REACT PROPS- HOMEWORK

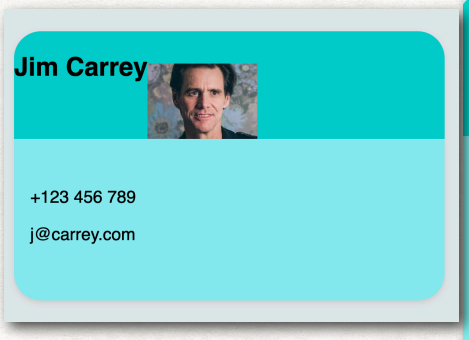
Import six-react-props-practice

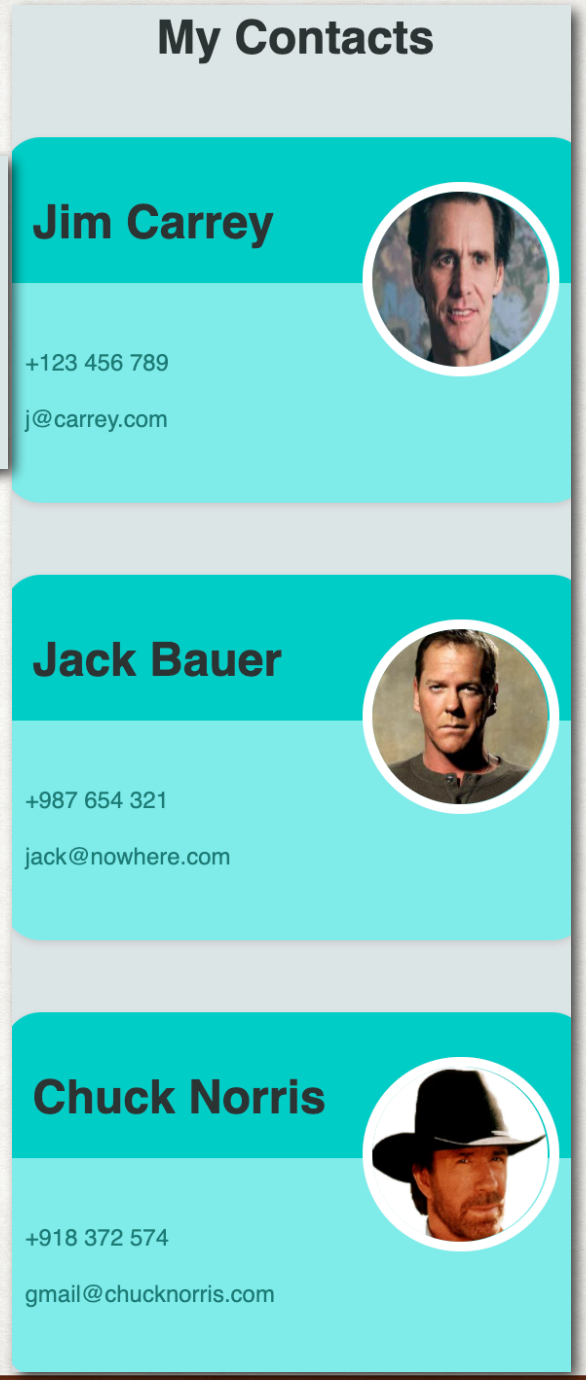
npm install react-scripts start, npm install, npm start

Match the page to the image on the right using the components

TASK STEPS:

```
//1. Apply CSS styles to App.jsx component
//to match the appearance on the completed app:
//2. Extract the contact card as a reusable Card
component.
//3. Use props to render the default Jim Carrey
contact card
//so the Card component can be reused for other
contacts.
//4. Import the contacts.js file to create card
components.
```





STEP 1

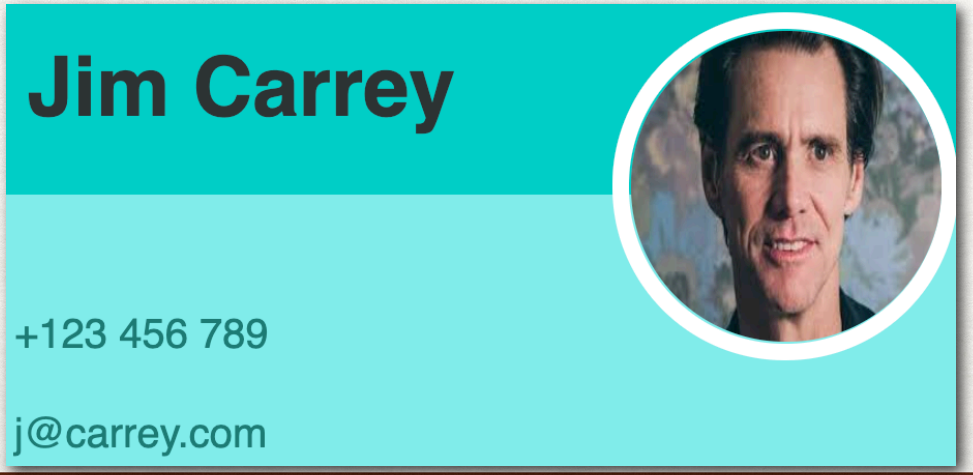
```
//1. Apply CSS styles to App.jsx component to match the appearance
```

We need to apply styles to h1, img, and p tags to style them. We need to

Use className attribute to apply external styles from styles.css sheet

```
<h2 className="name">Jim Carrey</h2>

<img className="circle-img"

<p className="info">+123 456 789</p>

<p className=“info">j@carrey.com</p>
```

NOW WE SHOULD SEE THE THE MATHCING STYLES



JS App.jsx

src > components > JS App.jsx > ...

```jsx
import React from "react";

function App() {
  return (
    <div>
      <h1 className="heading">My Contacts</h1>
      <div className="card">
        <div className="top">
          <h2 className="name">Jim Carrey</h2>
          <img className="circle-img"
            src="https://encrypted-tbn0.gstatic.
            alt="avatar_img"
          />
        </div>
        <div className="bottom">
          <p className="info">+123 456 789</p>
          <p className="info">j@carrey.com</p>
        </div>
      </div>
    </div>
  );
}

export default App;
```

STEP 2

```
//2. Extract the contact card as a reusable Card component.
```

1.Create Card.jsx in the components folder, Import React, create the Card function, return div that holds Card component.Export.

2.In App component, import Card and Add the <Card/> component in the code. DONE

App.jsx | Card.jsx | contacts.js

src > components > App.jsx > App

2

```jsx
import React from "react";
import Card from "./Card";

function App() {
  return (
    <div>
      <h1 className="heading">My Contacts</h1>
      <Card />
    </div>
  );
}

export default App;
```

op.jsx | Card.jsx | contacts.js

components > Card.jsx > default

1

```jsx
import React from "react";

function Card(){
    return <div className="card">
    <div className="top">
      <h2 className="name">Jim Carrey</h2>
      <img className="circle-img"
        src="https://encrypted-tbn0.gstatic.co
        alt="avatar_img"
      />
    </div>
    <div className="bottom">
      <p className="info">+123 456 789</p>
      <p className="info">j@carrey.com</p>
    </div>
  </div>
}

export default Card;
```

STEP 3

```
//3. Use props to render the default Jim Carrey contact card so the Card component can be reused for other contacts.
```

In the App component, in the <Card />, create custom properties for the required info:

1. Check which properties are reusable. Decide creating properties: **name, img, tel, email**

2. Inside the App.jsx, In the <Card />, Create the props in the Card component key-value pairs:

```
<Card
      name="Jim Carrey"
      img="https://encrypted-tbn0.gstatic.com/images?
q=tbn:ANd9GcSVl6f7CWwdVCgb_bEHUT9ZuoY8wz50EET_dw&usqp=CAU”
      tel="+123 456 789”
      email="j@carrey.com"
      />
```

3. Inside the Card.jsx, changing the hard coded parts by getting the value of the properties using props:

-> In Card component,Create props,

then use props.custom property keys

To get he values

STEP 3 DONE!

2

```
function App() {
  return (
    <div>
      <h1 className="heading">My Contacts</h1>
      <Card
      name="Jim Carrey"
      img="https://encrypted-tbn0.gstatic.com/images?q=t
      tel="+123 456 789"
      email="j@carrey.com"
      />
    </div>
  );
}
```

3

```
function Card(props){
    return (
        <div className="card">
        <div className="top">
          <h2 className="name">{props.name}</h2>
          <img className="circle-img"
            src={props.img}
            alt="avatar_img"
          />
        </div>
        <div className="bottom">
          <p className="info">{props.tel}</p>
          <p className="info">{props.email}</p>
```

STEP 4

```
//4. Import the contacts.js file to create card components.
```

1. contact.js file has the contact information as an array of javascript object! We will get data from contacs.js file for dynamic coding Currently they are hard coded.
   name, img, tel, email.

```
const contacts = [
  {
    name: "Jim Carrey",
    imgURL: "https://encryp
    phone: "+123 456 789",
    email: "j@carrey.com"
  },
```

```
function App() {
  return (
    <div>
      <h1 className="heading">My Contacts</h1
      <Card name = "Jim Carrey"
      img="https://encrypted-tbn0.gstatic.
       tel="+123 456 789"
       email="j@carrey.com"
```

3. To use contact.js in other class, EXPORT export default contacts;

```
export default contacts;
```
2

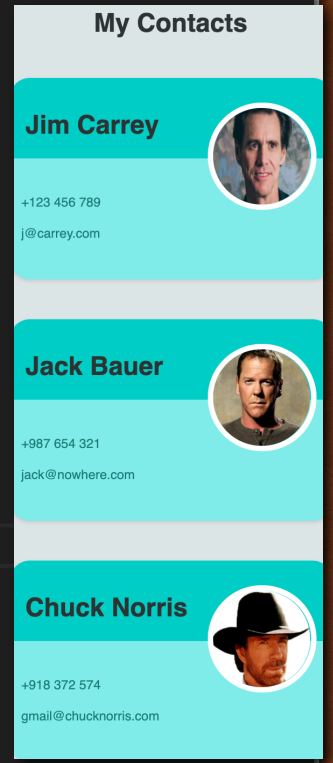
3. Import contact in App.jsx: import contacts from "../contacts"; **!!! Use .. /for path**

4.Use first contact with index, copy paste to get all 3 contacts:

```
<Card
      name={contacts[0].name}
      img={contacts[0].imgURL}
      tel={contacts[0].phone}
      email={contacts[0].email}
      />.  ON THE RIGHT
```

```
import contacts from "../contacts";
//Testing if i am able to read contact component
// console.log(contacts)
// console.log(contacts[0].name)

function App() {
  return (
    <div>
      <h1 className="heading">My Contacts</h1>
      <Card
      name={contacts[0].name}
      img={contacts[0].imgURL}
      tel={contacts[0].phone}
      email={contacts[0].email}
      />
      <Card
      name={contacts[1].name}
      img={contacts[1].imgURL}
      tel={contacts[1].phone}
      email={contacts[1].email}
      />
      <Card
      name={contacts[2].name}
      img={contacts[2].imgURL}
      tel={contacts[2].phone}
      email={contacts[2].email}
      />
    </div>
```
3

4



10. REACT DEV TOOL

We will create another component just for rendering Avatar. So identify Avatar related code and return in Avatar component

1. Create a component under components, **Avatar**, And return avatar related code, which is img tag in Card component.

Avatar.jsx

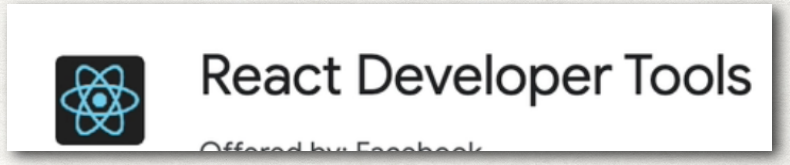
```jsx
import React from "react";

function Avatar(props) {
  return <img className="circle-img"
  src={props.img} alt="avatar_img" />;
}

export default Avatar;
```

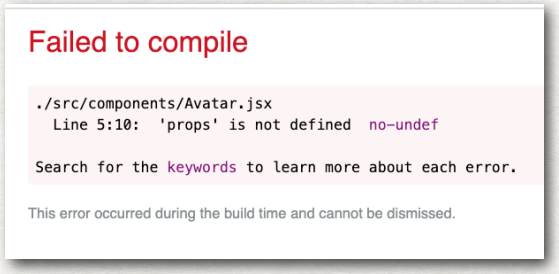
2. Then add Avatar component in the Card component. We need to pass image property in the Avatar component. Otherwise code will break

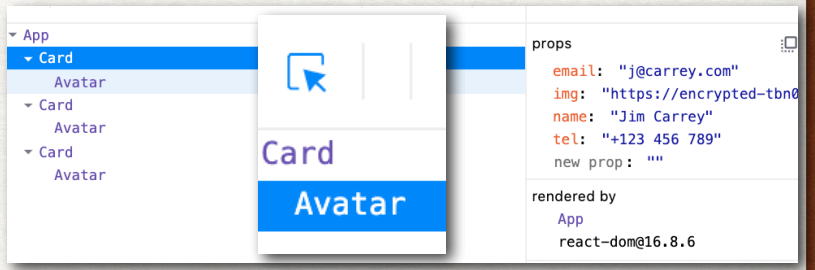
We can use react dev tool to identify the missing components

ar.jsx ● Card.jsx ● App.jsx

```jsx
import React from "react";
import Avatar from "./Avatar";

function Card(props) {
  return (
    <div className="card">
      <div className="top">
        <h2 className="name">{props.name}</h2>
        <Avatar img={props.img} />
      </div>
```







REACT DEV TOOL

We need to pass the property two levels:

<App /> => <Card /> => <Avatar />

We already passed from App to card

Now We must pass it from Card to Avatar

1. In Card.jsx <Avatar img ={props.img}/>

Then add props to Avatar Component

2. In Avatar.jsx: function Avatar(props){. DONE

```
function App() {
  return (
    <div>
      <h1 className="heading">My Contacts</h1>
      <Card name = {contacts[0].name}
        img={contacts[0].imgURL}
```

APP COMPONENT

```
function Card(props){
    return <div className="card">
    <div className="top">
      <h2 className="name">{props.name}</h2>
      <Avatar img={props.img}/>
```

CARD COMPONENT

```
function Avatar(props){
    return <img
    className="circle-img"
      src={props.img}
      alt="avatar_img" />
}
```

AVATAR COMPONENT

React dev tool helps us to identify which components has which props

ADD THAT AS CHROME EXTENSION

Go to airbnb

Check rect dev tool in production mode

We can check the source code, change the code, etc with react dev tool

ADD A IMAGE USING AVATAR QUICKLY

So we can split our website into smaller and smaller compost and do change on that small components when needed. This is the whole point of the react component. Reusability

It makes it faster and effective to develop our front end. We can reuse this Avatar component in other components. For example we add your favorite players image on the main page.

All we have to do is Create <Avatar img="img url"/> in the App component under header

1. In App.jsx add <Avatar and use img property to pass the image

So just using the simple Avatar component, added an Avatar

```
function App() {
  return (
    <div>
      <h1 className="heading">My Contacts</h1>
      <Avatar img ="
      <Card
```

SO FAR OUR COMPONENT

App.jsx | Avatar.jsx | Card.jsx

src > components > App.jsx > App

```jsx
import React from "react";
import contacts from "../contacts";
import Avatar from "./Avatar";
import Card from "./Card";

function App() {
  return (
    <div>
      <h1 className="heading">My Contacts</h1>
      <Avatar img="https://media.wfaa.com/assets
      <Card name = {contacts[0].name}
        img={contacts[0].imgURL}
        tel={contacts[0].phone}
        email={contacts[0].email}
      />
        <Card name = {contacts[1].name}
        img={contacts[1].imgURL}
        tel={contacts[1].phone}
        email={contacts[1].email}
      />
        <Card name = {contacts[2].name}
        img={contacts[2].imgURL}
        tel={contacts[2].phone}
        email={contacts[2].email}
      />
    </div>
  );
}

export default App;
```

```jsx
import React from "react";

function Avatar(props){
    return  <img
className="circle-img"
    src={props.img}
    alt="avatar_img"
  />
}

export default Avatar;
```

```jsx
import React from "react";
import Avatar from "./Avatar";
import Details from "./Details";

function Card(props){
    return <div
className="card">
    <div className="top">
      <h2
className="name">{props.name}</
h2>
      {/* <img
className="circle-img"
        src={props.img}
        alt="avatar_img"
      /> */}
      <Avatar img ={props.img}/>
    </div>
    <div className="bottom">
      {/* <p
className="info">{props.tel}</p>
      <p
className="info">{props.email}</
p> */}
      <Details
detailsInfo={props.tel}/>
      <Details
detailsInfo={props.email}/>
    </div>
  </div>
}

export default Card;
```

```jsx
import React from "react";

function Details(props){
    return <p
className="info">{props.detailsI
nfo}</p>
}

export default Details;
```

Index.js

```jsx
import React from "react";
import ReactDOM from "react-dom";
import App from "./components/App";

ReactDOM.render(<App />,
document.getElementById("root"));

//1. Apply CSS styles to App.jsx component
//to match the appearance on the completed app:
//2. Extract the contact card as a reusable Card
component.
//3. Use props to render the default Jim Carrey
contact card
//so the Card component can be reused for other
contacts.
//4. Import the contacts.js file to create card
components.
```

11. MAPPING DATA TO COMPONENTS AND KEY PROPERTY

App.jsx component has a lot of repetitive code. We can short the code using js map function

We can use map function to loop the repetitive data in the custom component

Map function is a javascript function that is useful to handles arrays, like our contacts array that is in the contact.js

Contact array currently has 3 items, each item has javascript objects that has key-value pairs

1. **In App.jsx comment out ALL <Card /> components**
2. **Call contacts array, call map function in curly braces, and pass a createCard function. {contacts.map(createCard)}**
3. **Create the function, createCard(), that function returns a custom <Card component with some custom properties**

```
{contacts.map(createCard)}
```

5. **We went to get the each contact object in the contacts array, and get each property(name,imgURL,...)**

1. **Map function expects actual function. We call map function to pass another function. This is called functional programming. It means, we use functions within a function, rather that a value**
2. **What functionality do we want to happen each of our contact?**
3. **Add those functionaties**

```
function App() {
  return (
    <div>
      <h1 className="heading">My Contacts</h1>
      {contacts.map(createCard)}
      {/* <Card
        name={contacts[0].name}
```

1 2

```
function createCard(contact){
  return <Card
  name={contact.name}
  />
}
```

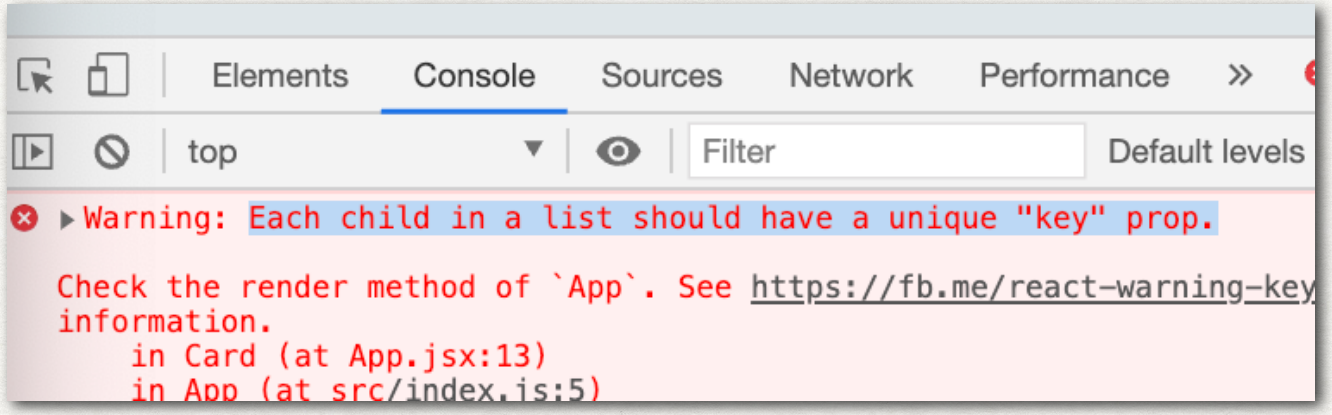
3

```
function createCard(contact){
  return <Card
  name={contact.name}
  img={contact.imgURL}
  tel={contact.phone}
  email={contact.email}
  />
}
```

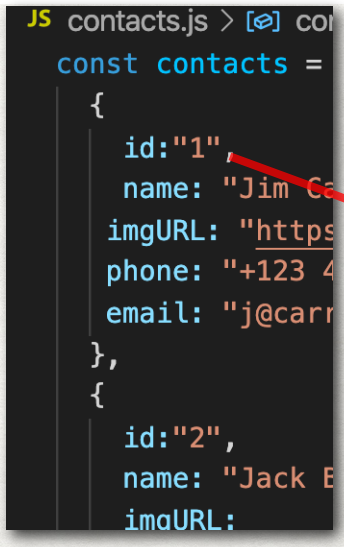
4

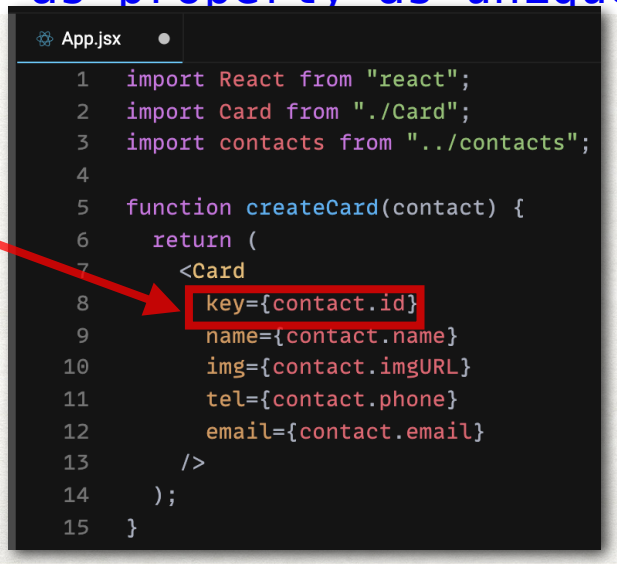
11. MAPPING DATA TO COMPONENTS AND KEY PROPERTY

1. THIS WORKS BUT THERE IS AN ERROR IN THE CONSOLE, CAUSE WE NEED TO USE A UNIQUE KEY IN THE CARD COMPONENT TO EFFICIENTLY RENDER COMPONENTS TO MAKE SURE EVERY SINGLE COMPONENT HAS A UNIQUE ELEMENT.



key={contact.id} -> Add as property as unique identifier





We have to assign unique property for each the object.

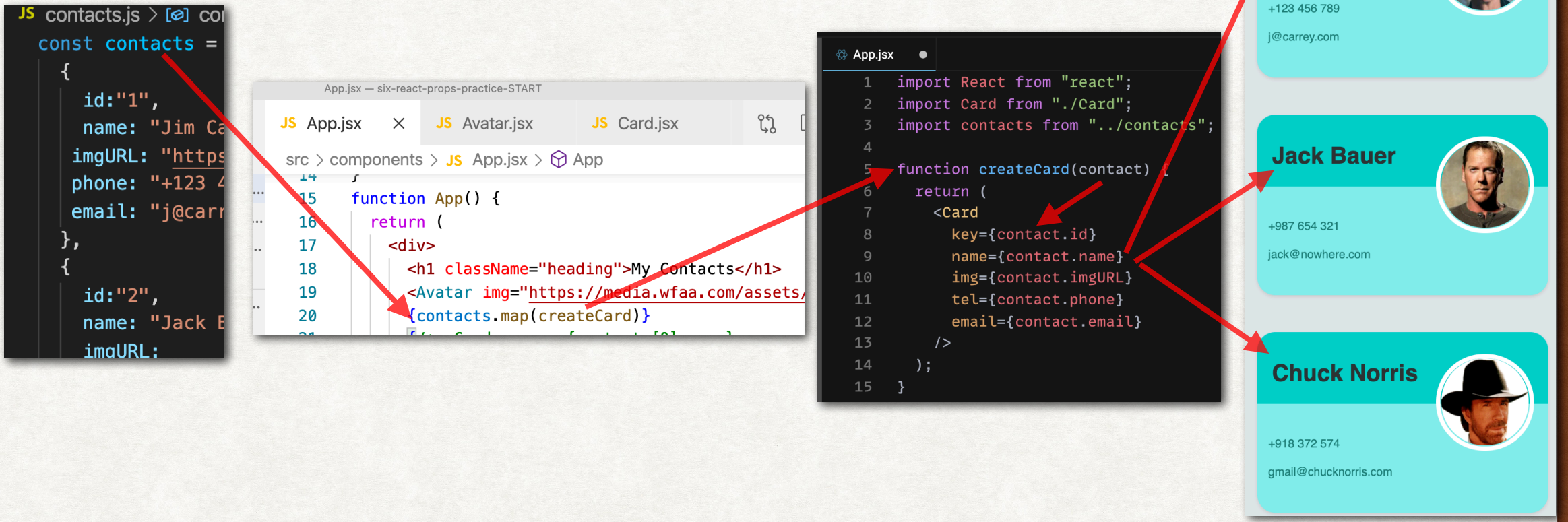
It will help identify each different component

For example we have id that is unique for each Card component so we use id for the key value like this.

If there was no unique id, We would ADD IT IN contacts ARRAY

So add key={contact.id} in the createCard so each card can be identified uniquely. Now error will be gone

MAPPING DATA TO COMPONENTS-WORK FLOW



MAPPING DATA TO COMPONENTS PRACTICE

Import seven mapping components practice .

npm install react-scripts start, npm install, npm start

TASK:

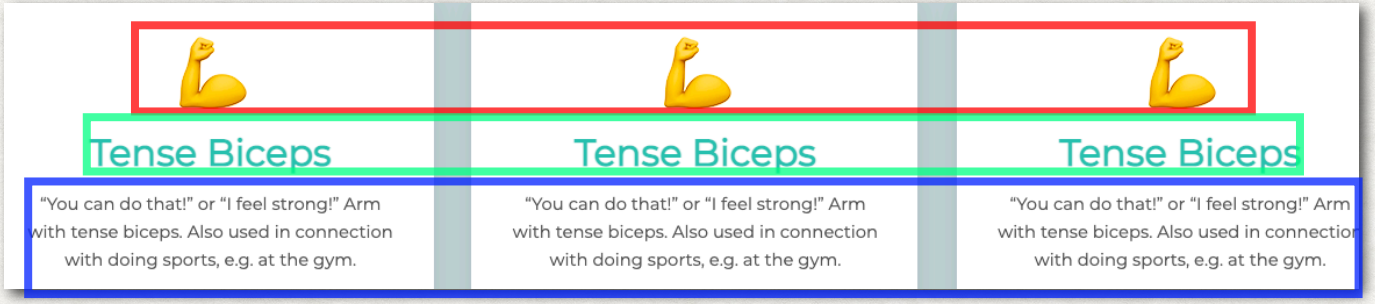
1. Create Entry.jsx component ,import react, create function, export, return the entire div className="term" in Entry.jsx
2. Create props to replace the hard coded part
3. Map through the emojipedia Entry.jsxarray and render Entry components

MAPPING DATA TO COMPONENTS PRACTICE

Analyze the App.jsx html code and determine what needs to taken out. What is keep repeating that will a good candidate to be a seperate as component

Parts that is repeating: The image of the emoji, the name of the emoji, and the description of the emoji.

So we can use create props and map function to render these 3 data.



MAPPING DATA TO COMPONENTS PRACTICE

1. **Create Entry.jsx component ,import react, create function, export, return the entire div className=“term” in Entry.jsx**

1. Create Entry.jsx component ,import react, create function, export, return the entire div className=“term” in Entry.jsx

```
components > JS Entry.jsx > default
import React from "react";

function Entry(){
    return( <div className="term">
    <dt>
      <span className="emoji" role="img" aria-label="Tense Biceps">
        💪
      </span>
      <span>Tense Biceps</span>
    </dt>
    <dd>
      “You can do that!” or “I feel strong!” Arm with tense biceps. Also
      used in connection with doing sports, e.g. at the gym.
    </dd>
  </div>
  )
}
export default Entry;
```

2. In App.jsx, delete repared div=“className” and add <Entry />.

We successfully extracted the Entry component from App component

But we see same data in the Entry components.

```
src > components > App.jsx > App
1   import React from "react";
2   import Entry from "./Entry";
3
4   function App() {
5     return (
6       <div>
7         <h1>
8           <span>emojipedia</span>
9         </h1>
10
11        <dl className="dictionary">
12          <Entry />
13          <Entry />
14          <Entry />
15
16        </dl>
17      </div>
18    );
19  }
20
21  export default App;
```

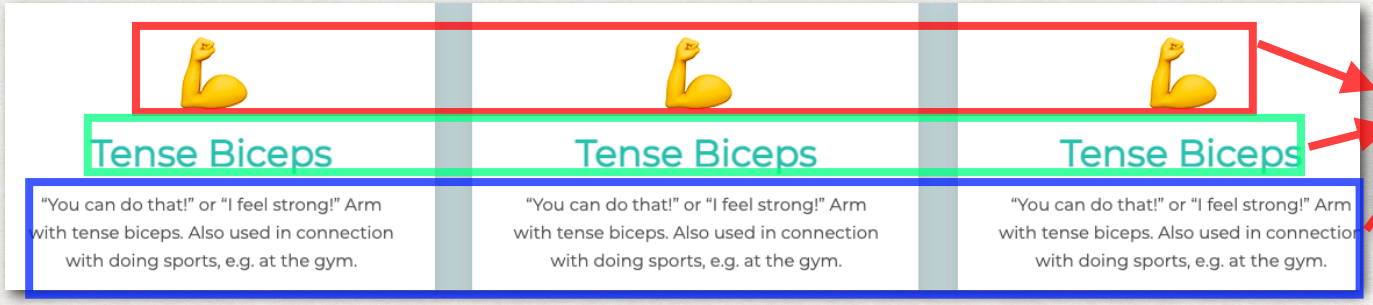
MAPPING DATA TO COMPONENTS PRACTICE-2. CREATE PROPS TO REPLACE THE HARD CODED PART

2. **Create the props for the Entry component to replace the hard coded data**

to create reusable dynamic component for the 3 things that keeps changing: emoji, name, description.

1. In App.jsx, create 3 props for Entry: emoji, name,description. These are the props we will capture in the Entry component

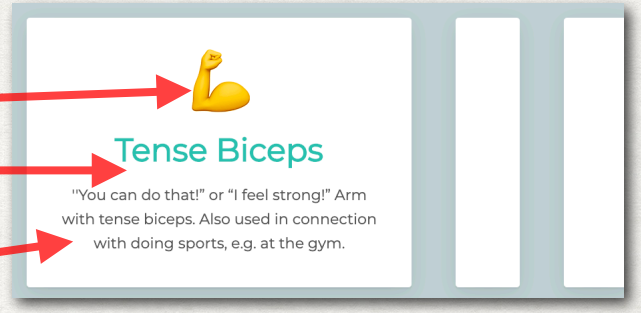
2. In Entry, pass props, and send props.emoji, props.name, props.descriptions




```
<Entry emoji="💪"
  name="Tense Biceps"
  description="'You can do that!" or "I feel strong!" Arm with tense biceps. Also
      used in connection with doing sports, e.g. at the gym.'"
/>
```

```
function App() {
  return (
    <div>
      <h1>
        <span>emojipedia</span>
      </h1>

      <dl className="dictionary">
        <Entry
        emoji="💪"
        name="Tense Biceps"
        description="'You can do that!" or "I feel strong!" Arm with tens
        <Entry />
        <Entry />

      </dl>
    </div>
  );
```

```
function Entry(props){

  return (
    <div className="term">
      <dt>
        <span className="emoji" role="img" aria-label="Tense Biceps">
          {/* 💪 */}
          {props.emoji}
        </span>
          <span>{props.name}</span>
      </dt>
      <dd>
        {/* "You can do that!" or "I feel strong!" Arm with tense bice
        used in connection with doing sports, e.g. at the gym. */}
          {props.description}
      </dd>
    </div>
  );
```



MAPPING DATA TO COMPONENTS PRACTICE

3. Map through the emojipedia Entry.jsxarray and render Entry components

3a. Import the Emojipedia const.

1. Export emojipejipedia from emojipedia.js:

```
];
export default emojipedia;
```

2. Then import in the App.jsx :Import emojipedia from "../emojipedia";

3b. Map thorough the emoji

1. Delete All our <Entry component from App.jsx:

2.Then use map function on the same place to get the elements from emojipedia array elements :

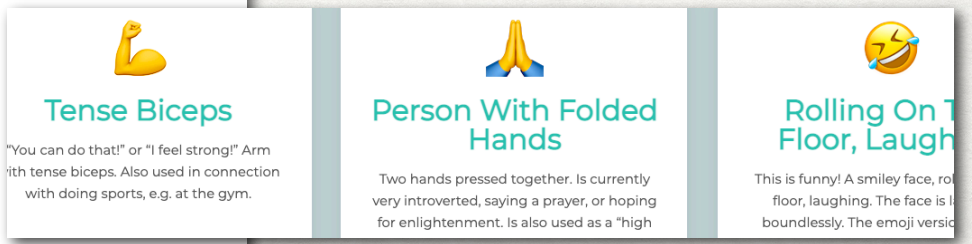
```
{emojipedia.map(createEntry)}
```

3. Remember this map will takes a function. So let's create a function in this App.jsx and name is createEntry.

App.jsx | emojipedia.js | Entry.jsx

src > components > App.jsx > createEntry

```
import React from "react";
import Entry from "./Entry";
import emojipedia from "../emojipedia";

function createEntry(emojiTerm){
return <Entry
    key={emojiTerm.id}
    emoji={emojiTerm.emoji}
    name={emojiTerm.name}
    description={emojiTerm.description}
/>
}

function App() {
  return (
    <div>
      <h1>
        <span>emojipedia</span>
      </h1>

      <dl className="dictionary">
      {emojipedia.map(createEntry)}

      </dl>
    </div>
  );
}

export default App;
```



MAPPING DATA TO COMPONENTS PRACTICE

We are going to pass some props from the Entry.jsx :emoji, name, description props we will pass. So create props in here and pass the Entry props here:

REMEMBER whenever we use map function to loop an array to render a react component we must add a key props that is expected by react so it can eficiently renders the props

```
return (
    <Entry
    key={emojiTerm.id}
    emoji={emojiTerm.emoji}
    name={emojiTerm.name}
    description={emojiTerm.description}
    />);
```

Now we are getting the data form the emojipedia.js dynamically.
No we can add any object in emojipediade or change the data in that file, then it will automatically reflects on the UI

components > App.jsx > App

```jsx
import React from "react";
import Entry from "./Entry";
import emojipedia from "../emojipedia";

// console.log(emojipedia);

function createEntry(emojiTerm){
  return (
    <Entry
    key={emojiTerm.id}
    emoji={emojiTerm.emoji}
    name={emojiTerm.name}
    description={emojiTerm.meaning}
    />
  );
}

function App() {
  return (
    <div>
      <h1>
        <span>emojipedia</span>
      </h1>

      <dl className="dictionary">
        {/* <Entry

        emoji="💪"
        name="Tense Biceps"
        description="''You can do that!" or "I feel strong!" Arm wi
        <Entry />
        <Entry /> */}
        {emojipedia.map(createEntry)}

      </dl>
    </div>
  );
}

export default App;
```

13. KEEPER APP PROJECT PART 2

Import eight-keeper-app-part-2

npm install react-scripts start

npm install

npm start

OR USE THE COMPLETED FILE FROM PART 1

KEEPER APP PROJECT PART 2

In part 2, we will apply the latest knowledges we learned

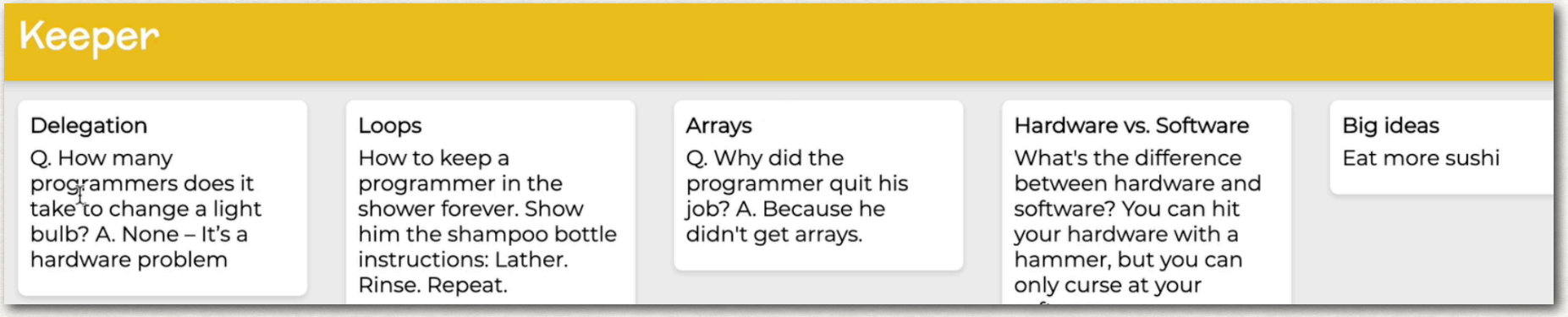
TASK://Challenge. Render all the notes inside notes.js as a seperate Note component.

In the Note we currently have h1 and p which is hard coded

```
<h1>This is the note title</h1>
 <p>This is the note content</p>
```

We will use props to make is Dynamic. We will use props to pass the data from notes.js to here

We will apply props and mapping. We should see below screenshot when do that.



KEEPER APP PROJECT PART 2

TASK://Challenge. Render all the notes inside notes.js as a seperate Note component.

1. In App.js, pass some custom props :title and content Then pass the values that is in h1 and p

```
<Note title="This is the note title"
      content="his is the note content"
```

2. In Note.js, Insert the title and content in ht Note.js using props:

```
<h1>{props.title}</h1>
 <p>{props.content}</p>
```

THIS RENDERS THE APP THE SAME BUT How to I render multiple notes that is in the note.js: Use

Map function instead of creating multiple Note component

3. In notes.js export notes.js, In App.js, import notes.js

notes.js

```
31  ];
32
33  export default notes;
```

App.js

```
5  import notes from "../notes";
```

4. In, App.js, REPLACE <Note /> and map function {notes.map(createNotes)}

5. Create createNote function, it will get single noteItem parameter and return Note components
6. <Note /> component will have props title and content They will get the values using this noteItem. Make sure to use key prop to identify unique component:

```
function createNotes(noteItem){
  return <Note
  key={noteItem.key}
  title = {noteItem.title}
  content = {noteItem.content}/>
}
```

```
function App() {
  return (
    <div>
      <Header />
      <Note title="This is the note title"
      content="his is the note content"
      />
      <Footer />
    </div>
  );
}
```

Note.js

components > JS Note.js > Note

```
import React from "react";

function Note(props) {
  return (
    <div className="note">
      <h1>{props.title}</h1>
      <p>{props.content}</p>
    </div>
  );
}

export default Note;
```

App.js

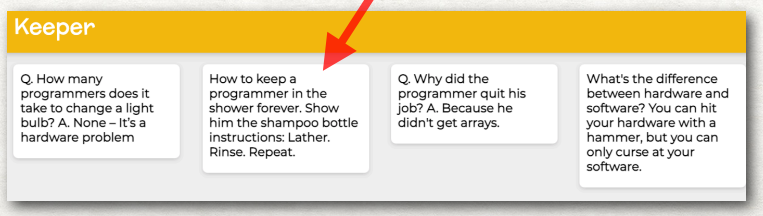
src > components > JS App.js > App

```
1   import React from "react";
2   import Header from "./Header";
3   import Footer from "./Footer";
4   import Note from "./Note";
5   import notes from "../notes";
6
7   function createNotes(noteItem){
8     return <Note
9     key={noteItem.key}
10    title = {noteItem.title}
11    content = {noteItem.content}/>
12  }
13
14  function App() {
15    return (
16      <div>
17        <Header />
18        {/* <Note title="This is the note title"
19        content="his is the note content"
20        /> */}
21        {notes.map(createNotes)}
22        <Footer />
23      </div>
24    );
25  }
26
27  export default App;
```

notes.js

src > JS notes.js > ...

```
1   const notes =
2     {
3       key: 1,
```



KEEPER APP PROJECT PART 2

We can write this code in different ways:

1a. We can put createNote function directly in the map function: So cut createNotes function and paste inside the map:

1b. I can make it anonymous ruction by deleting name:Delete createNotes

1c. Or Also remember we can use arrow functions, so we can simplify arrow to make it looks shorter: Delete function keyword, use =>

1d. Since we return only one single item, we can delete return, (), {}