE today?

We revised the learnings so far in React Native.

1. What is react native philosophy?
 - React philosophy believes that everything in an application is a component. Each component has its own properties, states, functions etc.
2. What is the difference between the javascript code we wrote normally while designing games versus the javascript code we are writing now?
 - The javascript code we wrote normally was imperative - we gave detailed instructions to the computer on what to do and how to do it.

- The javascript code we write in React native is declarative - we just declare the components we need and what it will do. The how part of it is abstracted inside the component.
3. How is the declarative code better than imperative code?
- Declarative code helps in organizing and makes thinking about our program easier.
 - A more organized program allows us to write more complex code bases to do more complex tasks.
4. What are props and states in a component?
- Props are like attributes/properties of a component.
 - It can be accessed using `this.props.<propName>`.
 - Props can be used to pass data to a component.
 - States are the different states of a component. The component can change depending on the current state.
For example, a component button can have a state whether it is pressed or not. Depending on that, the component can appear or do different things.
5. What is Component Lifecycle?
- Every component has a defined lifecycle stages:
 - Initialization: When the component is assigned its initial values.
 - Mounting: When the component mounts on the screen.
 - Updating: When the props or state values of a component change.
 - Unmounting: When the component unmounts from the screen.
6. What are the methods associated with these stages in the lifecycle of React Components?
- Initialization: **'constructor()'** in the class of the component
 - Mounting: **'render()'**, **'componentWillMount()'**, **'componentDidMount()'**
 - Updating: **'shouldComponentUpdate()'**, **'componentWillUpdate()'**, **'componentDidUpdate()'**
 - Unmounting: **'componentDidUnmount()'**, **'componentWillUnMount'**
7. When are these methods called?
- These methods get automatically called at the different stages of the life cycle.
8. How to define your own component in React Native?
- We can create a new component class by extending the Component class from react-native library. We can then export this component.

- The component can be imported and used inside App class or any other component.
9. What are the pre-defined React native components we have used in our code so far?
- Text, View, StyleSheet etc.
10. What are the other libraries and components we have used in our code for Monkey-Chunky or Wireless Buzzer Apps we have created so far?
- firebase, Audio from expo-av.
11. We also used data in our app in two different ways. Which are they?
- We learned how to access data from a remote database server using firebase.
 - We also learned how to access data locally from a local json file.

How did we DO the activities?

So far, when the 'Go' button is pressed, we are only getting the word chunks. Now, we can also get the phonic sound chunks.

1. We will have to declare another state called 'phonicSounds' as an empty array. This will hold the phonic sounds associated with each word chunk.
 - If you look in line numbers 59 to 61, we are mapping over each word chunk and creating a 'TouchableOpacity' button for each chunk.

```

7   TouchableOpacity,
8   Image,
9 } from 'react-native';
10 import { Header } from 'react-native-elements';
11 import db from './localdb';
12
13 export default class App extends React.Component {
14   constructor() {
15     super();
16     this.state = {
17       text: '',
18       chunks: [],
19       phonicSounds: [],
20     };
21   }
22   render() {
23     return (
24       <View style={styles.container}>
25         <Header
26           backgroundColor={'#003366'}
27           centerComponent={() => {
28             text: 'Monkey Chunky',
29             style: { color: '00FF00', fontSize: 28 },
30           }}
31         />
32
33         <Image
34           style={styles.imageIcon}
35           source={{
36             uri:
37               'https://www.shareicon.net/data/128x128/2011/08/06/000001_Face_512x512.png',
38           }}
39         />

```

```

39       />
40
41       <TextInput
42         style={styles.inputBox}
43         onChangeText={text => {
44           this.setState({ text: text });
45         }}
46         value={this.state.text}
47       />
48
49       <TouchableOpacity
50         style={styles.goButton}
51         onPress={() => {
52           this.setState({ chunks: db[this.state.text].chunks });
53           this.setState({ phonicSounds: db[this.state.text].phones });
54         }}
55       >
56         <Text style={styles.buttonText}>GO</Text>
57       </TouchableOpacity>
58
59       <View>
60         {this.state.chunks.map(item => {
61           return (
62             <TouchableOpacity style={styles.chunkButton}>
63               <Text style={styles.displayText}>{item}</Text>
64             </TouchableOpacity>
65           );
66         })}
67       </View>
68     );
69   }
70 }
71
72 const styles = StyleSheet.create({

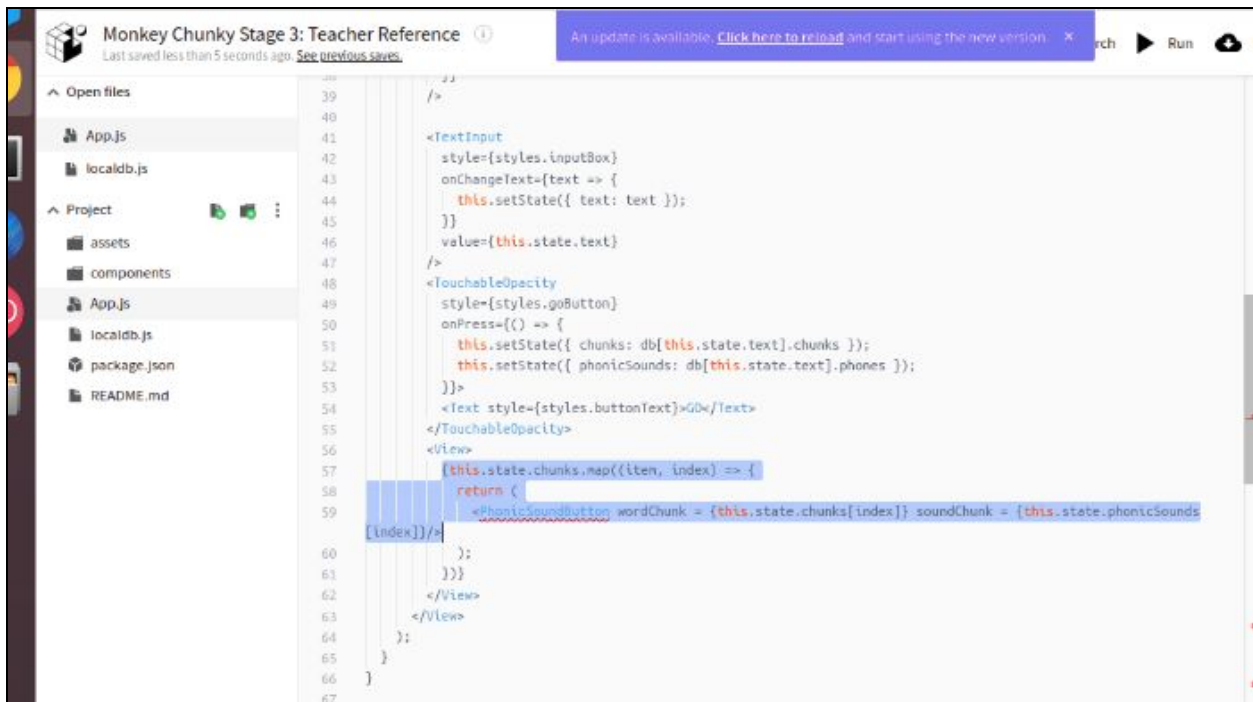
```

2. Now, we have to write code so that pressing the button plays the sound corresponding to the word chunk.
 - Since this component has a distinct functionality of its own, we can create a separate component of its own.
 - Let's call this component 'PhonicSoundButton'. It can take the word chunk and phone as props.
 - Actually before even creating the component, we can write code as if we have already created the component and it does what we expect it to do!
 - Later we can write code for the component and make it behave exactly as we want it to.

```

39      />
40
41      <TextInput
42        style={styles.inputBox}
43        onChangeText={text => {
44          this.setState({ text: text });
45        }}
46        value={this.state.text}
47      />
48      <TouchableOpacity
49        style={styles.goButton}
50        onPress={() => {
51          this.setState({ chunks: db[this.state.text].chunks });
52          this.setState({ phonicSounds: db[this.state.text].phones });
53        }}
54      >
55        <Text style={styles.buttonText}>GO</Text>
56      </TouchableOpacity>
57      <View>
58        {this.state.chunks.map(item => {
59          return (
60            <TouchableOpacity style={styles.chunkButton}>
61              <Text style={styles.displayText}>{item}</Text>
62            </TouchableOpacity>
63          );
64        })}
65      </View>
66    </View>
67  }
68 }
69
70 const styles = StyleSheet.create({

```



- You can see PhonicSoundButton is underlined with red because the computer does not know what it is.

3. Let's now define the 'PhonicSoundButton'.



- Let's write a render() method for this component.
- We want our word chunk to contain a text and act like a button. We can use a 'TouchableOpacity' Component with a Text inside it containing the text prop passed from the App Component.



```

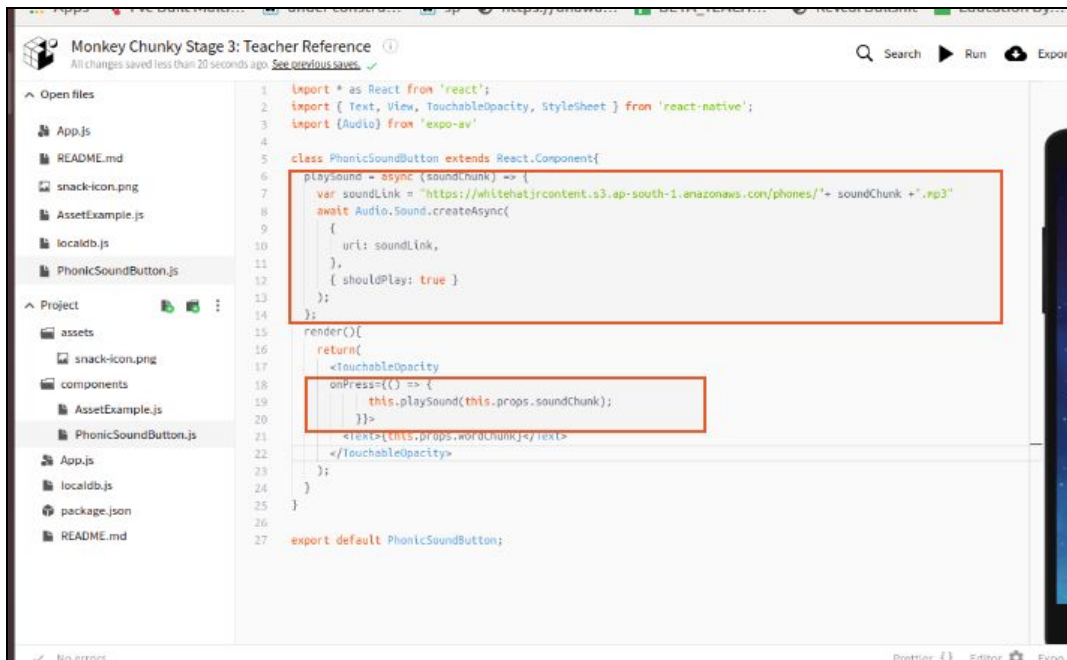
1  import * as React from 'react';
2  import { Text, View, TouchableOpacity, StyleSheet } from 'react-native';
3
4  class PhonicSoundButton extends React.Component{
5    render(){
6      return(
7        <TouchableOpacity>
8          <Text>{this.props.wordChunk}</Text>
9        </TouchableOpacity>
10      );
11    }
12  }
13
14  export default PhonicSoundButton;

```

6. Now, on pressing the 'TouchableOpacity', the sound corresponding to the phonic sounds should be played.
7. Look at all the sounds given in Student Activity 2. These are the same sounds which are stored in the phones array of the word in the database.

*Note: You can just change the file name at the end of the URL with the phone name to listen to the sound. When the button is pressed, you can concatenate the phone name with the URL to create the sound URI.

8. Let's play the sound of the phone when the button is pressed.
9. Remember, you will have to import Audio from 'expo-av' library.



```

1  import * as React from 'react';
2  import { Text, View, TouchableOpacity, StyleSheet } from 'react-native';
3  import { Audio } from 'expo-av';
4
5  class PhonicSoundButton extends React.Component{
6    playsound = async (soundChunk) => {
7      var soundLink = "https://whitehatjrcontent.s3.ap-south-1.amazonaws.com/phones/" + soundChunk + ".mp3";
8      await Audio.Sound.createAsync(
9        {
10          uri: soundLink,
11        },
12        { shouldPlay: true }
13      );
14    };
15
16    render(){
17      return(
18        <TouchableOpacity
19          onPress={() => {
20            this.playSound(this.props.soundChunk);
21          }}
22        <Text>{this.props.wordChunk}</Text>
23        </TouchableOpacity>
24      );
25    }
26  }
27  export default PhonicSoundButton;

```

10. Add proper styling to our 'TouchableOpacity' and Text we have used here.

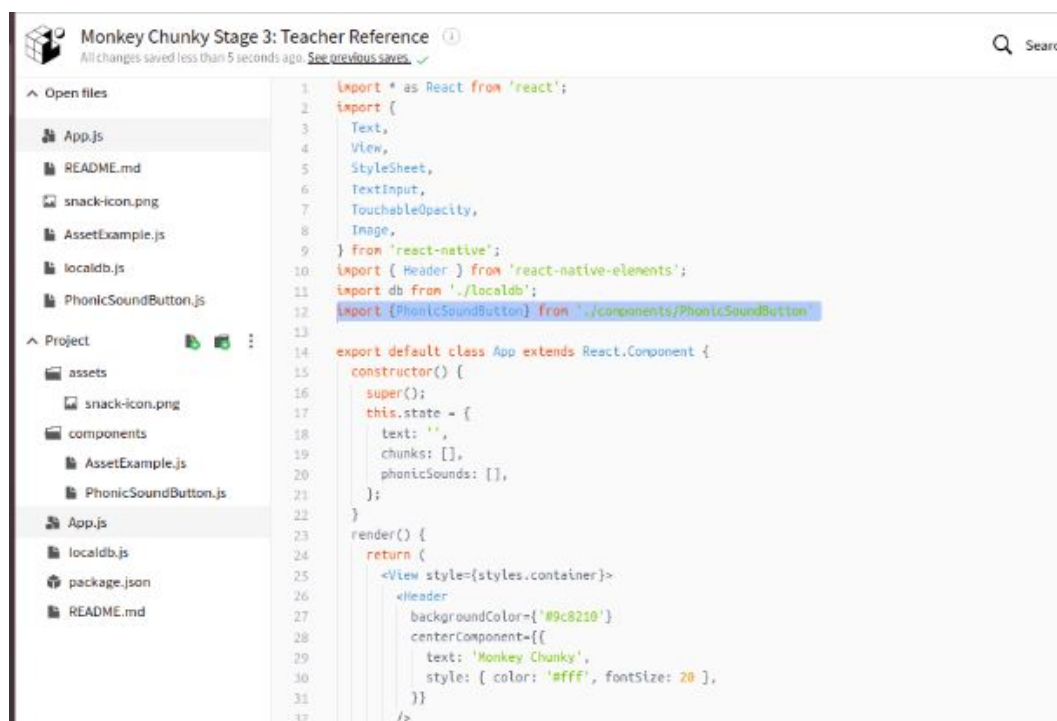


```

17   };
18   };
19   render() {
20     return (
21       <TouchableOpacity
22         style={styles.chunkButton}
23         onPress={() => {
24           this.playSound(this.props.soundChunk);
25         }}
26       <Text style={styles.displayText}>{this.props.wordChunk}</Text>
27     </TouchableOpacity>
28   );
29 }
30 }
31
32 const styles = StyleSheet.create({
33   displayText: {
34     textAlign: 'center',
35     fontSize: 30,
36     color: 'white'
37   },
38   chunkButton: {
39     width: '60%',
40     height: 50,
41     justifyContent: 'center',
42     alignItems: 'center',
43     alignSelf: 'center',
44     borderRadius: 10,
45     margin: 5,
46     backgroundColor: 'red'
47   }

```

We have the 'phonicSound' component ready now. We can import this in your App.js file.



```

1  import * as React from 'react';
2  import {
3    Text,
4    View,
5    StyleSheet,
6    TextInput,
7    TouchableOpacity,
8    Image,
9  } from 'react-native';
10 import { Header } from 'react-native-elements';
11 import db from './localdb';
12 import { PhonicSoundButton } from './components/PhonicSoundButton';
13
14 export default class App extends React.Component {
15   constructor() {
16     super();
17     this.state = {
18       text: '',
19       chunks: [],
20       phonicSounds: [],
21     };
22   }
23   render() {
24     return (
25       <View style={styles.container}>
26         <Header
27           backgroundColor={'#9c8210'}
28           centerComponent={[
29             text: 'Monkey Chunky',
30             style: { color: 'white', fontSize: 20 },
31           ]}
32         />

```


What's NEXT?

In the next class, we will learn how we can add more words and fix some small issues in our app.

EXTEND YOUR KNOWLEDGE

1. Phonic Sounds: <http://www.antimoon.com/how/pronunc-soundsipa.htm>