

Topic	Adding Phonic sounds	
Class Description	Student reviews 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 react native library etc. Students also design a phonicButton component which takes the word chunk and plays the sound of a phonic chunk corresponding to it.	
Class	C65	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> Review learnings in React Native. Design 'phonicButton' component which takes the word chunk and plays the sound of phonic chunk corresponding to it. 	
Resources Required	<ul style="list-style-type: none"> Teacher Resources <ul style="list-style-type: none"> Laptop with internet connectivity Earphones with mic Notebook and pen Android/iOS Smartphone with Expo App installed Student Resources <ul style="list-style-type: none"> Laptop with internet connectivity Earphones with mic Notebook and pen Android/iOS Smartphone with Expo App installed 	
Class structure	Warm Up Teacher-led Activity Student-led Activity Wrap up	5 mins 15 min 15 min 5 min
WARM-UP SESSION - 5 mins		
<u>CONTEXT</u> <ul style="list-style-type: none"> Review the Monkey-Chunky Code Base so far. 		



Teacher starts slideshow from slides 1 to 14

Refer to speaker notes and follow the instructions on each slide.

Activity details	Solution/Guidelines
<p><i>Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?</i></p> <p>Run the presentation from slide 1 to slide 4</p> <p>Following are the WARM-UP session deliverables:</p> <ul style="list-style-type: none"> • Greet the student. • Revision of previous class activities. • Quizzes 	<p>ESR: Hi, thanks, Yes I am excited about it!</p> <p>Click on the slide show tab and present the slides</p>
QnA Session	
Question	Answer
<p>Identify the option to display the definition of the word when the search button is pressed.</p> <p>A. <pre><Text style={{ fontSize:18}}> {this.state.definition} </Text></pre></p> <p>B. <pre><Text style={{ fontSize:18}}> {this.state.word} </Text></pre></p> <p>C. <pre><Text style={{ fontSize:18}}> {this.state.lexicalCategory} </Text></pre></p> <p>D. <pre><Text style={{ fontSize:18}}> definition </Text></pre></p>	<p>A</p>

Which of the following options will display the definition of the word after pressing the search button? The definition is stored in the variable “definition”.



- A. `var definition = dictionary[text]`
- B. `var definition = ["definition"]`
- C. `var definition = dictionary["definition"]`
- D. `var definition = dictionary[text]["definition"]`

D

Continue the WARM-UP session

Activity details

Run the presentation from slide 5 to slide 14 to set the problem statement.

Following are the WARM-UP session deliverables:

- Appreciate the student.
- Recall the learnings of React Native.

Solution/Guidelines

Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.

Teacher ends slideshow



TEACHER-LED ACTIVITY - 15 mins

Teacher Initiates Screen Share

CHALLENGE



- Recall the learnings in React Native so far:
 - React philosophy
 - States and Props
 - Using React components
 - Designing React components
 - Importing/Exporting components from libraries
 - Connecting to Remote/local database

Step 2: Teacher-led Activity (15 min)	Alright. So can you explain in your own words what reactive native philosophy?	React philosophy believes that everything in an application is a component. Each component has its own properties, states, functions etc.
	What is the difference between the javascript code we wrote normally while designing games versus the javascript code we are writing now.	<p>The javascript code we wrote normally was imperative - we gave detailed instructions to the computer on what to do and how to do it.</p> <p>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.</p>
	How is declarative code better than imperative code?	Declarative code helps in organizing and makes thinking about our program easier. More organized program allows us to write more complex code bases to do more complex tasks.

	<p>Awesome! You already know why React philosophy is so powerful and how react re-imagines all programs as made up of components.</p> <p>Let's talk more about props and states of a component. What are props and states in a component?</p>	<p>Props are like attributes/properties of a component. It can be accessed using <code>this.props.<propName></code>. Props can be used to pass data to a component.</p> <p>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.</p>
	<p>You are doing very well <code><student name></code></p> <p>Can you also recall the Component Lifecycle?</p>	<p>Every component has a defined lifecycle stages:</p> <ul style="list-style-type: none"> - 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.
	<p>Brilliant!</p>	<p>ESR:</p> <p>Initialization: <code>'constructor()'</code> in the class of the component</p>

	Do you also remember any of the methods associated with these stages in the lifecycle of React Components:	<p>Mounting: 'render()', 'componentWillMount()', 'componentDidMount()'</p> <p>Updating: 'shouldComponentUpdate()', 'componentWillUpdate()', 'componentDidUpdate()'</p> <p>Unmounting: 'componentDidUnmount()', 'componentWillUnMount'</p>
	When are these methods called?	These methods get automatically called at the different stages of the lifecycle.
	Do you remember how to define your own component in React Native?	<p>ESR:</p> <p>Yes. We can create a new component class by extending the Component class from react-native library.</p> <p>We can then export this component.</p> <p>The component can be imported and used inside the App class or any other component.</p>
	<p>What are the pre-defined React native components we have used in our code so far?</p> <p>Note: Let the student explain each of the components and what it does.</p>	<p>ESR:</p> <p>Text, View, StyleSheet etc.</p>

	<p>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?</p> <p>Note: The teacher can ask the student to revisit the links for these projects and recall how these components were used.</p>	<p>ESR: firebase, Audio from expo-av.</p>
	<p>We also used data in our app in two different ways. Can you recall them?</p>	<p>ESR: 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.</p>
	<p>You remember a great deal!</p> <p>You are in a perfect place to finish off the final crucial part of the Monkey-Chunky app where we add sounds to our word chunks.</p> <p>Let's get started.</p>	
Teacher Stops Screen Share		
STUDENT-LED ACTIVITY - 25 mins		
<ul style="list-style-type: none"> • Ask Student to press ESC key to come back to panel • Guide Student to start Screen Share • Teacher gets into Fullscreen 		
<p align="center"><u>ACTIVITY</u></p> <ul style="list-style-type: none"> • Design a phonic button which takes the word chunk and plays the sound of phonic chunk corresponding to it. 		

<div>  <p>Teacher starts slideshow from slides 15 to 18</p> <p>Refer to speaker notes and follow the instructions on each slide.</p> </div>		
	Now it's your turn. Please share your screen with me.	
<div>  <p>Teacher ends slideshow</p> </div>		
Step 3: Student-Led Activity (15 min)	<p>Let's quickly open the last project and review the code base.</p> <p>Can you loudly articulate what we are doing in our codebase?</p>	<p>Student opens <u>Student Activity 1</u> and articulates what the code is doing so far.</p>
	<p>So far, when the 'Go' button is pressed, we are only getting the word chunks.</p> <p>Now, we can also get the phonic sound chunks.</p> <p>We will have to declare another state called 'phonicSounds' as an empty array. This will hold the phonic sounds associated with each word chunk.</p>	<p>The student creates a new state called 'phonicSounds' and gets the phonic sound names associated with each word chunk inside it.</p>


```
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: 'fff', fontSize: 20 },  
31            }}  
32          />  
33  
34          <Image  
35            style={styles.imageIcon}  
36            source={{  
37              uri:  
38                'https://www.shareicon.net/data/128x128/2015/08/06/80805_face_512x512.png',  
39            }}  
40          />  
41        )  
42      )
```

```

40      />
41
42      <TextInput
43        style={styles.inputBox}
44        onChangeText={text => {
45          this.setState({ text: text });
46        }}
47        value={this.state.text}
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        <Text style={styles.buttonText}>GO</Text>
56      </TouchableOpacity>
57      <View>
58        {this.state.chunks.map((item, index) => {
59          return (
60            <PhonicSoundButton
61              wordChunk={this.state.chunks[index]}
62              soundChunk={this.state.phonicSounds[index]}
63            />
64          );
65        })}
66      </View>
67    </View>
68  );
69  }
70  }
71
72  const styles = StyleSheet.create({
73    container: {
74      flex: 1,
75      backgroundColor: '#b8b8b8',
76    },

```

If you look in line numbers 59 to 61, we are mapping over each word chunk and creating a 'TouchableOpacity' button for each chunk.

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.

Student writes the related code.

Let's call this component 'PhonicSoundButton'. It can take the word chunk and phone as props.

```

40     />
41
42     <TextInput
43       style={styles.inputBox}
44       onChangeText={text => {
45         this.setState({ text: text });
46       }}
47       value={this.state.text}
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     <Text style={styles.buttonText}>GO</Text>
56   </TouchableOpacity>
57   <View>
58     {this.state.chunks.map((item, index) => {
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
71 const styles = StyleSheet.create({
72   container: {
73     flex: 1

```

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.

The student writes the code.

Teacher explains that .map() passes both item and index for each item in the array to the callback function.

```

41
42 <TextInput
43   style={styles.inputBox}
44   onChangeText={text => {
45     this.setState({ text: text });
46   }}
47   value={this.state.text}
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   <Text style={styles.buttonText}>GO</Text>
56 </TouchableOpacity>
57 <View>
58   {this.state.chunks.map((item, index) => {
59     return (
60       <PhonicSoundButton
61         wordChunk={this.state.chunks[index]}
62         soundChunk={this.state.phonicSounds[index]}
63       />
64     );
65   })}
66 </View>
67 </View>
68 );
69 }
70 }
71

```

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

Let's now define the 'PhonicSoundButton'.

Can you quickly do that?

Student creates a new file called 'PhonicSoundButton.js' and starts with a template code for creating a new Component class.

```

1  import * as React from 'react';
2  import { Text, View, TouchableOpacity, StyleSheet } from 'react-native';
3  import { Audio } from 'expo-av';
4
5  export default class PhonicSoundButton extends React.Component {
6    |
7  }
8

```

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.

The student writes the code.


```

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

export default class PhonicSoundButton extends React.Component {
  render() {
    return (
      <TouchableOpacity>
        <Text>{this.props.wordChunk}</Text>
      </TouchableOpacity>
    );
  }
}
|

```

	<p>Now, on pressing the 'TouchableOpacity', the sound corresponding to the phonic sounds should be played.</p> <p>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.</p> <p>Note: You can just change the file name at the end of the URL with the phone name to listen to the sound.</p>	<p>The student looks at the sounds uploaded on Amazon.</p>
	<p>When the button is pressed, you can concatenate the phone name with the URL to create the sound URI.</p> <p>Let's play the sound of the phone when the button is pressed.</p> <p>Remember, you will have to import Audio from 'expo-av' library.</p>	<p>The student writes code to play the sound of the word chunk when the button is pressed.</p> <p>Student can refer to the Quiz Buzzer App written earlier.</p>


Monkey Chunky Stage 3: Teacher Reference ⓘ
 All changes saved 2 minutes ago. [See previous saves.](#) ✓

Open files
 App.js
 PhonicSoundButton.js
 Project
 assets
 components
 AssetExample.js
 PhonicSoundButton.js
 App.js
 localdb.js
 package.json
 README.md

```

1  import * as React from 'react';
2  import { Text, View, TouchableOpacity, StyleSheet } from 'react-native';
3  import { Audio } from 'expo-av';
4
5  export default class PhonicSoundButton extends React.Component {
6    playSound = async (soundChunk) => {
7      console.log(soundChunk);
8      var soundLink =
9        'https://s3-whitehatjrcontent.whjr.online/phones/' + soundChunk + '.mp3';
10     await Audio.Sound.createAsync(
11       {
12         uri: soundLink,
13       },
14       { shouldPlay: true }
15     );
16   };
17   render() {
18     return (
19       <TouchableOpacity
20         style={styles.chunkButton}
21         onPress={() => {
22           this.playSound(this.props.soundChunk);
23         }}
24       >
25       <text style={styles.displayText}>{this.props.wordChunk}</text>
26     </TouchableOpacity>
27   );
28   }
29
30   const styles = StyleSheet.create({
31     displayText: {
32       textAlign: 'center',
33       fontSize: 30,
34       color: 'white',

```

	Let's add proper styling to our 'TouchableOpacity' and Text we have used here.	The student adds styling using StyleSheet.
--	--	--

```

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   }
48 });

```

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

The student imports 'PhonicSoundButton' in App.js.


```

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           }}

```

Now you can run and test your app to see if it works.

The student runs and tests the app on their phone using the expo app.

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 5 Mins

Teacher starts slideshow




from slide 19 to slide 28

Activity details

Solution/Guidelines

<p>Run the presentation from slide 19 to slide 28</p> <p>Following are the wrap-up session deliverables:</p> <ul style="list-style-type: none"> ● Explain the facts and trivias ● Next class challenge ● Project for the day ● Additional Activity 	<p>Guide the student to develop the project and share with us.</p>
<p>Quiz time - Click on in-class quiz</p>	
Question	Answer
<p>Which function from the expo-av library is used to play the sound?</p> <p>A. await Audio.Sound.createAsync() B. await Audio.Sound.play() C. await Audio.play() D. sound.createAsync()</p>	<p>A</p>
<p>In which stage of the Lifecycle do props or state values of a component change?</p> <p>A. mounting B. initialization C. updating D. unmounting</p>	<p>C</p>
<p>Using which component can you style each text as a clickable button?</p> <p>A. TouchableOpacity component B. Touchable component C. Opacity component D. Button</p>	<p>A</p>

End the quiz panel		
FEEDBACK <ul style="list-style-type: none"> Encourage the student to review React Native using the reference material provided in the links. 		
	<p>You get a “hats off”.</p> <p>Till next class then. See you. Bye!</p>	<p>Make sure you have given at least 2 Hats Off during the class for:</p> <div>Creatively Solved Activities +10</div> <div>Great Question +10</div> <div>Strong Concentration +10</div>
	<p>In the next class, we will learn how we can add more words and fix these small issues in our app.</p> <p>Also, we will be learning more about git and how we can contribute to Open Source Applications using git.</p> <p>It is going to be an exciting session!</p>	
Project Pointers and Cues (5 min)	<p>* This Project will take only 30 mins to complete. Motivate students to try and finish it immediately after the class.</p> <p>TEXT TO SPEECH CONVERTER</p> <p>Goal of the Project:</p>	<p>Note: You can assign the project to the student in class itself by clicking on the Assign Project button which is available under the projects tab.</p>

	<p>Today you learned to play the corresponding sound of a phonic chunk.</p> <p>In this text to speech converter app you have to use expo speech library to convert a text entered by the user to a pronounced word.</p> <p>Story:</p> <p>Saisha's friend Lisa is visiting her from France. She understands English, but speaks only French.</p> <p>Help Saisha create a Text-to-Speech app for Lisa so she can communicate with people in India.</p> <p>I am very excited to see your project solution and I know you both will do really well.</p> <p>Bye Bye!</p>	
<div> <div>Teacher Clicks</div> <div>✕ End Class</div> </div>		
<div> <div>Teacher ends slideshow</div>  </div>		
Additional Activities	<p>Encourage the student to write reflection notes in their reflection journal using markdown.</p> <p>Use these as guiding questions:</p>	<p>The student uses the markdown editor to write her/his reflection in a reflection journal.</p>

	<ul style="list-style-type: none"> • What happened today? <ul style="list-style-type: none"> - Describe what happened - Code I wrote • How did I feel after the class? • What have I learned about programming and developing games? • What aspects of the class helped me? What did I find difficult? 	
--	---	--

Activity	Activity Name	Links
Teacher Activity 1	Teacher Reference	https://snack.expo.io/@rajeevtfi/monkey-chunky-stage-3:-teacher-reference
Student Activity 1	Class Activity	https://snack.expo.io/@rajeevtfi/monkeychunky-stage2--reference
Student Activity 2	Phone sounds link	https://s3-whitehatjrcontent.whjr.online/phones/V.mp3
Student Activity 3	Student Reference	https://facebook.github.io/react-native/docs/tutorial
Teacher Reference visual aid link	Visual aid link	https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/BJFC_PRO_V3_C65_withcues.html
Teacher Reference In-class quiz	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/bb5084c2-5a22-4fcd-94c1-890f4dede8b6.pdf