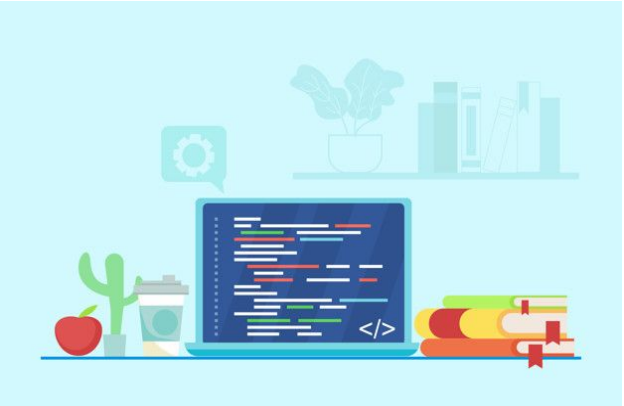
Accessing data from local files



What is our GOAL for this MODULE?

We learned to access data from local files and used the data from a local file to break a given word into smaller chunks associated with a phoneme sound.

What did we ACHIEVE in the class TODAY?

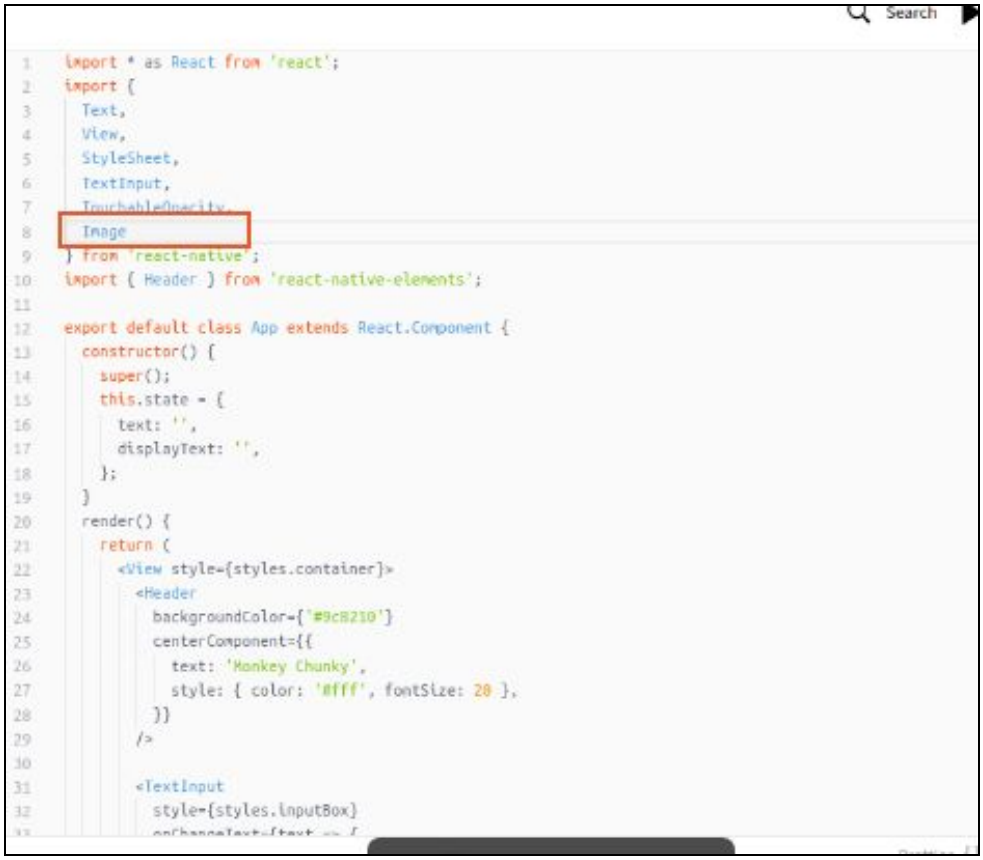
- Exported and imported data from a local file.
- Got smaller chunks of a word and displayed it using the ‘map’ method.

Which CONCEPTS/ CODING BLOCKS did we cover today?

- Access local data
- Array.map()

How did we DO the activities?

1. Let's first add some image to our Monkey-chunky app to give it some branding, using 'Image' Component.



```
export default class App extends React.Component {
  constructor() {
    super();
    this.state = {
      text: '',
      displayText: '',
    };
  }
  render() {
    return (
      <View style={styles.container}>
        <Header
          backgroundColor={'#9c8210'}
          centerComponent={{
            text: 'Monkey Chunky',
            style: { color: '#fff', fontSize: 20 },
          }}
        />

        <Image
            style={styles.imageIcon}
            source={{
              uri:
                'https://www.shareicon.net/data/128x128/2015/08/06/80005_face_512x512.png',
            }}
          />

        <TextInput
          style={styles.inputBox}
          onChangeText={text => {
            this.setState({ text: text });
          }}
```

```
  },
  inputBox: {
    marginTop: 50,
    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,
  },
  imageIcon:{
    width: 150,
    height: 150,
    marginLeft: 75,
  }
});
```

iOS Android Web

Monkey Chunky

GO

Prettier Editor Expo v36.0.0 Devices Preview

2. Right now our app has an input box where we type text and it displays the same word below. Instead of the same word, we need to chunk the words.

3. To do that we need Firebase Realtime Database where the chunks of each word are stored.

- That's an online database where we stored data in JSON format. To use Firebase Database, our users have to stay connected to the internet.
- There is another way we can store and use data - in a local file. We can store 'json' object in a local file and use it to access the data we need.

4. We have JSON data here which contains chunks of a few words in an array. It also contains the associated phonemes which we will use later.

5. The "chunks" and "phones" of each word is stored inside the word keyname.
For example: For the word "the", the chunks are stored in the array ["th", "e"]

6. Create a file called 'localdb', where we are going to store these words.

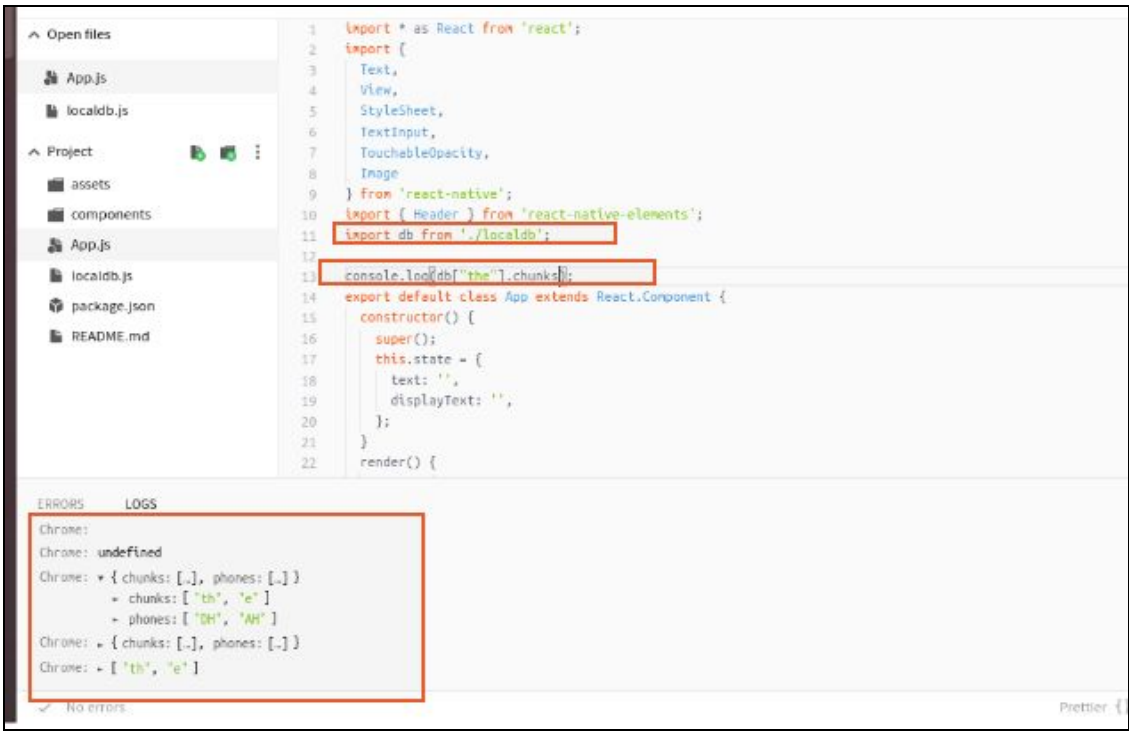
7. We can create a variable called db which will hold this JSON object.
   - Since this is not going to change in the program, we will make it a constant using "const" keyword.


```
const db = {
  the: { chunks: ['th', 'e'], phones: ['DH', 'AH'] },
  of: { chunks: ['o', 'f'], phones: ['AH', 'V'] },
  and: { chunks: ['a', 'n', 'd'], phones: ['AH', 'N', 'D'] },
  to: { chunks: ['t', 'o'], phones: ['T', 'UW'] },
  a: { chunks: ['a'], phones: ['AH'] },
  in: { chunks: ['i', 'n'], phones: ['IH', 'N'] },
  for: { chunks: ['f', 'o', 'r'], phones: ['F', 'AO', 'R'] },
  is: { chunks: ['i', 's'], phones: ['IH', 'Z'] },
  on: { chunks: ['o', 'n'], phones: ['AA', 'N'] },
  that: { chunks: ['th', 'a', 't'], phones: ['DH', 'AE', 'T'] },
  by: { chunks: ['b', 'y'], phones: ['B', 'AY'] },
  this: { chunks: ['th', 'i', 's'], phones: ['DH', 'IH', 'S'] },
  with: { chunks: ['w', 'i', 'th'], phones: ['W', 'IH', 'DH'] },
  i: { chunks: ['i'], phones: ['AY'] },
  you: { chunks: ['y', 'ou'], phones: ['Y', 'UW'] },
  it: { chunks: ['i', 't'], phones: ['IH', 'T'] },
  not: { chunks: ['n', 'o', 't'], phones: ['N', 'AA', 'T'] },
  or: { chunks: ['o', 'r'], phones: ['AO', 'R'] },
  be: { chunks: ['b', 'e'], phones: ['B', 'IY'] },
  are: { chunks: ['a', 're'], phones: ['AA', 'R'] },
  from: { chunks: ['f', 'r', 'o', 'm'], phones: ['F', 'R', 'AH', 'M'] },
  at: { chunks: ['a', 't'], phones: ['AE', 'T'] },
  as: { chunks: ['a', 's'], phones: ['AE', 'Z'] },
  your: { chunks: ['y', 'ou', 'r'], phones: ['Y', 'AO', 'R'] },
  all: { chunks: ['a', 'll'], phones: ['AO', 'L'] },
  have: { chunks: ['h', 'a', 've'], phones: ['HH', 'AE', 'V'] },
  new: { chunks: ['n', 'ew'], phones: ['N', 'UW'] },
  more: { chunks: ['m', 'o', 're'], phones: ['M', 'AO', 'R'] },
  an: { chunks: ['a', 'n'], phones: ['AE', 'N'] },
  was: { chunks: ['w', 'a', 's'], phones: ['W', 'AA', 'Z'] },
  we: { chunks: ['w', 'e'], phones: ['W', 'IY'] },
};
```

8. Now, we need to export this variable db so that we can use it in our app wherever we need it.

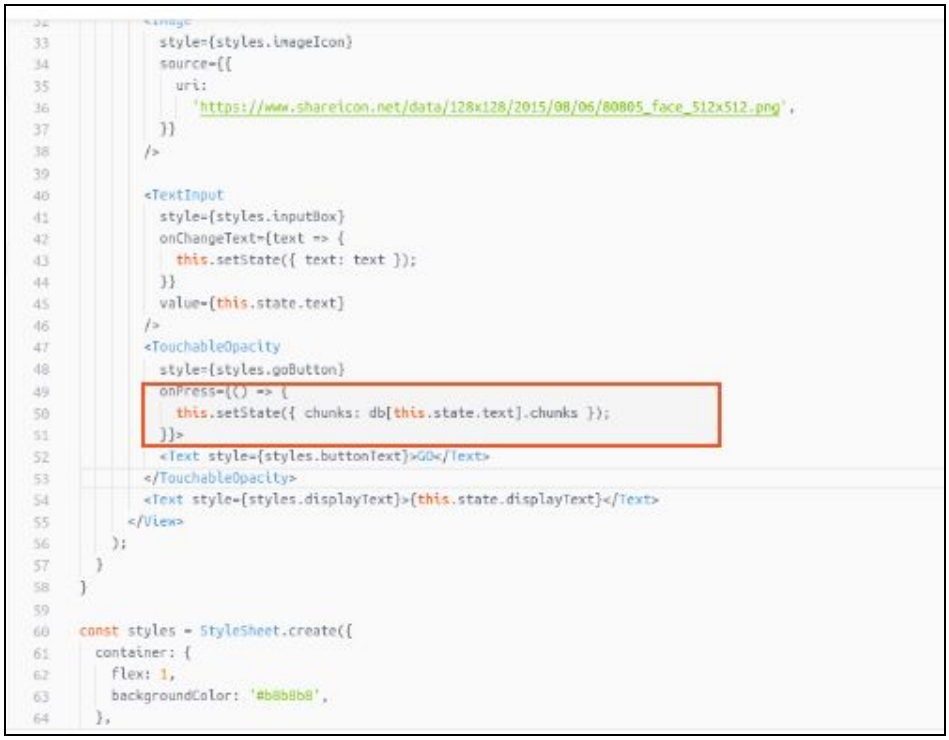
```
  or: { chunks: ['o', 'r'], phones: ['AO', 'R'] },
  be: { chunks: ['b', 'e'], phones: ['B', 'IY'] },
  are: { chunks: ['a', 're'], phones: ['AA', 'R'] },
  from: { chunks: ['f', 'r', 'o', 'm'], phones: ['F', 'R', 'AH', 'M'] },
  at: { chunks: ['a', 't'], phones: ['AE', 'T'] },
  as: { chunks: ['a', 's'], phones: ['AE', 'Z'] },
  your: { chunks: ['y', 'ou', 'r'], phones: ['Y', 'AO', 'R'] },
  all: { chunks: ['a', 'll'], phones: ['AO', 'L'] },
  have: { chunks: ['h', 'a', 've'], phones: ['HH', 'AE', 'V'] },
  new: { chunks: ['n', 'ew'], phones: ['N', 'UW'] },
  more: { chunks: ['m', 'o', 're'], phones: ['M', 'AO', 'R'] },
  an: { chunks: ['a', 'n'], phones: ['AE', 'N'] },
  was: { chunks: ['w', 'a', 's'], phones: ['W', 'AA', 'Z'] },
  we: { chunks: ['w', 'e'], phones: ['W', 'IY'] },
};
export default db;
```

9. Now we can simply import the variable wherever we need it and use it in our app.



10. Instead of 'displayText', let's create a state called 'chunks'.
    - 'chunks' will be an array that will hold the parts of the word typed in the input box. For now it can be an empty array.

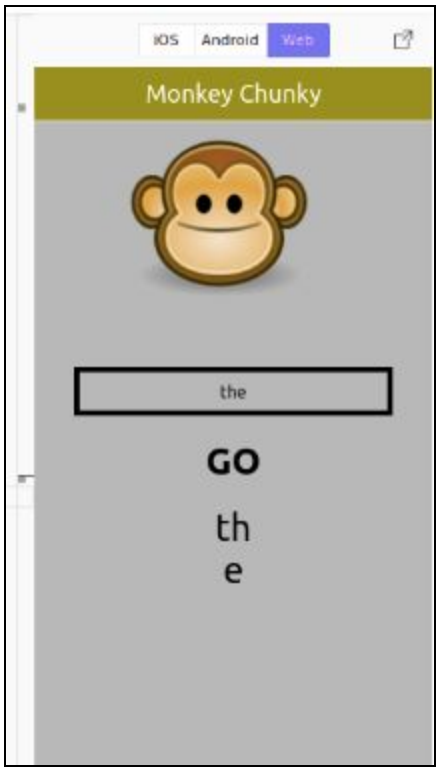
11. When "Go" Button is pressed, update the chunks.



```
        style={styles.imageIcon}
        source={{
          uri:
            'https://www.shareicon.net/data/128x128/2015/08/06/80805_face_512x512.png',
        }}
      />

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

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

12. In render() function, inside a View Component iterate over all the elements inside the 'chunks' state and render a text for each chunk.

```
        onChangeText={text => {
          this.setState({ text: text });
        }}
        value={this.state.text}
      />
      <TouchableOpacity
        style={styles.goButton}
        onPress={() => {
          this.setState({ chunks: db[this.state.text].chunks });
        }}>
        <Text style={styles.buttonText}>GO</Text>
      </TouchableOpacity>
      <View>
        {this.state.chunks.map(item => {
          return <Text style={styles.displayText}>{item}</Text>;
        })}
      </View>
    </View>
  );
 }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#b8b8b8',
  },
  inputBox: {
    marginTop: 50,
    width: '80%',
    alignSelf: 'center',
    height: 40,
```

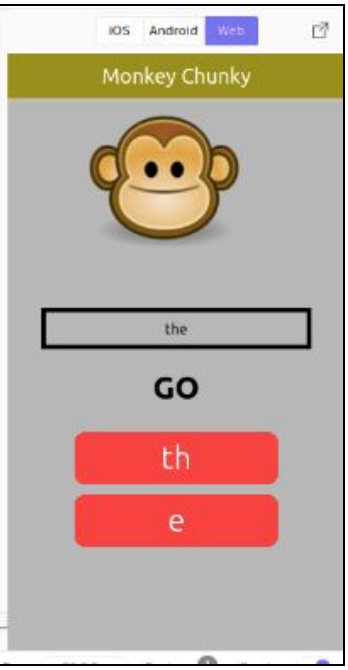


- Now in our App, the user will be able to press on each chunk to listen to the sound of the associated phoneme.

- To allow the user to press the chunk, each chunk should look like a button.

```
<Image
  style={styles.imageIcon}
  source={{
    uri:
      'https://www.shareicon.net/data/128x128/2015/08/06/80805_face_512x512.png',
  }}
/>

<TextInput
  style={styles.inputBox}
  onChangeText={text => {
    this.setState({ text: text });
  }}
  value={this.state.text}
/>
<TouchableOpacity
  style={styles.goButton}
  onPress={() => {
    this.setState({ chunks: db[this.state.text].chunks });
  }}>
  <Text style={styles.buttonText}>GO</Text>
</TouchableOpacity>
<View>
  {this.state.chunks.map(item => {
    return (
      <TouchableOpacity
      style={styles.chunkButton}
      >
      <Text style={styles.displayText}>{item}</Text>
      </TouchableOpacity>
      );
  })}
```

```
goButton: {
  width: '50%',
  height: 55,
  alignSelf: 'center',
  padding: 10,
  margin: 10,
},
buttonText: {
  textAlign: 'center',
  fontSize: 30,
  fontWeight: 'bold',
},
displayText: {
  textAlign: 'center',
  fontSize: 30,
  color: 'white'
},
imageIcon: {
  width: 150,
  height: 150,
  marginLeft: 75,
}
chunkButton:{
  width: '60%',
  height: 50,
  justifyContent: 'center',
  alignItems: 'center',
  alignSelf: 'center',
  borderRadius: 10,
  margin: 5,
  backgroundColor: 'red'
}
```



What's NEXT?

We will add sounds of the respective phonemes to the buttons.

EXTEND YOUR KNOWLEDGE

1. Map method: https://www.w3schools.com/jsref/jsref_map.asp