WHAT IS REACT?

React is a javascript library for building user interface

https://www.airbnb.com/

We create smaller custom html elements with styles And put them inside the big structure

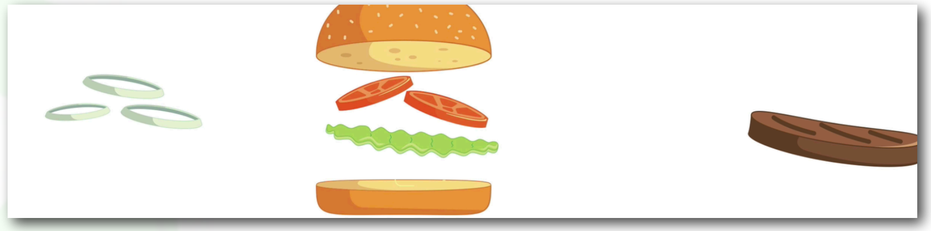
This helps to keep the code clean and reusable components

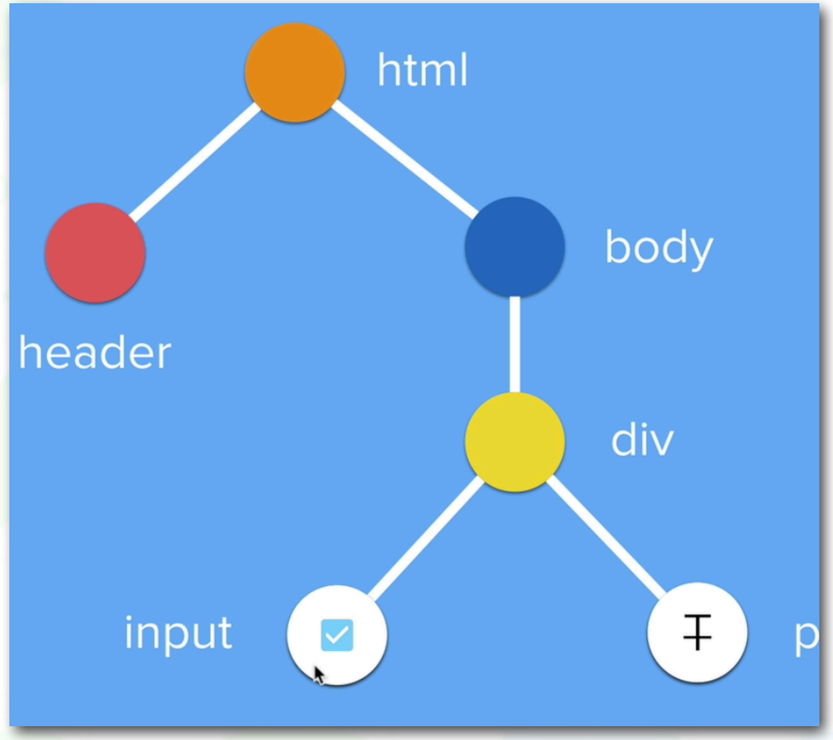
We can think this component as an ingredient inside a burger

We can add small components inside the app like an ingredients

To create a single component, we will use a mix of html, css, and javascript

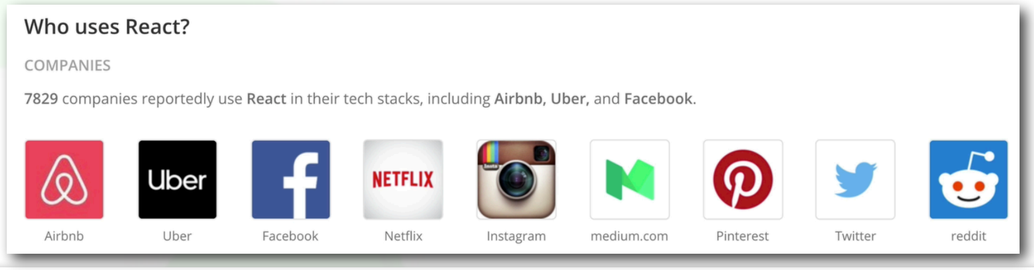
React only renderes/updates the changing components.

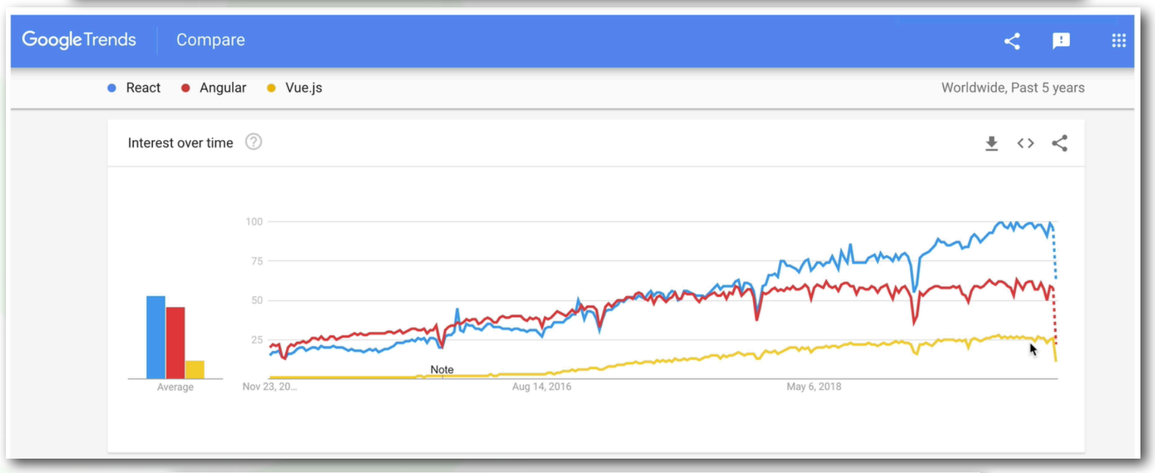


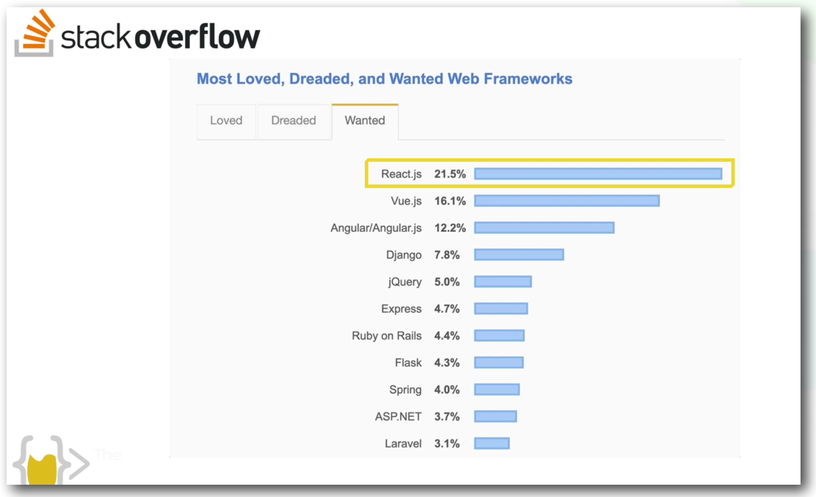


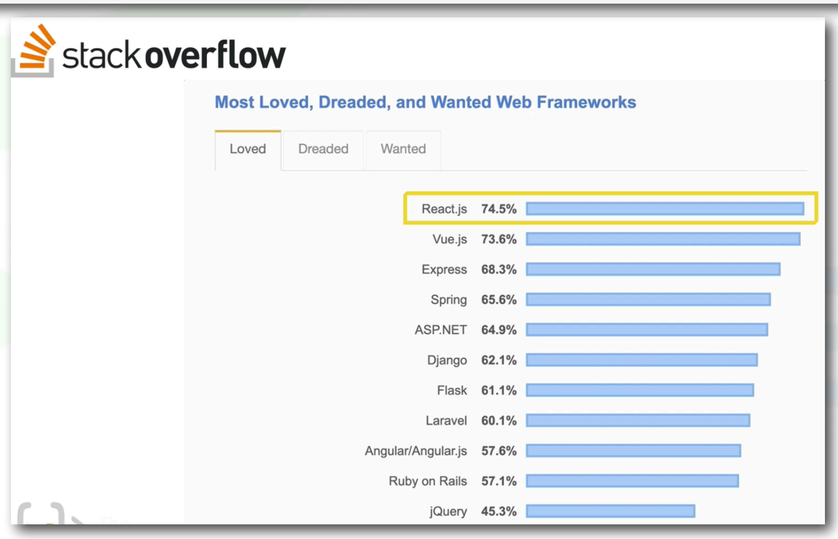
WHY REACT?

- Simplicity
- Predictable
- Flexibility
- Performance
- Testability
- Community support



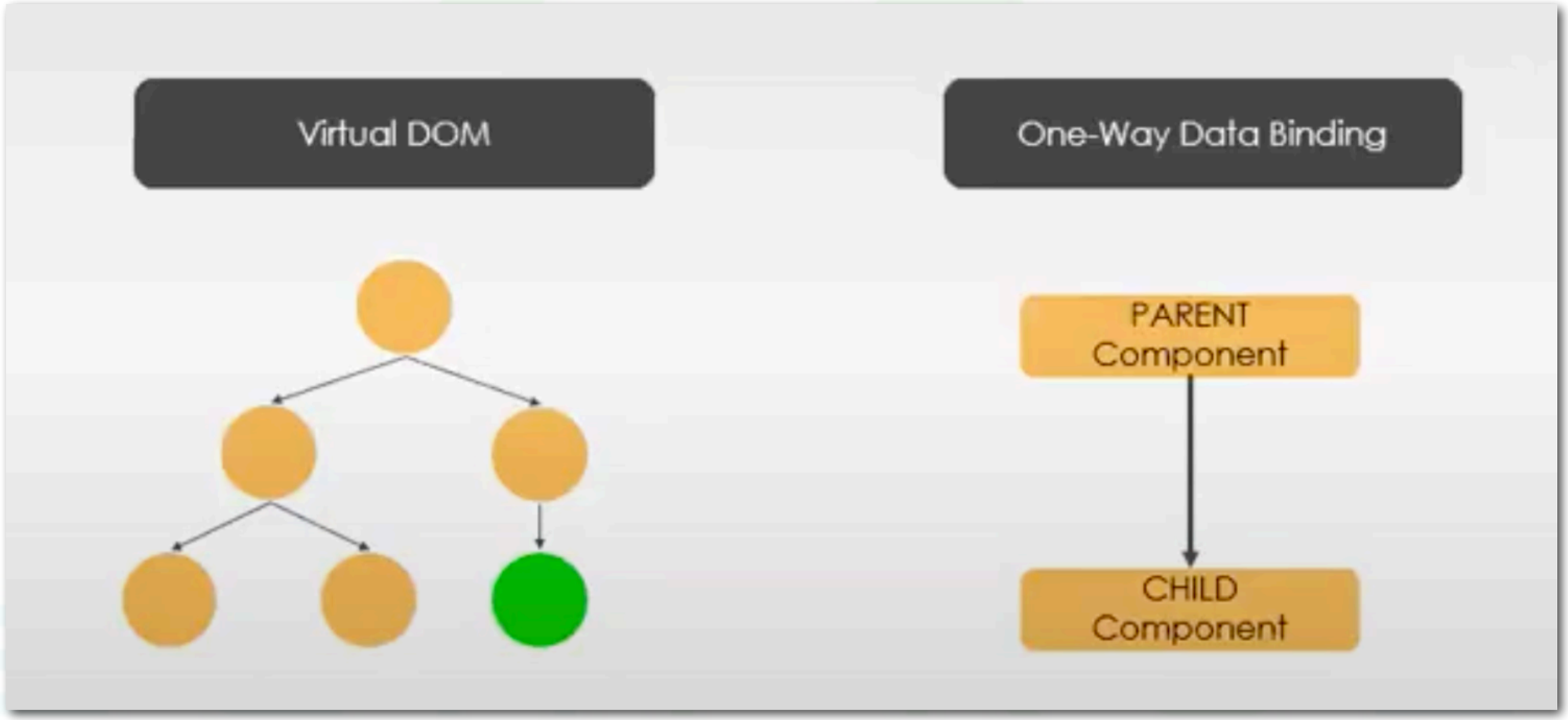






WHY REACH IS FAST

VIRTUAL DOM
ONE-WAY DATA BINDING



WHAT IS VIRTUAL DOM?

Reach DOM-Virtual DOM- Copy of actual DOM

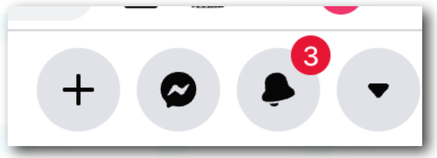
React let us update the page without changing the actual DOM.
(no more document.getElementById("id")).innerHTML='Hello')

React create a virtual copy of actual DOM. Every time there is a chance in the in the virtual DOM, then really update only that changing part. This makes app faster.

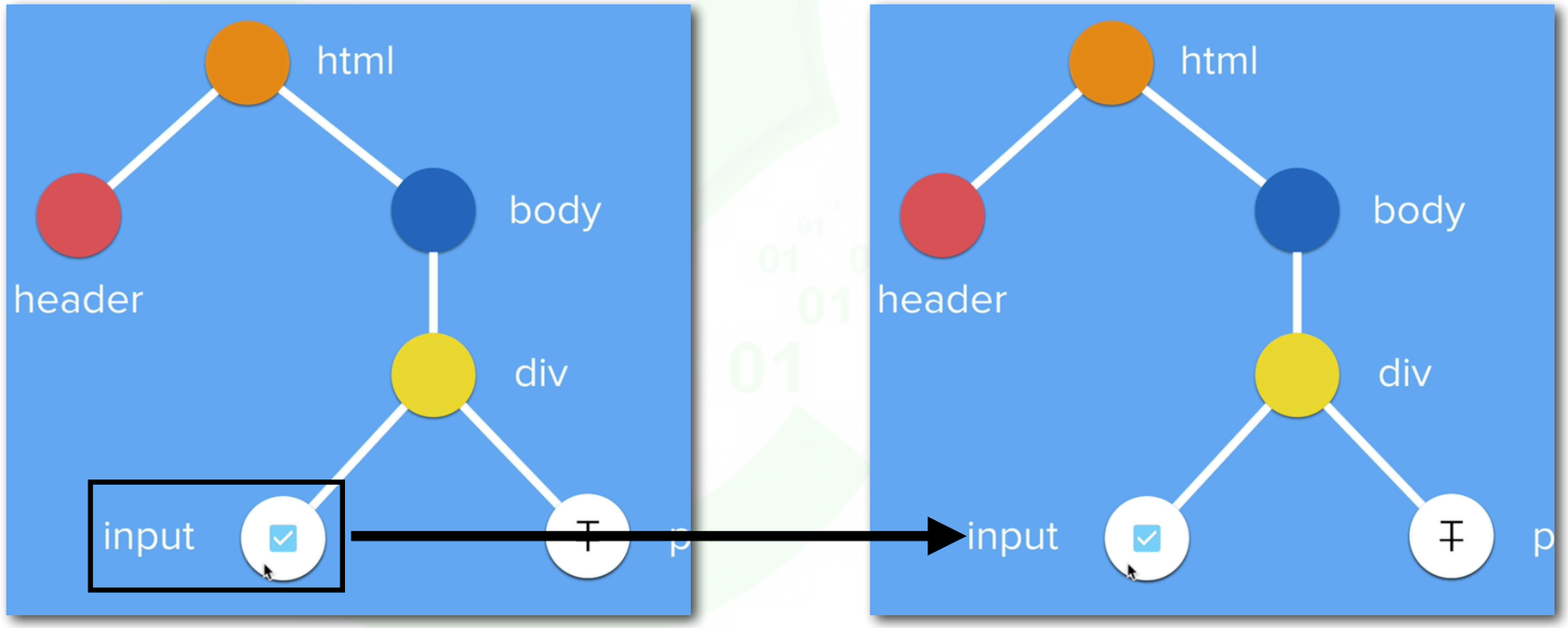
React can detect single component changes and render ONLY the changes, without rendering the entire application. This makes the application faster.

In this DOM tree, react will only render the input if that is the only changing element

For this reason, react help us to handle even small part of the application effectively.



Actual DOM



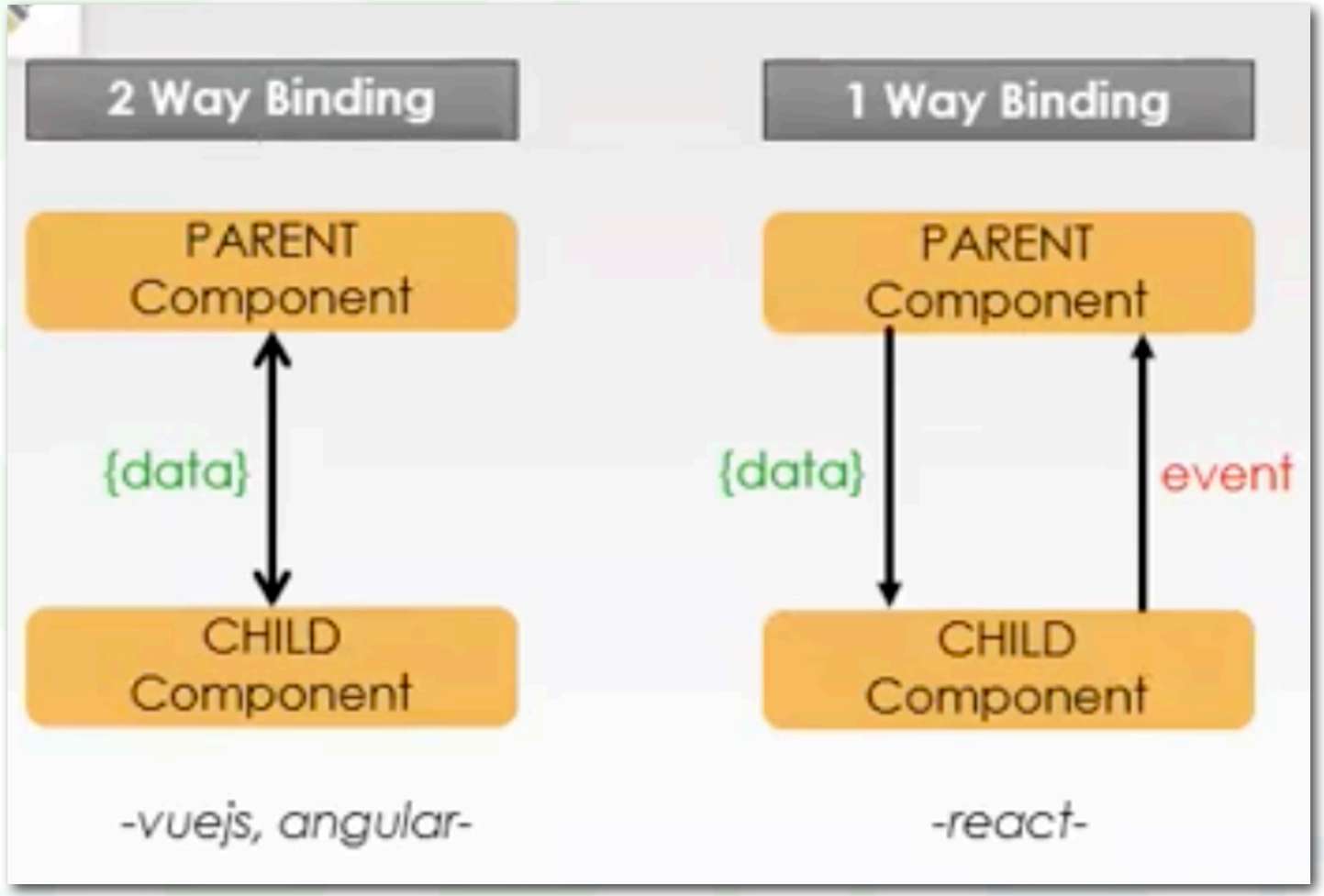
When there is a change in input
Then react will update only this specific element in the actual DOM
This makes app work faster
https://www.youtube.com/watch?v=BYbgopx44vo

ONE WAY DATA BINDING

IN 2 WAY DATA BIDING, WHEN STATE OF A PARENT CHANGES, THEN STATE OF A CHILD CHANGES VICE VERSA.

IN 1 WAY DATA BIDING, WHEN STATE OF A PARENT CHANGES, THEN STATE OF A CHILD. BUT WHEN STATE OF CHILD CHANGES, ADDITIONAL EVENT MUST BE USED TO REFLECT THE SAME CHANGES IN THE PARENT

ONE WAY DATA BINDING IS A LITTLE FASTER



REACT IS COMPONENT BASED

REACTS IS BASED ON REUSABLE COMPONENTS

COMPONENTS ARE REUSABLE

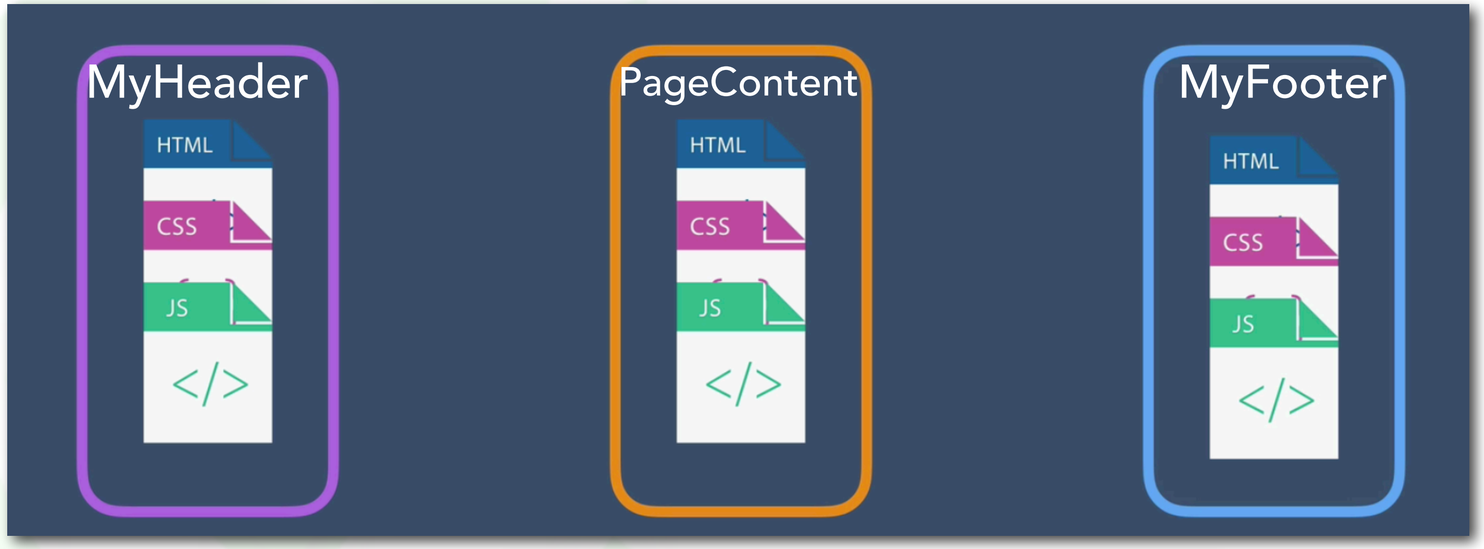
COMPONENTS CAN WORK INDEPENDENTLY

ONE COMPONENT CAN BE USED IN OTHER COMPONENTS

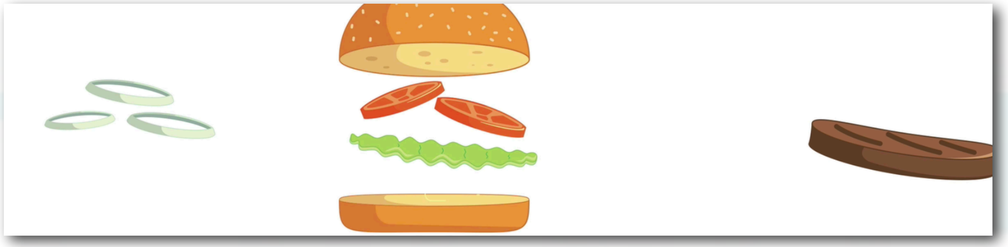
DATA CAN BE CALLED BETWEEN THE COMPONENTS

A COMPONENT CAN HAVE ITS OWN STATE
THIS STATE CAN HOLD THE DATA AND STYLE.
STATES CAN BE USED IN OTHER COMPONENTS

COMPONENTS ARE LIKE INGREDIENTS OF A SANDWICH

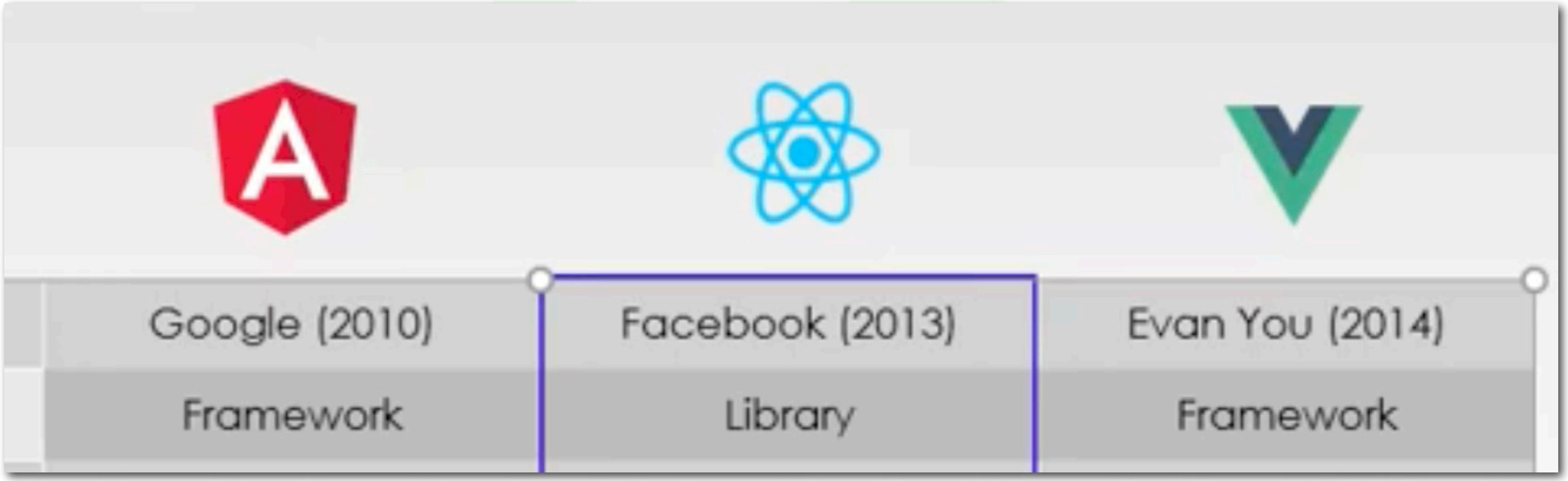



```
<body>
<MyHeader />
<PageContent />
<MyFooter />
</body>
```



REACT ALTERNATIVES

3 POPULAR JAVASCRIPT FRAMEWORK OR LIBRARY



Lesson : Java Dev - Day 2 - Part 2
Topic : Node.js, Npm, React Intro

techproeducation

techproeducation

techproeducation

techproeducation

USEFUL LINK

https://reactjs.org/docs/getting-started.html

https://codesandbox.io/s/new?file=/src/App.js

1. LOCAL ENVIRONMENT SETUP & CREATE FIRST REACT APP

1. Check node, npm. npm —version AND node —version

2. Create a react app using package manager: https://reactjs.org/docs/create-a-new-react-app.html

> Create a folder on your react-classes folder: my-first-react-app

> Open your VS Code terminal and run below command to create a react project

1. Check Node is Up to Date
2. Install VSCode
3. Create React App
4. Run App

npx create-react-app app-name => This will install react related packages

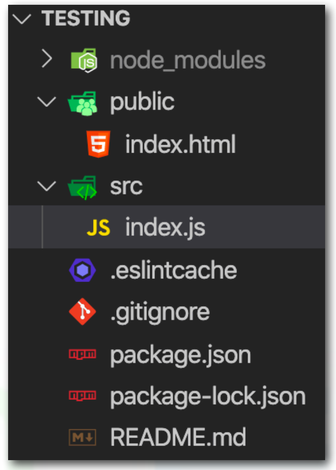
npx create-react-app 01-my-first-app

3. cd app-name ->>>> cd 01-my-first-app

```
cd testing
npm start
```

```
Compiled successfully!
```

4. **npm start** => Now you should see the browser opened react



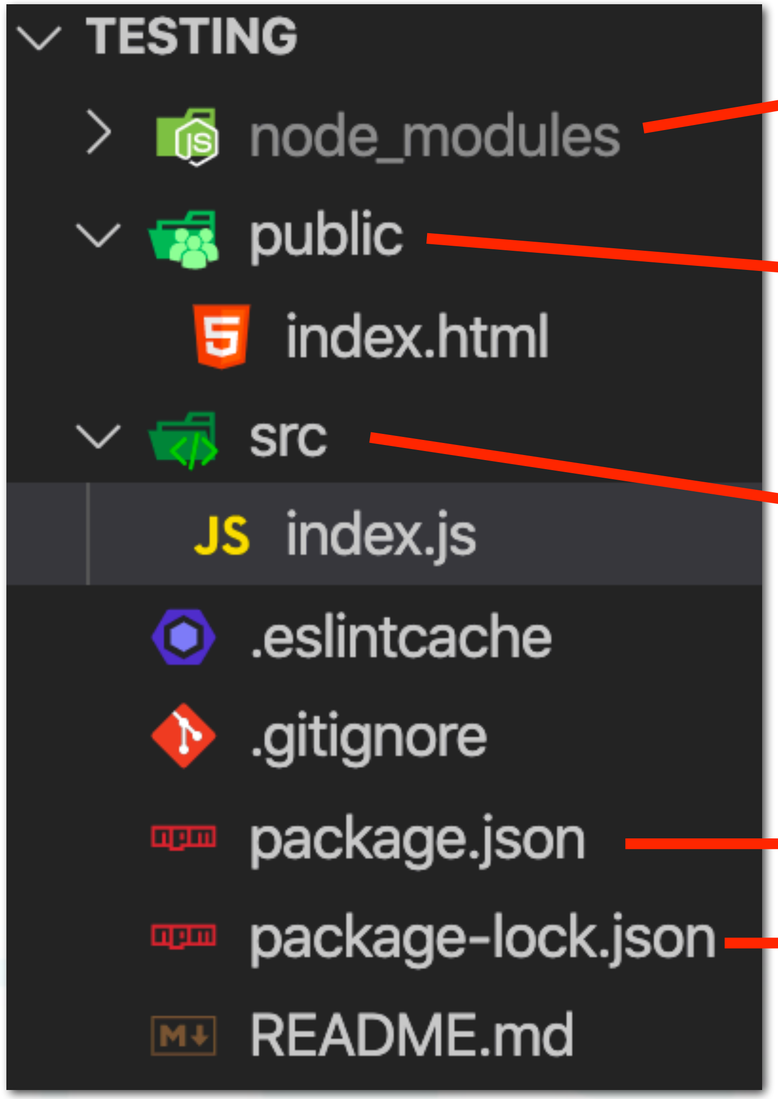
6. Delete the unnececery files. Go to public and delete everything **but index.html**

7. Go to src delete everything **but index.js**

8. Go to **index.html,** delete the unnecessary parts , and **add <script>**

9. Go to **index.js, delete unnecessary files,** write the code to display Hello World.

PROJECT STRUCTURE



Contains all react modules and libraries.
npm install creates this folder
Size is too big so we should not push to git when you share

Public folder has static files, such as index.html, images, data files, etc.
We don't write code in public folder

Src folder has resource files.
index.js, and other component will be created here
This will has the react codes

Project information is stored here.

Project information is stored here.
This is created after npm install

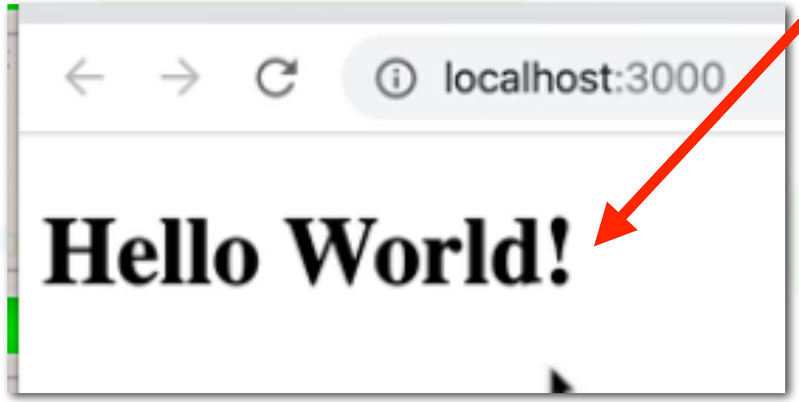
HELLO WORLD

index.html

```
public > index.html > html > body > script
1   <!DOCTYPE html>
2   <html lang="en">
3     <head>
4       <title>React App</title>
5     </head>
6     <body>
7       <div id="root"></div>
8       <script src="./src/index.js" type="text/jsx"></script>
9     </body>
10  </html>
```

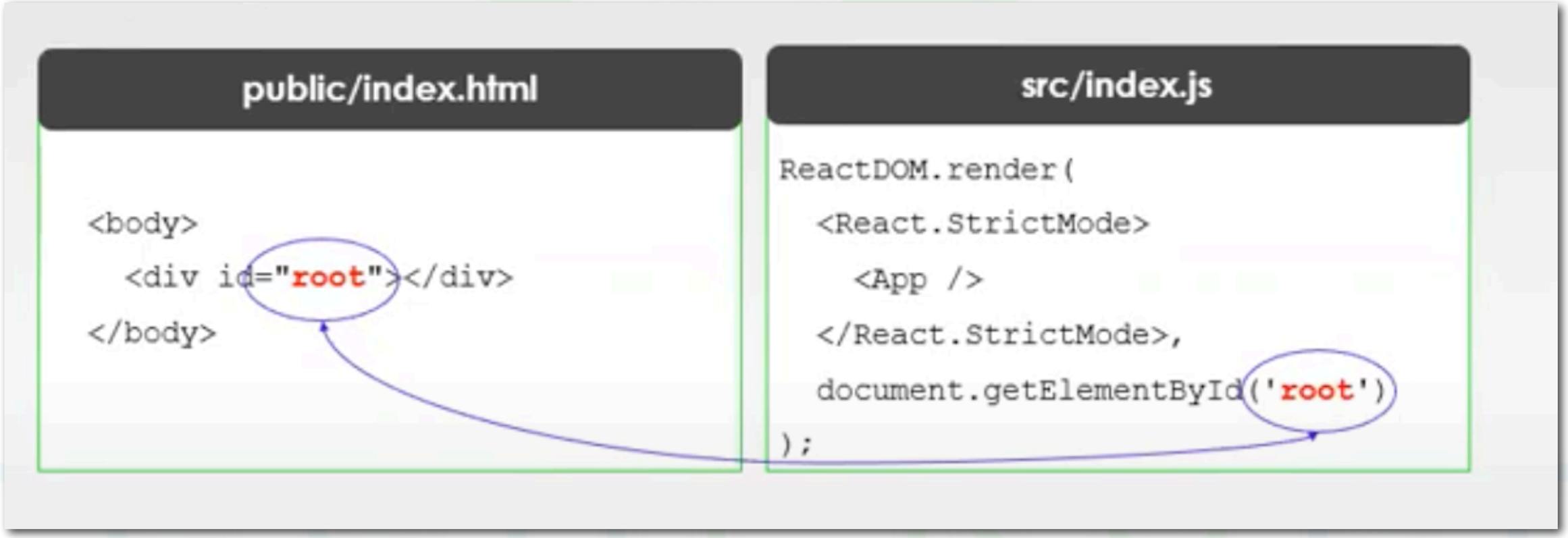
index.js

```
1   import React from 'react';
2   import ReactDOM from 'react-dom';
3
4
5   ReactDOM.render(
6     <div>
7       <h1>Hello Wold!</h1>
8     </div>,
9     document.getElementById('root')
10  );
```



RENDERING

REACTDOM.RENDER GETS THE CODE FROM INDEX.JS AND RENDERS IN THE BODY OF HTML



CREATING A NEW PROJECTS WHEN YOU HAVE EXITING REACT APP

We can create a template for new react apps

That template will not have node_module cause it is too big

To install : npm install

to run : npm start

Steps:

1. Copy the previous code in the my-react-project folder

2. Delete node-module folder

3. Change the name of the folder : template

3. Now that I created a template, I can Copy the template one more time and change name : 02-intro-to-react

4. Open the terminal in vs code and change the directory to this folder

5. install : npm install

6. run : npm start

REACT STARTING FILE

Download the starting folder

Unzip

Open in vs code

In terminal run these scripts:

npm install -> install nod_modules

npm start -> open on local host

ERROR Then RUN: npm install react-scripts start

RUN npm start

Then you should see a blank page

TRADITIONAL HELLO WORLD RENDERING

index.js > ...

```
const ReactDOM = require('react-dom')

// Show Hello World on as h1 in the website.
//TRADITIONAL JS EVELOPMENT
var h1=document.createElement('h1');
h1.innerHTML="Hello World !!!!"
document.getElementById("root").appendChild(h1);
```

Hello World !!!!

```
7
8     <body>
9       <div id="root"></div>
10      <script src="../src/inde
```

WITHOUT REACT AND JSX

index.js U

src > index.js > ...

```
// *************Createing an h1 and displaying on the
//1. creating an h1 element
var heading = document.createElement('h1');
// 2. assign a text value
heading.innerHTML='Hello World!';
// 3. show this on the UI
document.getElementById('root')
.appendChild(heading);
//Then you will see the Hello world on the UI
// *******Creating a paragrapgh and displaying on the
var parag=document.createElement('p');
parag.innerHTML='<i>Hello World!</i>';
document.getElementById('root')
.appendChild(parag)
// *******Creating an input and diplaying on the UI**
var passwordInput = document.createElement('input');
document.getElementById('root')
.appendChild(passwordInput);
//********Giving Line Break */
var lineBreak = document.createElement('hr');
document.getElementById('root')
.appendChild(lineBreak);
//******Creating a button and displaying on the UI***
var submitButton = document.createElement('button');
submitButton.innerHTML='Submit!!!'
document.getElementById('root')
.appendChild(submitButton)
```

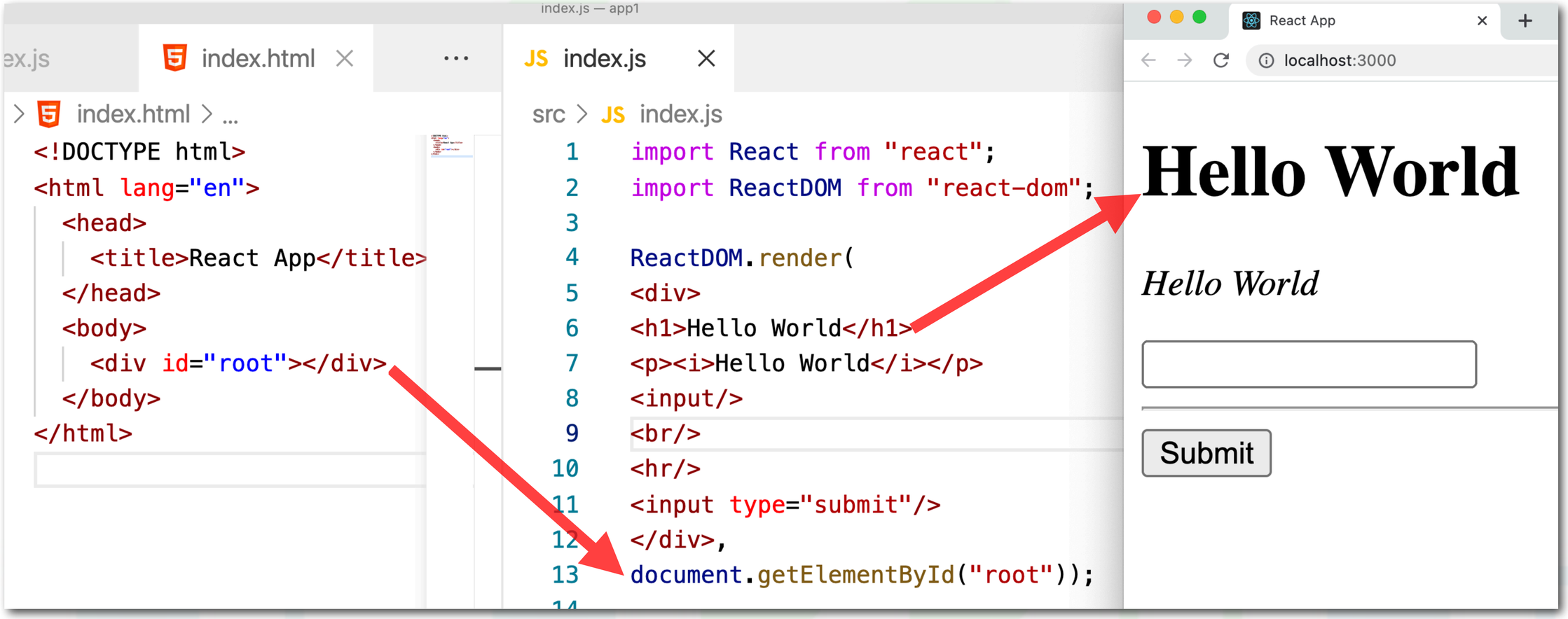
index.html U

public > index.html > html > body

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>React App</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

WITH REACT AND JSX

WE CAN WRITE JSX(HTML) IN JS FILES



JSX

We will not touch index.html. All our code will be written using js in index.js

It has some html code, and

All react apps have an id=root by convention.Everything in our app will go into this div

TASK:

Show Hello World on as h1 in the website.

We are in a js file but we are able to write html code.How this is possible?

React works by creating JSX FILES

Html is picked up and compiled to javascript using react modules.

IF YOU DELETE react("react"), then you get compiler issue

WE SHOW h1 IN in index.html so use document.getElementById("root"))

```
ReactDOM.render(<h1>Hello World</h1>, document.getElementById("root"));//root in the index.html
Render method is used to display elements in the root div
```

index.html

```
<body>
  <!-- <div id="root"></div>=> THIS IS WHERE WE WILL SHOW OUR CODE-
  <div id="root"></div>
  <script src="../src/index.js" type="text/javascript"></script>
  <!-- ../src/index.js => PATH OF index.js FILE
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
```

```
ReactDOM.render("WHAT TO SHOW , WHERE TO SHOW");
```

```
ReactDOM.render(<h1>Hello World</h1> , document.getElementById("root"));
```

Use ReachDOM.render module to render and display h1 html element

JSX VS VANILLA JS

Which one is shorter?

Rendering using react module

```
ReactDOM.render(<h1>Hello World!</h1>, document.getElementById("root"));
```

Rendering using vanilla javascript

```
var h1 = document.createElement("h1");
h1.innerHTML = "Hello World!";
document.getElementById("root").appendChild(h1);
```

```
1 ReactDOM.render(<h1>Hello World</h1>,
  document.getElementById("root"));
```

```
1 "use strict";
2
3 ReactDOM.render(
  /*#__PURE__*/React.createElement("h1", null,
  "Hello World"),
  document.getElementById("root"));
```

JSX

How can we render multiple elements on a page?

How do add a paragraph on the page write under the heading like. **Below code will crash**

```
ReactDOM.render(
<h1>Hello World</h1>
<p >This is the paragraph</p>,
 document.getElementById(“root"));
```

It is because render method takes **ONLY ONE SINGLE HTML ELEMENT.** So we need to turn multiple html elements into one. HOW?? So we can embed this two html element into one single **div** element just like so

Below we have one div element. The div element has two Childs <h1>, and <p>

```
ReactDOM.render(
   <div>
       <h1>Hello World</h1>
       <p>This is the paragraph</p>
   </div>
,
 document.getElementById("root"));
```

```
// var React = require("react");
// var ReactDOM=require("react-dom");
import React from "react";
import ReactDOM from "react-dom";
ReactDOM.render(
<div>
<h1>Hello World</h1>
<p>This is paragraph</p>
</div>,
document.getElementById("root"));
```

REVIEW

- What is react?
  - React is a JS library to build front end applications
  - React allows you to create re-usable and reactive components consisting of HTML and JavaScript (and CSS)
- React is component based application
- What component?
  - Reusable smaller part of the application. It is like an ingredient of a food

    Heading component

    Information component

    Images component …
- What is Virtual DOM?
  - Copy of the actual DOM. When react detects any change, it will update the actual DOM only that part of change. For example, clock is ticking(hh mm ss(ss will change every second)),Every second, only this element will be refreshed and updated, Not the entire page
  - When we make a change in one of the component it doesn't refresh the whole page, it just updates that specific component
  - react updates a component of app by injecting the necessary part without refreshing all page, what makes react project fast
- What is JSX?
  - JSX is an inline markup that looks like HTML and gets transformed to JavaScript. We use JSX to create React elements/component.

BABEL

Babel is a Javascript Compiler:

It takes next generation javascript and compile it down to javascript so every browser even old browser can understand the code

https://babeljs.io/en/repl

Copy react code and paste, then you should see javascript version

JSX CODE TASK

Complete the task:

Create a new Reach app from scratch:

Name: 03-intro-to-JSX

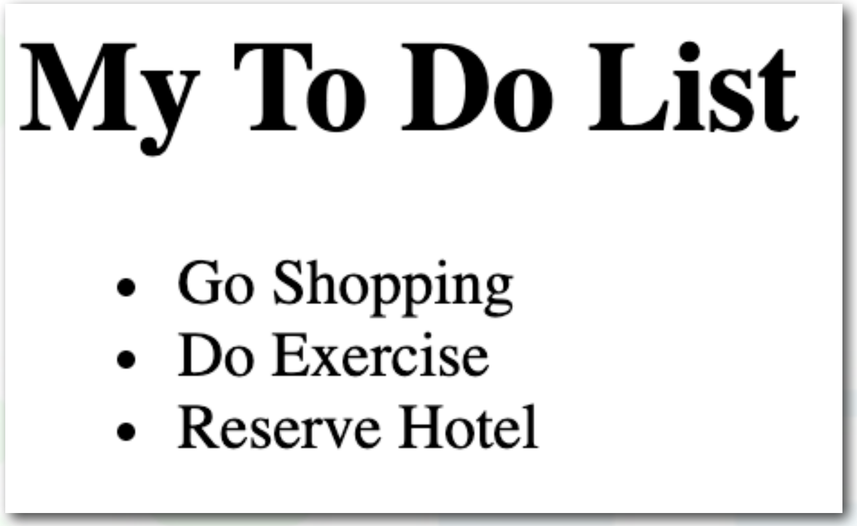
Note that you can use template(RECOMMENDED) or create a new react app from scratch

//It should display a h1 heading.

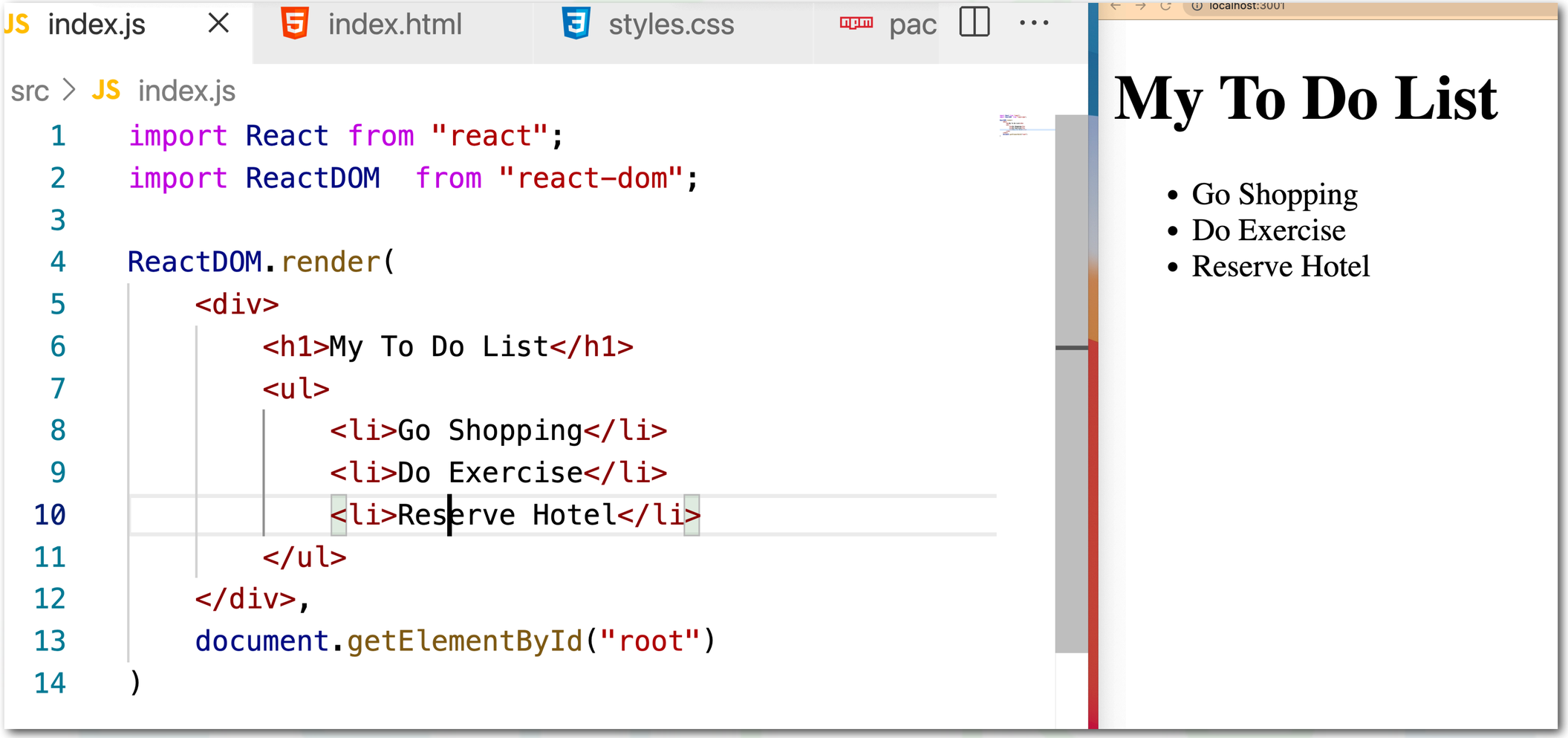
//It should display an unordered list (bullet points).

//It should contain 3 list elements.

Steps:
1.Duplicate the template and change name : 03-intro-to-JSX
2. Open in CS code and run the commands
npm install
npm start
3. Write your code in index.js



JSX CODE TASK SOLUTION



Lesson : Java Dev - Day 5
Topic :
js expressions
Templete literals
Styling React Elements

techproeducation

techproeducation

techproeducation

techproeducation

REACT STARTING FILE

Use the template :04-template-literals

Open in vs code

In terminal run these scripts:

npm install -> install nod_modules

npm start -> open on local host

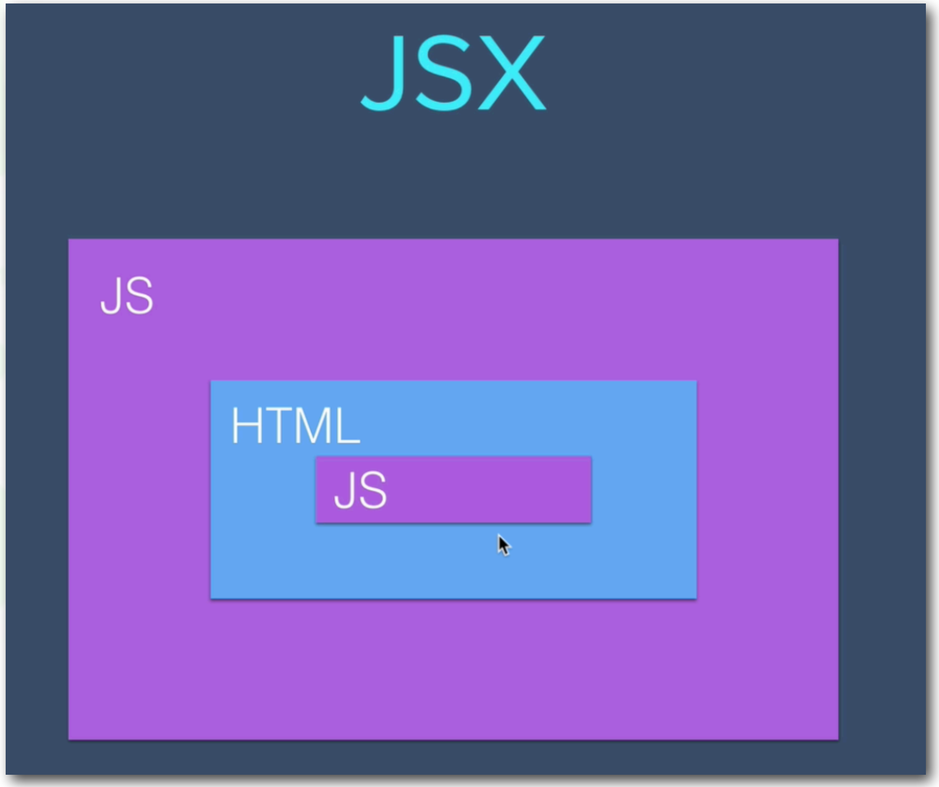
ERROR Then RUN: npm install react-scripts start

RUN npm start

Then you should see a blank page

JAVASCRIPT EXPRESSIONS IN JSX & ES6 TEMPLETE LITERALS

We inserted HTML inside a Javascript file.
We can do more than that
We can insert js in html that is in js
JSX lets us insert javascript in an html inside a javascript



But how do we render a variable on the page?????

We use **curly braces { }** if we want to insert a javascript code inside an html inside the javascript file

```
//create a variable, assign your name and display it
on the screen
const name ="Ahmet";

ReactDOM.render(<h1>Hello {name}</h1>,
document.getElementById("root"));
```

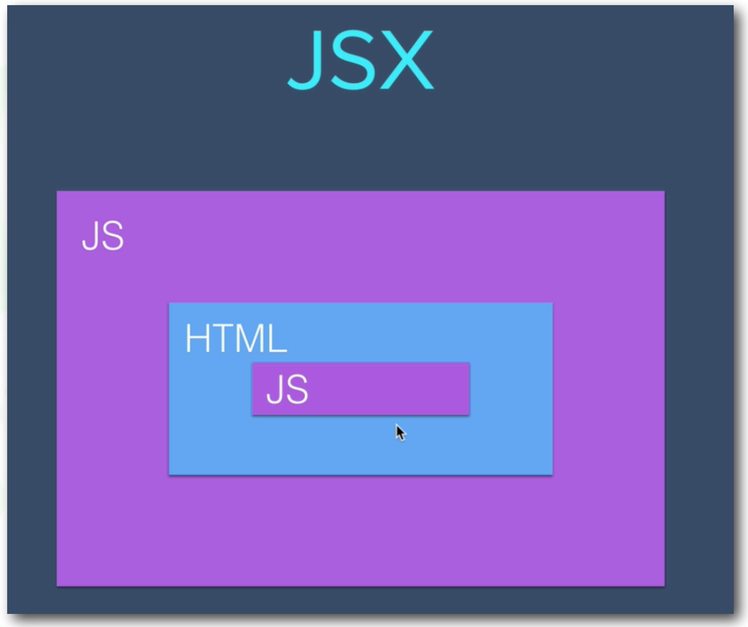
JAVASCRIPT EXPRESSIONS IN JSX & ES6 TEMPLETE LITERALS

YOUR TURN:

Create a const myFavNum, assign a value and render on the screen

Render your favorite number on the page in a **paragraph** and print: ‘My favorite number is 11’

Get 11 as a js variable



SOLUTION

YOUR TURN:

Render your favorite number on the page in a paragraph and print: ‘My favorite number is 11’

Create a const myFavNum, assign a value and render on the screen:


```
const myName = "Ahmet"
const myFavNum = 11

ReactDOM.render(
    <div>
        <h1>My To Do List</h1>
        <ul>
            <li>Go Shopping</li>
            <li>Do Exercise</li>
            <li>Reserve Hotel</li>
        </ul>
        <h1>JS Obsects on the UI</h1>
        <p>Hi {myName}</p>
        <p>My favourite number is {myFavNum}</p>
        <h1>JS Function on the UI</h1>
        <p>My Random Number {Math.floor(Math.random()*10)}</p>
    </div>,
    document.getElementById("root")
)
```

My To Do List

- Go Shopping
- Do Exercise
- Reserve Hotel

JS Obsects on the UI

Hi Ahmet

My favourite number is 11

JS Function on the UI

My Random Number 9

ES6 TEMPLETE LITERALS

```
const name ="Ahmet";
const lName="Bayram";
```

{} => Use to pass JS object inside html element.
All 3 codes below displays same outcome

1. Space can be placed between two js objects

```
<h1>Hello {name} {lName}</h1>
```

2. Passing js object insite html element using {}. " " is used for space

```
<h1>Hello {name + " " + lName}</h1>
```

3. We can inject a string into a javascript in ES6 with back-tick with multiple ways

`` `${name} ${lName}` `` => string using template literals

`` {`${name} ${lName}`} `` => string inside a javascript using JSX

`` <h1>{Hello `${name} ${lName}`}</h1> `` => then that h1 is inserted inside the javascript code in the javascript file(index.js) to display on the page

```
<h1>{Hello `${name} ${lName}`} </h1>
```

JAVASCRIPT EXPRESSIONS IN JSX & ES6 TEMPLETE LITERALS PRACTICE

```
//PRACTICE
//The paragraphs should say:
//Copyright CURRENTYEAR(js object).
//Example:
//Created by Ahmet Bayram.
//Copyright ©2022.
```

Styling REACT elements

JSX VS HTML STYLING

JSX differences

Class -> className

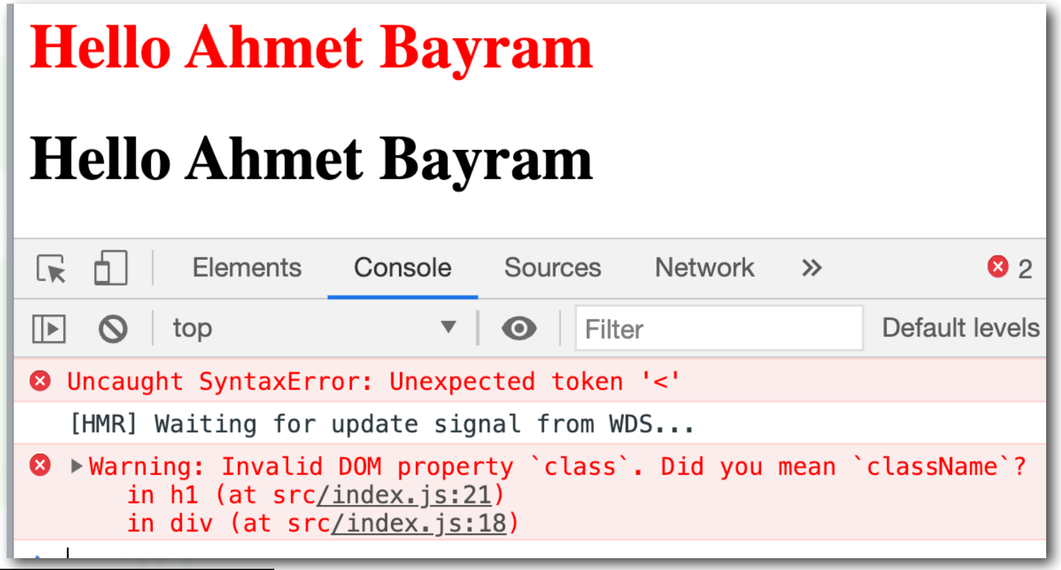
for -> htmlFor

camelCase property naming convention

- onclick -> onClick
- tabindex -> tabIndex

JSX ATTRIBUTES & STYLING REACT ELEMENTS

TASK:
Change the color of the headings to red
Go to styles.css and add class style for heading to change the color to red.
We will target heading to make it red



We need to use JS Styles not HTML style because h1 is rendered down to javascript!!!
THEY ARE NOT HTML THEY ARE JSX ELEMENTS

Normal html attribute is not camel case. They are kebab-case. font-size:"50px"

```
<h1 class="heading">Hello {name}</h1> NOT CORRECT FOR REACT
```

In JSX file, we need to use camelCalse. fontSize

```
<h1 className="heading">Hello {name}</h1> CORRECT FOR REACT
```

So use className for styling Reach Elements

JSX ATTRIBUTES & STYLING REACT ELEMENTS

TASK: Make the heading editable

TASK: Make the heading not editable

TASK: Ignore the spell check:

USE INLINE STYLING

SOLUTION

TASK: Make the heading editable

TASK: Make the heading not editable

TASK: Ignore the spell check:

```
<div>
    <h1
    className="heading-style"
    contentEditable="true"
    spellCheck="false">My To Do List</h1>
    <ul>
```

TASK

Make all p tag

color blue

Font size 2 rem

Way 1: using p tag

Way 2: className

```jsx
<p className='my-parag'>My fav num {myFavNum}</p>
<p className='my-parag'>My fav num 3+4</p>
<p className='my-parag'>My fav num {3+4}</p>
```

```css
.my-parag{
    color: blue;
    font-size: 2rem;
}
```

HTML STANDART ATTRIBUTES

https://www.w3schools.com/tags/ref_standardattributes.asp

JSX ATTRIBUTES & STYLING REACT ELEMENTS

TASK 1: Add an image and style it

https://cdn.mos.cms.futurecdn.net/ntFmJUZ8tw3ULD3tkBaAtf.jpg

TASK 2: Size the image by 100px by 100px using css style

https://upload.wikimedia.org/wikipedia/commons/thumb/5/5b/Lula_kebab_2.jpg/2880px-Lula_kebab_2.jpg

TASK 3: Add two more image and style:

https://secure.img1-fg.wfcdn.com/im/08892826/resize-h800%5Ecompr-r85/4307/43073707/Playing+Tabby+Kitten+Statue.jpg

TASK 4:

Border radius : 20 px

Margin between the images by 10px

2px Border color solid black

JSX ATTRIBUTES & STYLING REACT ELEMENTS

We learned so far how to use js properties(className) as an attributes in our JSX(camelCase for more than 1 word)

We can use javascript objects as the value of an attribute

TASK: 1. Add another image and style that image using some ready features :

https://picsum.photos/

Blurred :

```
<img src="https://picsum.photos/id/237/200/300"></img>
<img src="https://picsum.photos/id/237/200/300?blur"></img>
```

We can pass javascript objects AS A VALUE OF AN ATTRIBUTE

2. Create variable for image

```
const dogImg="https://picsum.photos/id/237/200/300?grayscale"
```

3. Use that image in the JSX as javascript object

4. Gray out the image using this website

```
<img
 className="dog-img"
 src={dogImg}></img>
</div>
```

Lesson : Java Dev - Day 5 Part 2
Topic :
js expressions
Templete literals
Styling React Elements

techproeducation

techproeducation

techproeducation

techproeducation

INLINE STYLING FOR REACT ELEMENTS

Last we added an image object as a value of src attribute
We will create a style as **javascript object** and pass as a **value of style attribute**
https://www.w3schools.com/react/react_css.asp

HOW WOULD YOU DO INLINE STYLING IN HTML? Css Style: font-size

```
<h1 style="color: orange">Hello {name + " " + lName}</h1>    This won't work for styling react
```

HOW WOULD YOU DO INLINE STYLING IN REACT? JS style: fontSize

STEPS for inline styling:

1. Create a js object:

```
{color:"orange"}
```

2. Add is inside any element you want using the style attribute as javascript object:

```
<h1 style={{color:"orange"}}>Hello {name + " " + lName}</h1>
```

We are using another {} because that is how we insert a javascript object in JSX

3. We can create reusable js object and use as a value of attribute as well

```
const redColor={color:"red"}
```

```
<h3 style={redColor}>{fName+' '+lName}</h3>
```

INLINE STYLING FOR REACT ELEMENTS

=>We can create custom style as js object and use that object for styling

Create a customStyle object

-color:red,fontSize:20px,border:2px solid blue

```
const customStyle={
    color: "red", fontSize:"20px", border: "2px solid blue"};
Adding the custom style to the element
<h1 style={customStyle}>Hello {name + " " + lName}</h1>
```

BENEFIT OF THIS THIS USAGE IS IF SOMETHING IS CHANGES,AND WANT TO CHANGE THE STYLE WE CAN JUST UPDATE SPECIFIC OBJECT AND USE IN THE ELEMENT.

THIS IS ONE OF THE REASON USING INLINE STYLING. ELEMENT CAN BE UPDATED ON THE GO FOR EXAMPLE CHANGE COLOR TO GREEN

```
customStyle.color="green";
```

```
//  Create a customStyle object
//  -color:red,fontSize:20px,border:2px solid blue
const customStyle={
    color:"orange",
    fontSize:"20px",
    border:"2px solid blue"
}
console.log(customStyle.color);//orange
// Benefit of using js object for styling elements is
// we can edit the attributes when needs
//For example, I can change ONLY the color bby edittin
customStyle.color="green"
console.log(customStyle.color);//green
customStyle.fontSize="30px"
```

```
<h3 style={customStyle}>{fName+' '+lName}</h3>
```

John Cash

REACT STYLING PRACTICE

1. Show a single h3 that says :
"Good morning" if between midnight and 12PM.
or "Good Afternoon" if between 12PM and 6PM.
or "Good evening" if between 6PM and midnight.

2. Apply the “time-style" style in the styles.css for the element
Font size=50 px, font weight= Bold, Border-buttom= 5px solid yellow
Dynamically change the color of the h3 using inline css styles.
Morning = red, Afternoon = green, Night = blue.

HINT:
Create Date() object
Get the hour as a JS object and store as currentTime
Create an object called greeting
Create a customStyle object and initialize the color
Use if else statement to assign values for greeting and costomStyle color based on currentTime
Render the h1 and use the customStyle and greeting where needed.

SOLUTION

```
//Creating greeting object to render on the
let greeting;
//Getting the current hour
// const fullDate=new Date(2021,1,1,19)
const fullDate=new Date()
//getHours() returns the hour only
const currentTime = fullDate.getHours()
console.log(currentTime)
//creating a color object that will be used
const customColor={
    color:""
}
if(currentTime<12){
    greeting="Good morning"
    customColor.color='red'
}else if(currentTime<18){
    greeting="Good afternoon"
    customColor.color='green'
}else{
    greeting="Good night"
    customColor.color='blue'
}


ReactDOM.render(
```

```
.time-style{
    font-size: 15px;
    font-weight: bold;
    border-bottom: 5px solid yellow;
    font-style: italic;
}
```

```
<h3
style={customColor}
className="time-style">{greeting}</h3>
```

Good afternoon

REACT COMPONENTS

REACT COMPONENT

When a file gets too big, we use components

Components is used to split up large or complex files into small parts.

We can reuse smaller component in bigger parts as well

We will split up the index.js into smaller components

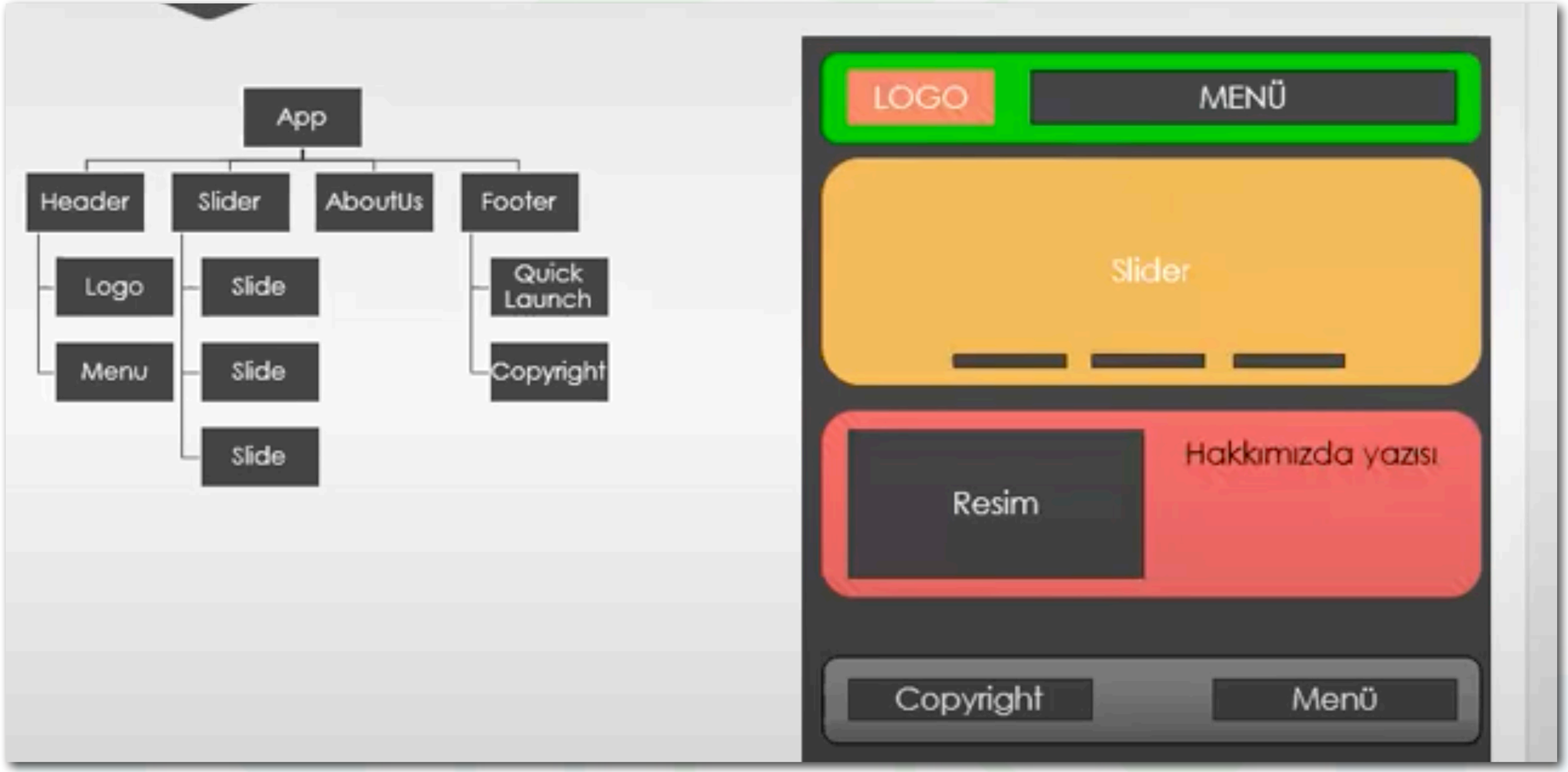
https://github.com/airbnb/javascript/tree/master/react has the information about best practices and conventions

Heading Component

JS index.js > ...

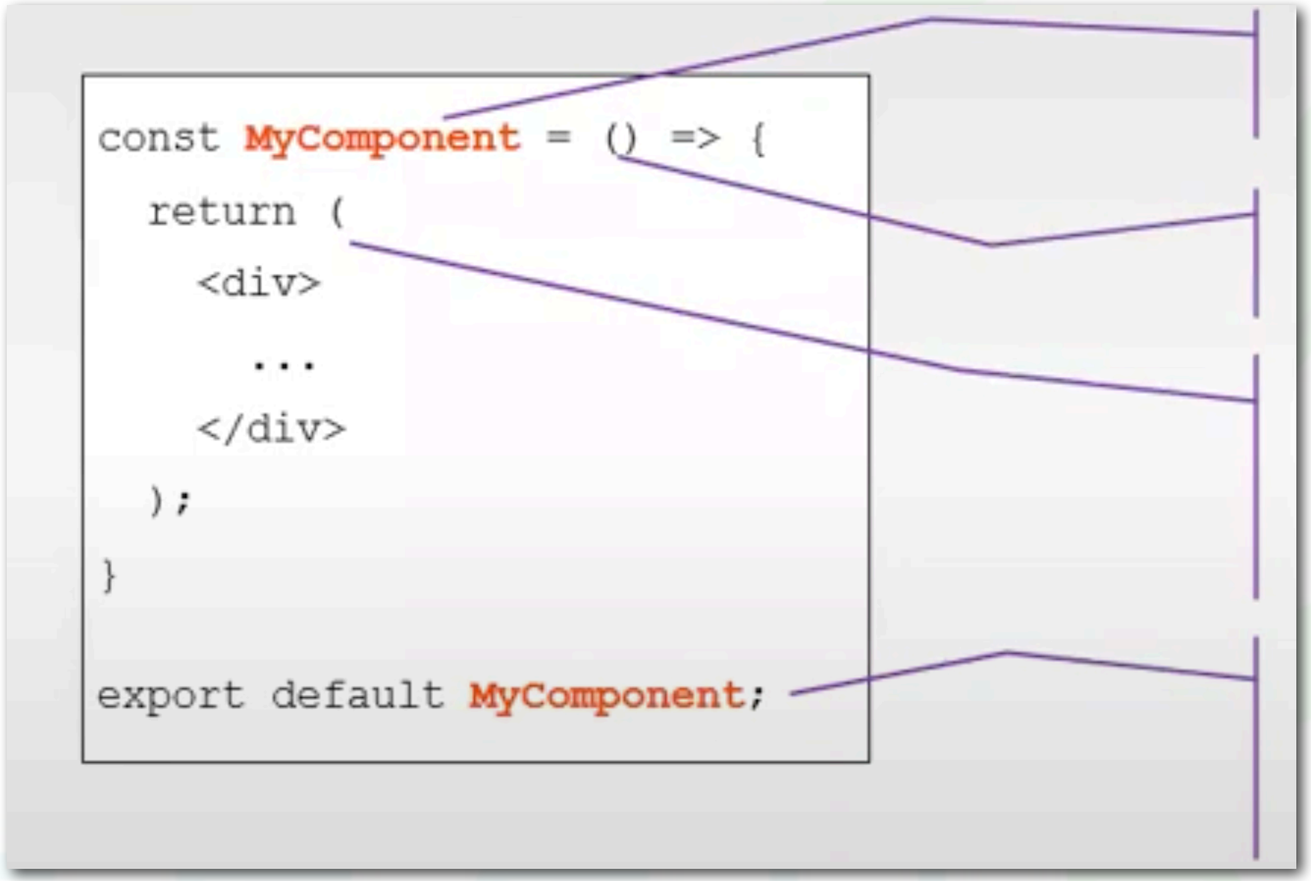
```
import React from 'react';
import ReactDOM from 'react-dom';

//Creating a Heading component
function Heading(){
return <h1>Hello World</h1>
}

ReactDOM.render(
  <div>
    <Heading />
    <ul>
      <li>Go Shopping</li>
      <li>Do Exercise</li>
      <li>Reserve HoTel</li>
    </ul>
  </div>
,
  document.getElementById('root')
);
```

COMPONENT EXAMPLE



SAMPLE COMPONENT

COMPONENTS ARE SIILAR TO FUNCTIONS



PASCAL CASE

ARROW FOR NORMAL FUNCTION CAN BE USED
PARAMETER CAN BE PASSED HERE

CONTENT OF THE COMPONENT IS WRITTEN INSIDE THE RETURN
REACT RETURN ONLY ONE SINGLE ELEMENT SO USE DIV TAG

EXPORT THE COMPONENT SO IT CAN BE IMPORTED FROM OTHER COMPONENTS

REACT STARTING FILE

Use the template : 05-react-components

Open in vs code

In terminal run these scripts:

npm install -> install nod_modules

npm start -> open on local host

ERROR Then RUN: npm install react-scripts start

RUN npm start

Then you should see a blank page

REACT COMPONENT

1. Create a new react application that renders h1 heading and ul list
2. Split the heading component inside the index.js first. To create component we create a function outside of ReactDOM.render.

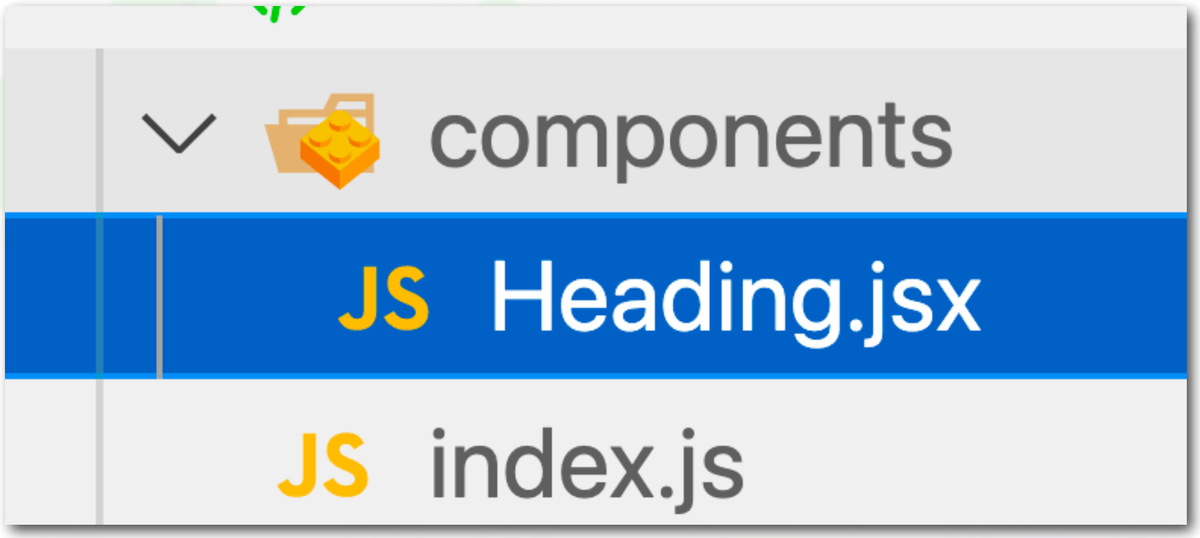
JS index.js > ...

```
import React from 'react';
import ReactDOM from 'react-dom';

//Creating a Heading component
function Heading(){
return <h1>Hello World</h1>
}

ReactDOM.render(
  <div>
    <Heading />
    <ul>
      <li>Go Shopping</li>
      <li>Do Exercise</li>
      <li>Reserve HoTel</li>
    </ul>
  </div>
,
  document.getElementById('root')
);
```

REACT COMPONENT- COMPONENTS FOLDER

1. Normally we should create a components folder and move all components in that folder
2. Separate the Heading component as a separate jsx file.
    1. Create a components folder under src
    2. Create Heading component under components folder=Heading.jsx
3. Put the custom Heading component inside ReactDOM.render in index.js.

`Steps are in the next slide`



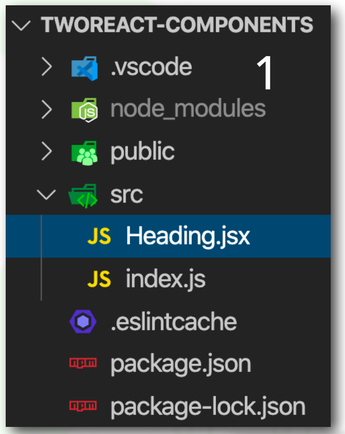
REACT COMPONENT

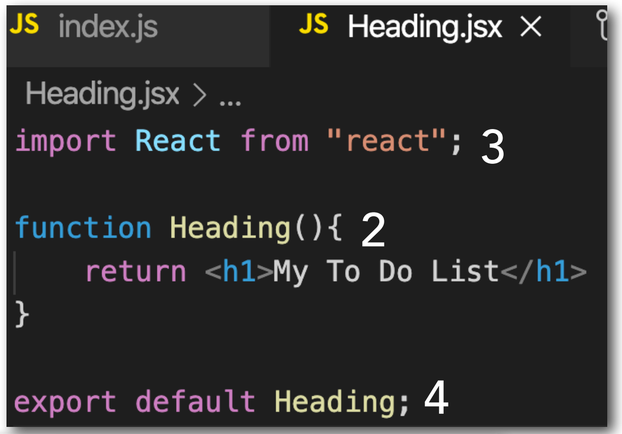
We will use ES6 feature to make our code better.

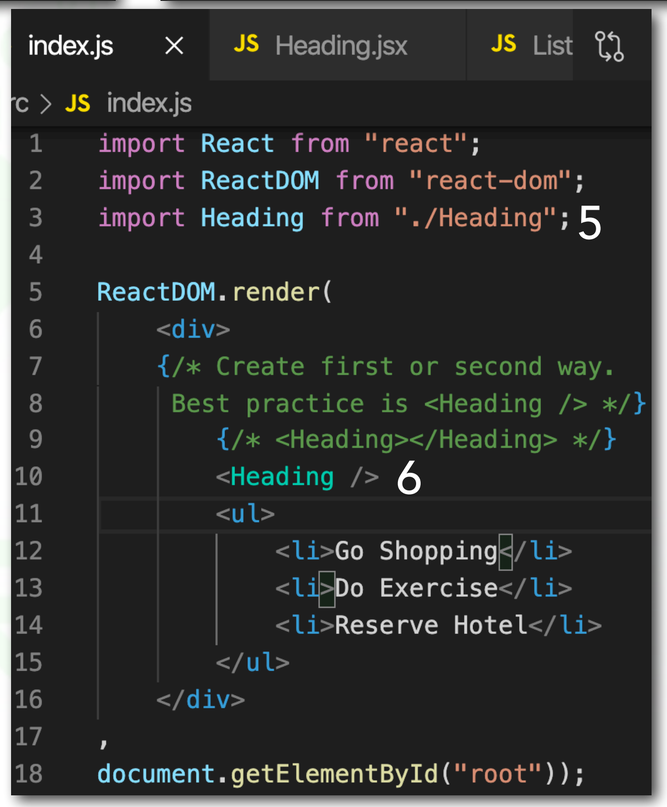
We should move the smaller components in a different file

We will create a new file for heading component and put everything about heading in that component

Then call the component in the index.js







TASK:

1. Create a file: **Heading.jsx**
2. Cut and paste the Heading function from index.js to the Heading component.
3. Import the React from “react”; in the Heading component
4. Export the Heading jsx so we can use it in other classes.
5. Inside index.jsx Import Heading component
6. And use the reusable Heading component in the index.js

REACT COMPONENT PRACTICE

TASK:

Create a component called List

That will have ul elements in it

And use that component in the index.js

REACT COMPONENT-PRACTICE SOLUTION

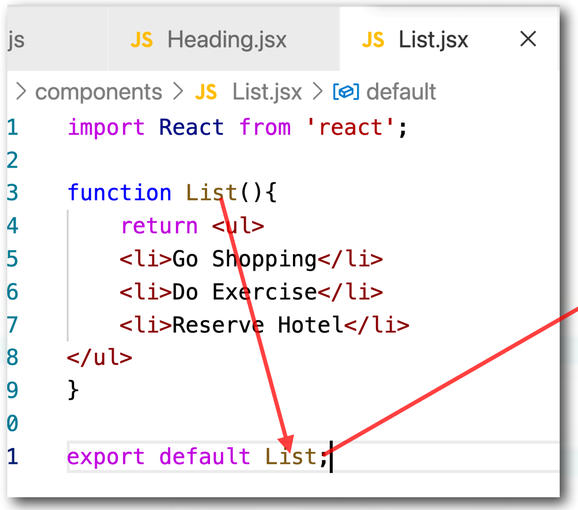
Crate List.jsx component

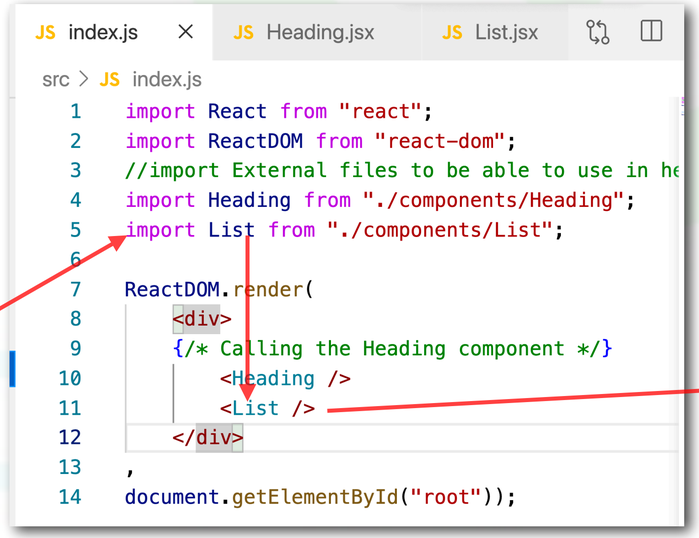
Cut and paste ul element in the List.jsx

Export List.jsx

Import List.jsx in the index.js

Use <List /> in the ReachDOM.render





Lesson : Java Dev - Day 6
Topic :
Components
Props

techproeducation

techproeducation

techproeducation

techproeducation

WHAT DO YOU REMEMBER FROM THE LAST CLASS?

- COMPONENTS:
  - React is components based javascript front-end development library
  - Components are used to create smaller parts of an application
    - Header component, List component, Search component, Image component, Info component, Footer components,…
    - We use an App component to render in the UI.
    - All components must be connected to the index.js
- STYLING REACT ELEMENTS
  - Styling react elements are similar to styling html elements, but there are syntax differences
    - camalCase (fontSize, fontWeight,..) instead of kebab-case(font-size,font-weight)
    - className instead of class attribute
    - {} can be used as a value of style attribute for inline styling <h1 style={{color:"orange"}}>Hello {name + " " + lName}</h1>
    -

```
const redColor={color:"red"}
```

```
<h3 style={redColor}>{fName+' '+lName}</h3>
```

- CUSTOM STYLE
  - const customStyle={color: "red", fontSize:"20px", border: "2px solid blue"};
- Using the custom style
  - <h1 style={customStyle}>Hello {name + " " + lName}</h1>
- We can override the property of the elements
  - customStyle.color="green";

One advantage of creating inline styling is that we can individually style the elements

We can override the values for conditional styling on the go

NOTE: We can use both external style(className) and internal style(style) for the same element. In this case both will apply but INTERNAL STYLE WILL OVERRIDE the EXTERNAL STYLE, if there is a match in the properties.

If there is no matching property, then both will apply.

- USING JS IN THE JSX
  - {} is used to pass a javascript object in the JSX(html)

REACT COMPONENT- <APP /> COMPONENT

Normally index.js doesn't have any custom component like <Heading /> or <List /> we just did

We create an **<App /> component** and place It in the index.js

1. Create App.jsx component
2. Crete App() function and return <Heading /> and <List /> components in a div
3. Import react, and the components you want to render(Heading and List)
4. Export App class: **export default App;**
5. Go to index.js and import, Delete Other components and add App class import App from "./App"
6. Render only <App /> component in ReactDOM.render

REACT COMPONENT- <APP /> COMPONENT

Solution

App.jsx

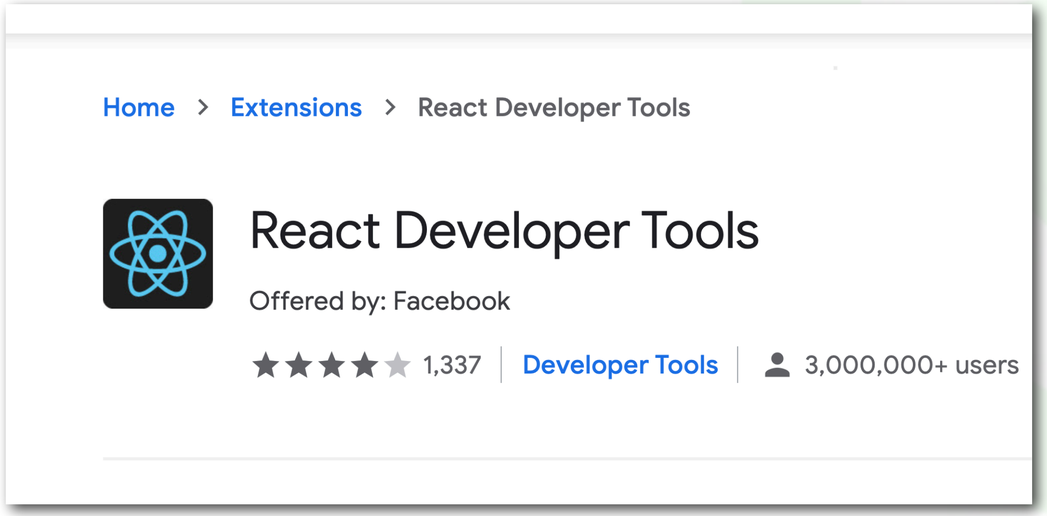
```
src > components > JS App.jsx > [] default
1   import React from "react";
2   import Heading from "./Heading";
3   import List from "./List";
4
5   function App(){
6       return <div>
7                   <Heading />
8                   <List />
9               </div>
10  }
11
12  export default App;
```

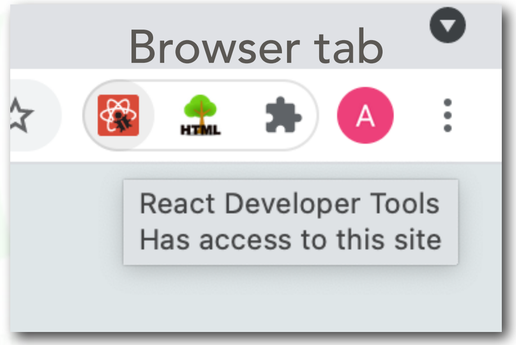
index.js

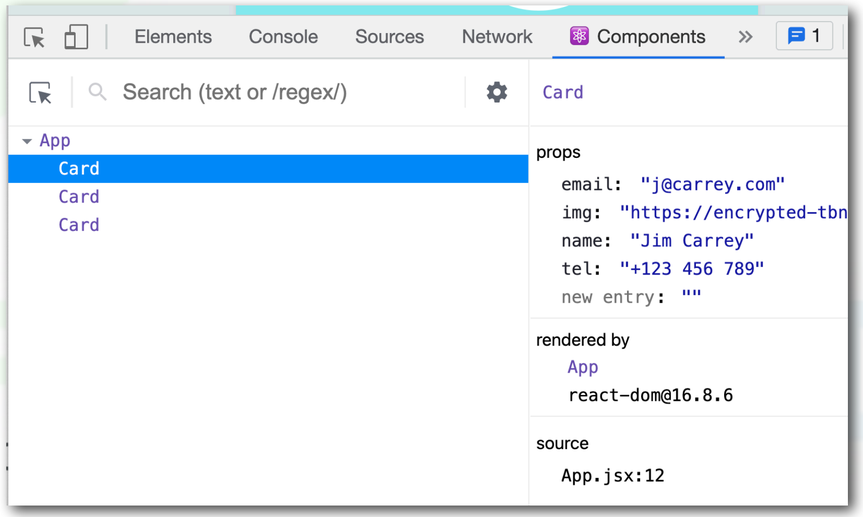
```
src > JS index.js
1   import React from 'react';
2   import ReactDOM from 'react-dom';
3   import App from "./components/App";
4   ReactDOM.render(
5     <div>
6       <App />
7     </div>
8   ,
9     document.getElementById('root')
10  );
11
12
```

INSTALL REACT DEV TOOL

1. Search for react developer tools on chrome
2. Install the extension







React Dev tool help us see the the component tree

Props, hooks, etc.

REACT COMPONENTS PRACTICE

Delete node_modules

Duplicate 04-template-literals

Rename : 06-react-component-practice

Open in VS code

Npm install

Npm start

REACT COMPONENTS PRACTICE

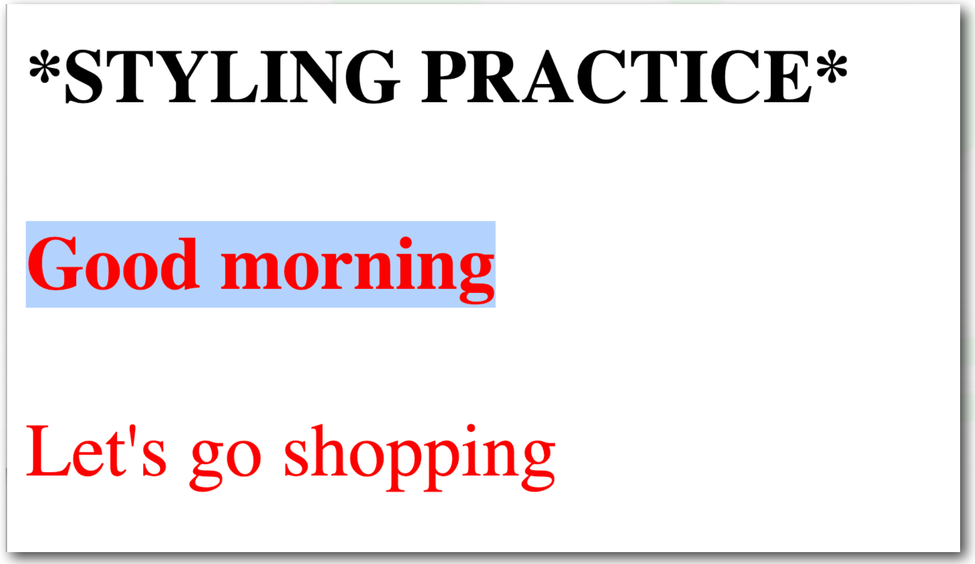
TASK 1:

Create a Greeting.jsx component

And render

```
<h1>*STYLING PRACTICE*</h1>
<h4 style={customStyle}>{greeting}</h4>
<p style={customStyle}>Let's go shopping</p>
```

At the end we should see the same UI



Greeting.jsx — 06-react-component-practice

JS Greeting.jsx U

src > components > JS Greeting.jsx > Greeting

```jsx
import React from "react";

let currentDate = new Date();
let greeting;
let customStyle={
  color:"",
  fontSize:""
};
const currentHour= currentDate.getHours();
if( currentHour < 12 ){
  greeting ='Good morning';
  customStyle.color='red';
  customStyle.fontSize='2rem'
}else if(currentHour < 18 ){
  greeting='Good afternoon';
  customStyle.color='green';
  customStyle.fontSize='3rem'
}else{
  greeting='Good night';
  customStyle.color='blue'
  customStyle.fontSize='4rem'
}
function Greeting(){
    return <div>
        <hr/>
      <h1>*STYLING PRACTICE*</h1>
      <h4 style={customStyle}>{greeting}</h4>
      <p style={customStyle}>Let's go shopping<
    </div>
}
export default Greeting;
```

JS index.js U

src > JS index.js > ...

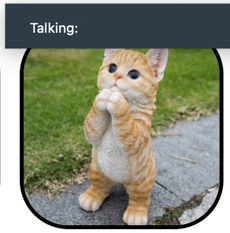
```jsx
import React from 'react';
import ReactDOM from 'react-dom';
import Greeting from './components/Greeting';

        alt='kebab-img'/>
        <img
        src='https://secure.img1-
        className='cat-img'
        alt='cat-img'/>
        <img
        src='https://picsum.photo
        className='random-img'
        alt='random-img'/>
        <img
        src={randomImage +'?grays
        className='random-img'
        alt='random-img'/>

        <Greeting />

    </div>,
    document.getElementById('root
);
```

React App

localhost:3000

EXTERNAL STYLING






STYLING PRACTICE

Good morning

Let's go shopping

REACT COMPONENTS PRACTICE

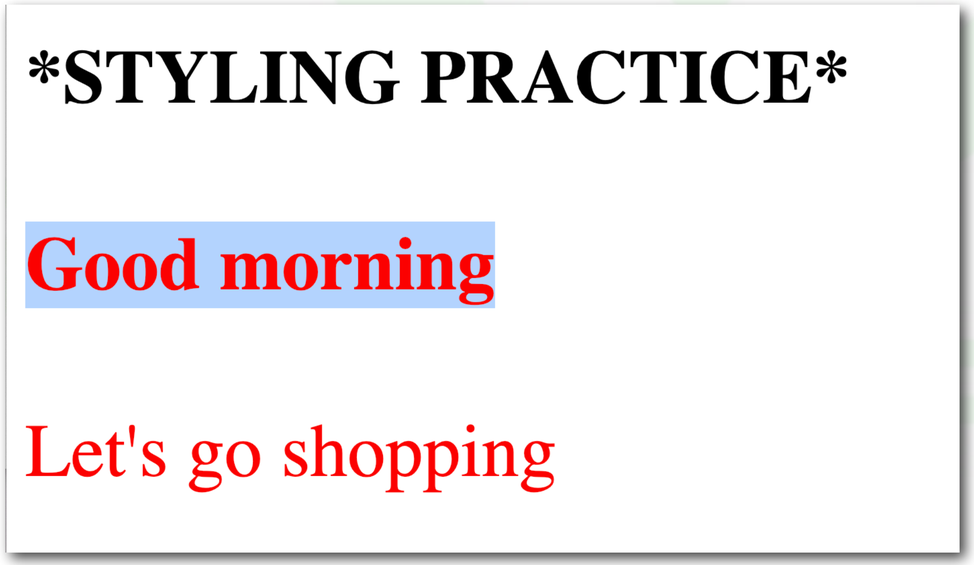
TASK 2:

Create an App.jsx component

And render Greeting component

And use the App inside the index.js

At the end we should see the same UI



JS Greeting.jsx U

src > components > JS Greeting.jsx > default

```jsx
import React from "react";

let currentDate = new Date();
let greeting;
let customStyle={
  color:"",
  fontSize:""
};
const currentHour= currentDate
if( currentHour < 12 ){
  greeting ='Good morning';
  customStyle.color='red';
  customStyle.fontSize='2rem'
}else if(currentHour < 18 ){
  greeting='Good afternoon';
  customStyle.color='green';
  customStyle.fontSize='3rem'
}else{
  greeting='Good night';
  customStyle.color='blue'
  customStyle.fontSize='4rem'
}
function Greeting(){
    return <div>
        <hr/>
      <h1>*STYLING PRACTICE*</
      <h4 style={customStyle}>
      <p style={customStyle}>L
    </div>
}
export default Greeting;
```

JS App.jsx U

src > components > JS App.jsx > default

```jsx
import React from "react";
import Greeting from "./Greeti

function App(){
    return <div>
        <Greeting />
    </div>
}

export default App;
```

JS index.js U

src > JS index.js > ...

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

            className='random
            alt='random-img'/

            {/* <Greeting />
            <App />

    </div>,
    document.getElementBy
);
```

Talking:

React App

localhost:3000

1. Hello Ahmet Bayram

2. Hello Ahmet Bayram

3. Hello Ahmet Bayram

3. Hello Ahmet Bayram

Copyright ©2022

*EXTERNAL STYLI

*STYLING PRACTI

Good morning

Let's go shopping

IMPORT, EXPORT AND MODULES

HOW TO USE A SINGLE ITEM FROM OTHER JS FILES:

TASK: Create math.js file in components folder and const pi, and display it on the first li on the screen:

Add unordered list in the index.js module:

```
<div>
    <App />
    <ul>
      <li>num1</li>
      <li>num2</li>
      <li>num3</li>
    </ul>
  </div>
```

In the index.js:

1. Import pi from math class
2. Render the item on the UI.

IMPORT, EXPORT AND MODULES

What is a have more that one item to How I use them in other js file?

In math.js, create function **doublePi()**, and function **triplePi()** functions

//Create doublePi function and return 2 times pi

//Create triplePi function and return 3 times pi

//export pi as default. That is already there and export the other two functions.
When we export multiple item we use {} !!!!!!

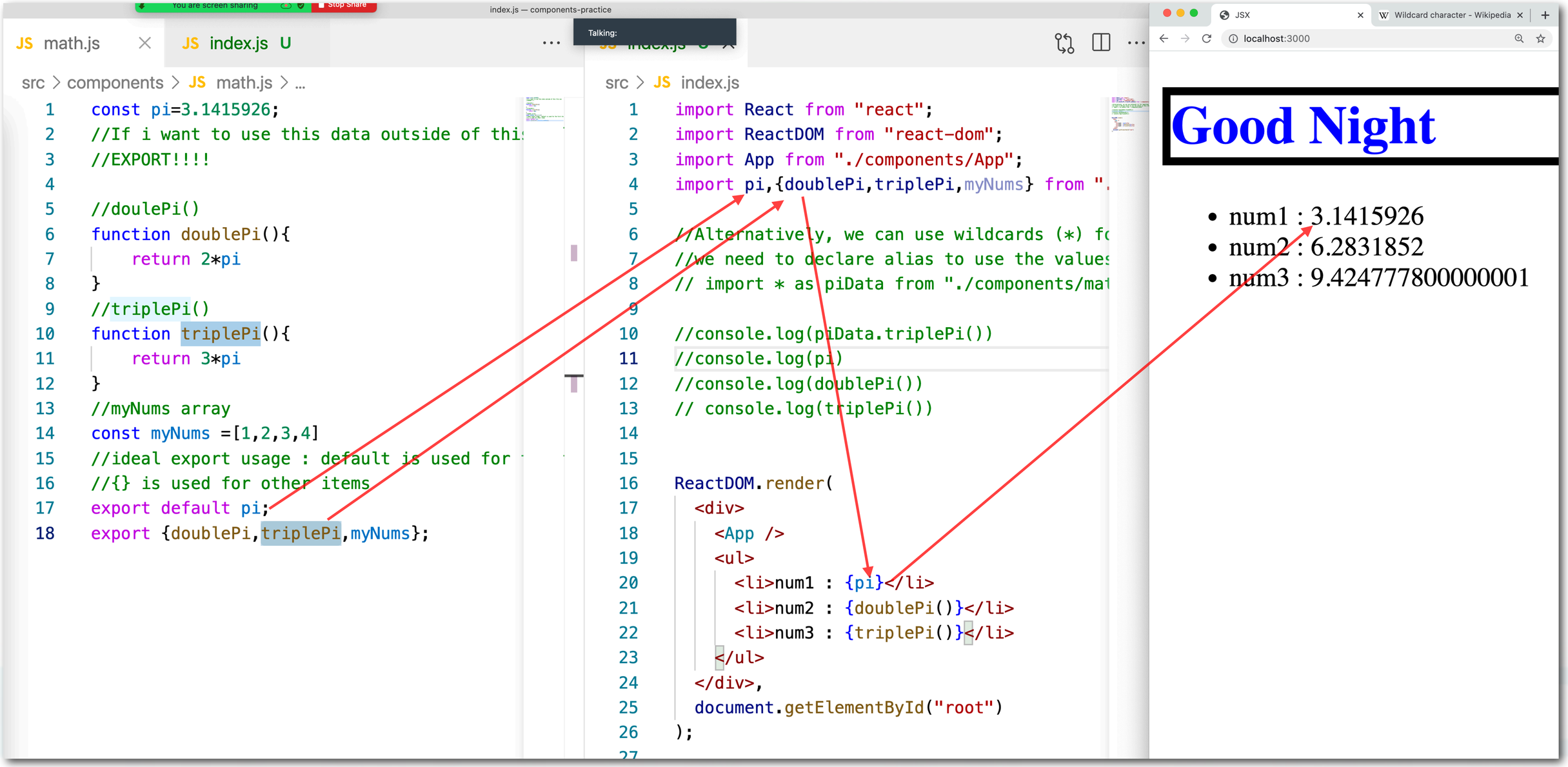
//import the functions in index.js. Make sure to use the names {}

//Now we can use the pi and the functions in the render method:

So we can split up large files into smaller manageable components, and use inside each other using import and export keyword

- num1 : 3.141596
- num2 : 6.283192
- num3 : 9.424788

IMPORT, EXPORT AND MODULES SOLUTION




```
const pi=3.1415926;
//If i want to use this data outside of this
//EXPORT!!!!

//doulePi()
function doublePi(){
    return 2*pi
}
//triplePi()
function triplePi(){
    return 3*pi
}
//myNums array
const myNums =[1,2,3,4]
//ideal export usage : default is used for
//{} is used for other items
export default pi;
export {doublePi,triplePi,myNums};
```

```
import React from "react";
import ReactDOM from "react-dom";
import App from "./components/App";
import pi,{doublePi,triplePi,myNums} from "
//Alternatively, we can use wildcards (*) fo
//we need to declare alias to use the values
// import * as piData from "./components/mat

//console.log(piData.triplePi())
//console.log(pi)
//console.log(doublePi())
// console.log(triplePi())


ReactDOM.render(
  <div>
    <App />
    <ul>
      <li>num1 : {pi}</li>
      <li>num2 : {doublePi()}</li>
      <li>num3 : {triplePi()}</li>
    </ul>
  </div>,
  document.getElementById("root")
);
```

IMPORT, EXPORT AND MODULES

I can use wildcard to import everything form math.js

This is not recommended

```
// import pi,{doublePi,triplePi} from "./math.js";
// we can import everything as pi
import * as pi from "./math.js";
// pi has everything from math.js module
// Now we can get the elemetns in the pi :
// Inside the li elements I can use:
// pi.default
// pi.doublePi();
// pi.triplePi();
// Using wildcards are discouradged
```

```
<li>num1 : {pi.default}</li>
{/* when we pass functions use () */}
<li>num2 : {pi.doublePi()}</li>
<li>num3 : {pi.triplePi()}</li>
```

KEEPER APP PROJECT 1

Import keeper-app-part1-starting , npm install react-scripts start, npm install, npm start

OR use the last one you created, add google fonts and css link in the head in index.html

```
<link href="https://fonts.googleapis.com/css?family=McLaren|Montserrat&display=swap" rel="stylesheet"/>
<link rel="stylesheet" href="styles.css" />
```

AND create styles.css in public add the the classes that is on the right

HERE IS WHAT WE WILL DO :

//1. Create a new React app. Create component folder under src

//2. Create a App.jsx component.

//3. Create a Header.jsx component that renders a <header> element

//to show the Keeper App name in an <h1>.

//4. Create a Footer.jsx component that renders a <footer> element

//to show a copyright message in a <p> with a dynamically updated year.

//5. Create a Note.jsx component to show a element with a

//<h1> for a title and a <p> for the content.

//6. Make sure that the final website is styled like the example shown here:

//HINT: You will need to use the classes in the styles.css file to appy styling

index.html | styles.css | index.js

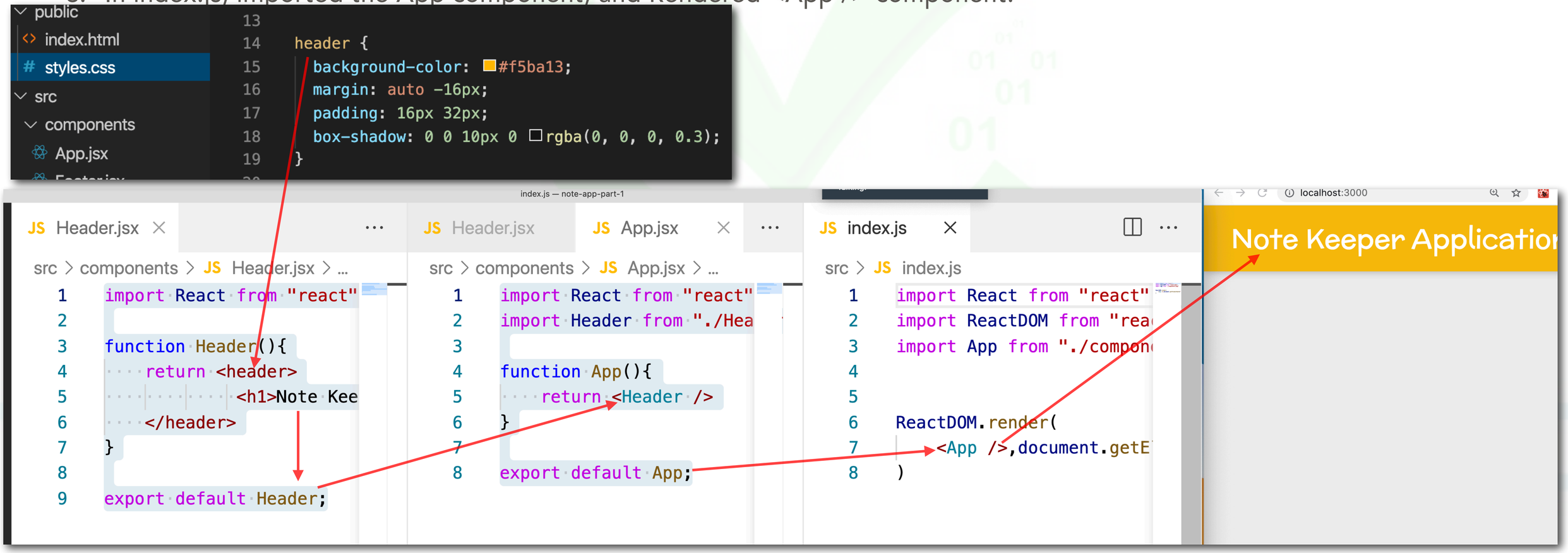
public > index.html > ...

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>React App</title>
    <link href="https://fonts.googleapis.com/css?family=Mcl
    <link rel="stylesheet" href="styles.css" />
  </head>
  <body>
    <div id="root"></div>
    <script src="./src/index.js" type="text/jsx"></script>
  </body>
</html>
```


```
* {
  padding: 0;
  margin: 0;
  box-sizing: border-box;
}
html {
  font-family: "Montserrat", sans-serif;
}
body {
  background: #eee;
  padding: 0 16px;
}

header {
  background-color: #f5ba13;
  margin: auto -16px;
  padding: 16px 32px;
  box-shadow: 0 0 10px 0 rgba(0, 0, 0, 0.3);
}

header h1 {
  color: #fff;
  font-family: "McLaren", cursive;
  font-weight: 200;
}

footer {
  position: absolute;
  text-align: center;
  bottom: 0;
  width: 100%;
  height: 2.5rem;
}

footer p {
  color: #ccc;
}
.note {
  background: #fff;
  border-radius: 7px;
  box-shadow: 0 2px 5px #ccc;
  padding: 10px;
  width: 240px;
  margin: 16px;
  float: left;
}
.note h1 {
  font-size: 1.1em;
  margin-bottom: 6px;
}
.note p {
  font-size: 1.1em;
  margin-bottom: 10px;
  white-space: pre-wrap;
  word-wrap: break-word;
}
```

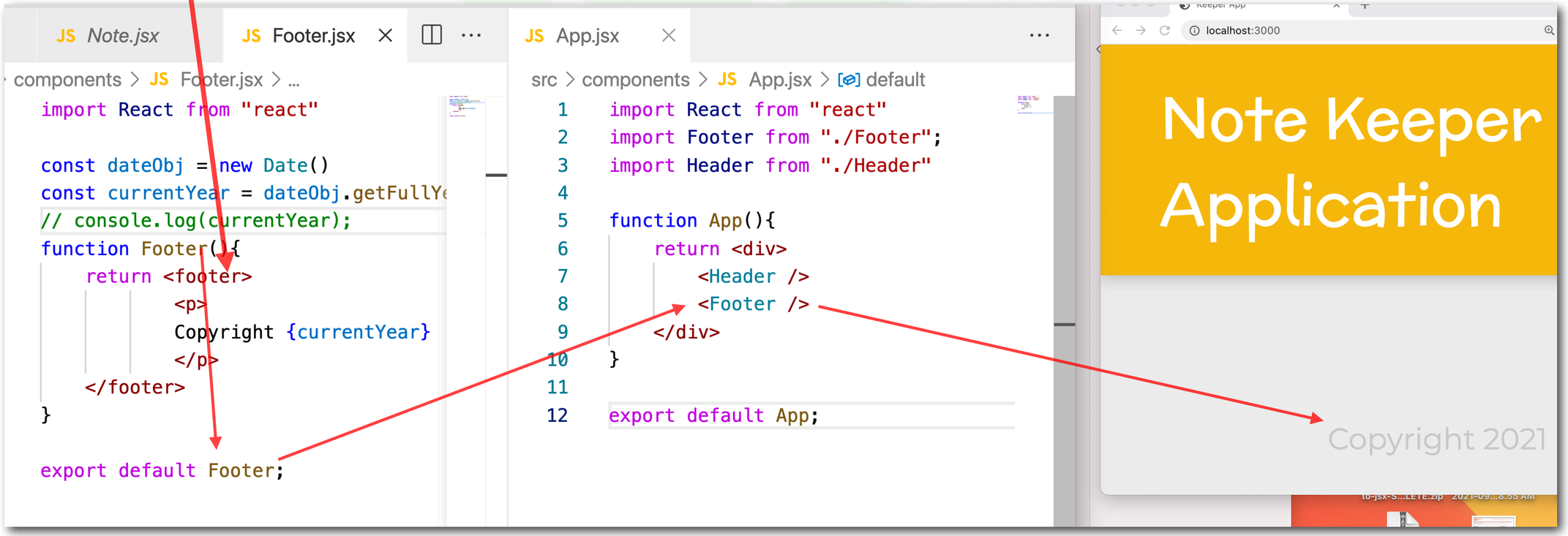
HEADER

1. Created Header component
    1. Returned <header /><h1>Note Keeper Application</h1></header>. Then exported Header
2. Created App component.
    1. Returned <Header />. Then exported App.
3. In index.js, imported the App component, and Rendered <App /> component.



FOOTER.JSX

```css
footer {
  position: absolute;
  text-align: center;
  bottom: 0;
  width: 100%;
  height: 2.5rem;
}
```



NOTE.JSX

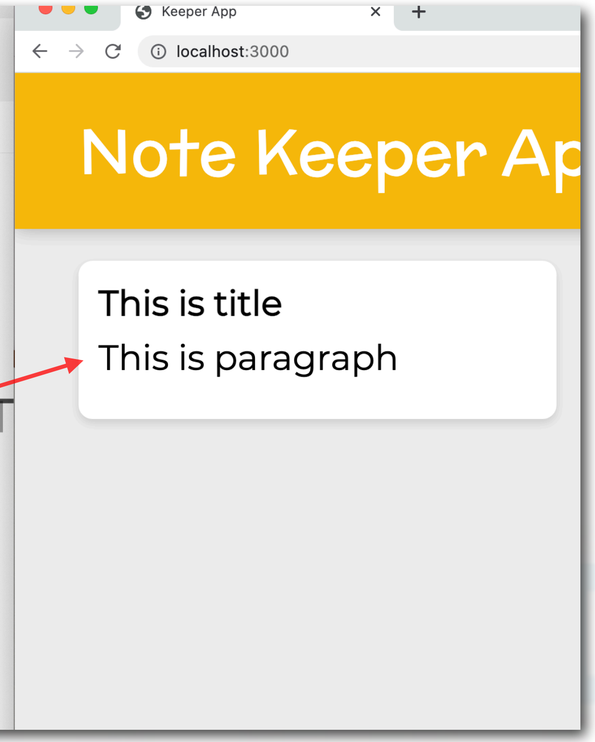
```css
.note {
  background: #fff;
  border-radius: 7px;
  box-shadow: 0 2px 5px #ccc;
  padding: 10px;
  width: 240px;
  margin: 16px;
  float: left;
}
```

Note.jsx — src > components > JS Note.jsx > Note

```jsx
import React from "react";

function Note(){
    return <div className="note">
        <h1>This is title</h1>
        <p>This is paragraph</p>
    </div>
}

export default Note;
```

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

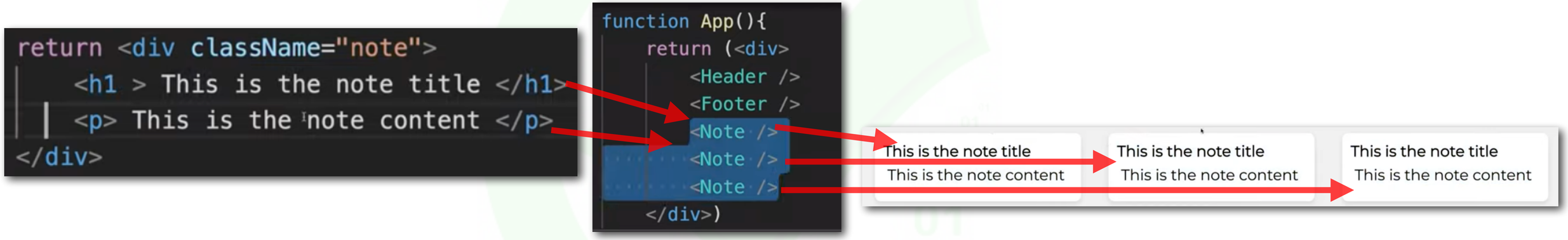
```jsx
import React from "react"
import Footer from "./Footer";
import Header from "./Header"
import Note from "./Note"
function App(){
    return <div>
        <Header />
        <Footer />
        <Note />
    </div>
}

export default App;
```



PROPS

So far we created reusable components. For example, I can use Note component

Notice we see 3 SAME note components. This means this reusability doesn't actually help



How can we CUSTOMIZE a component???

With props I can pass custom data for each component.

For example I want to show different names for each Note component

PROPS

How can we CUSTOMIZE a component???

With props I can pass custom data for each component.

For example I want to show different names for each Note component

Creating name props
for Note component

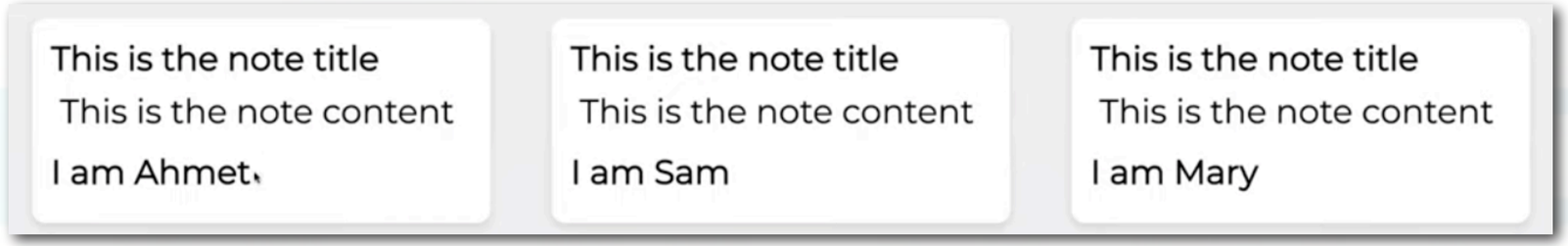
```
return (<div>
    <Header />
    <Footer />
    <Note name="Ahmet"/>
    <Note name="Sam"/>
    <Note name="Mary"/>
</div>)
```

Calling props inside
Note component

```
function Note(props){
    console.log(props);
    console.log(props.name);
```

Showing name props as JS object

```
console.log(props.lName);
return <div className="note">
    <h1> This is the note title </h1>
    <p> This is the note content </p>
    <h1> I am {props.name} </h1>
</div>
```

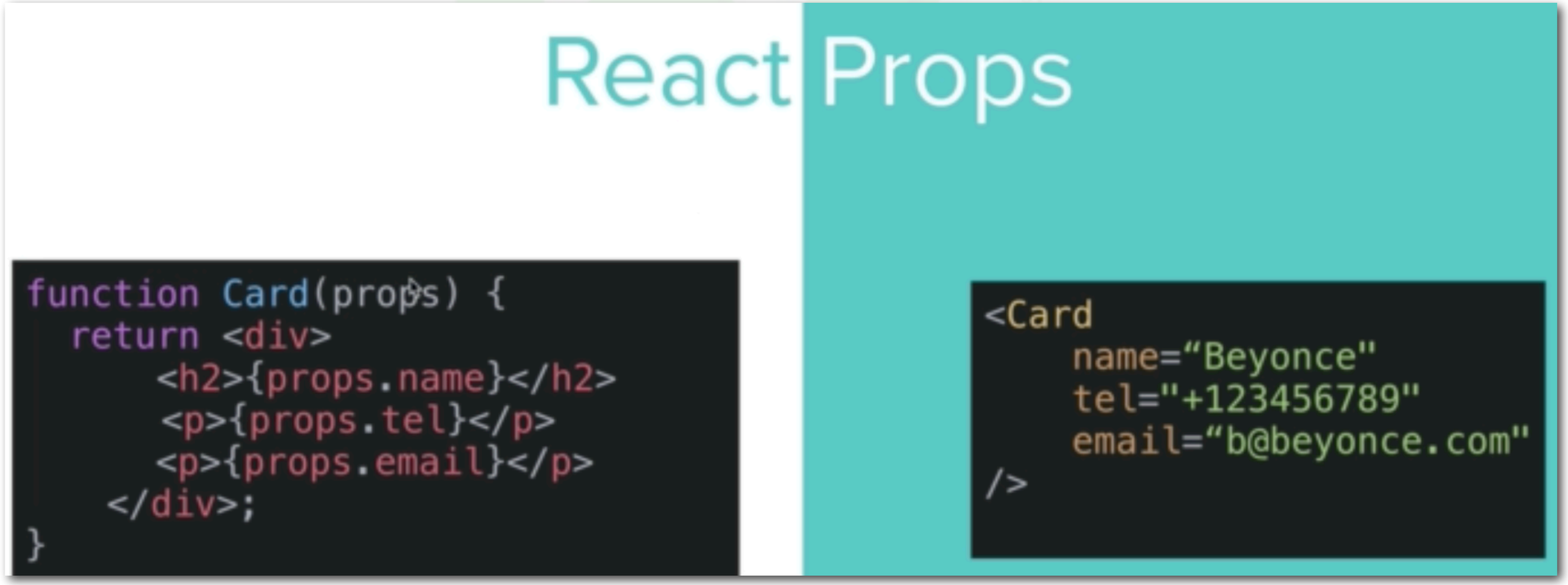


PROPS

Props means property in react.

Props is used to pass data from one component to another component

Props pass data from parent to child only



REACT PROPS - HTML TAG VS REACT PROPERTIES

Html tag :

Fixed attributes

Pass custom data to access

Then access using .id, .placehoder, .value properties

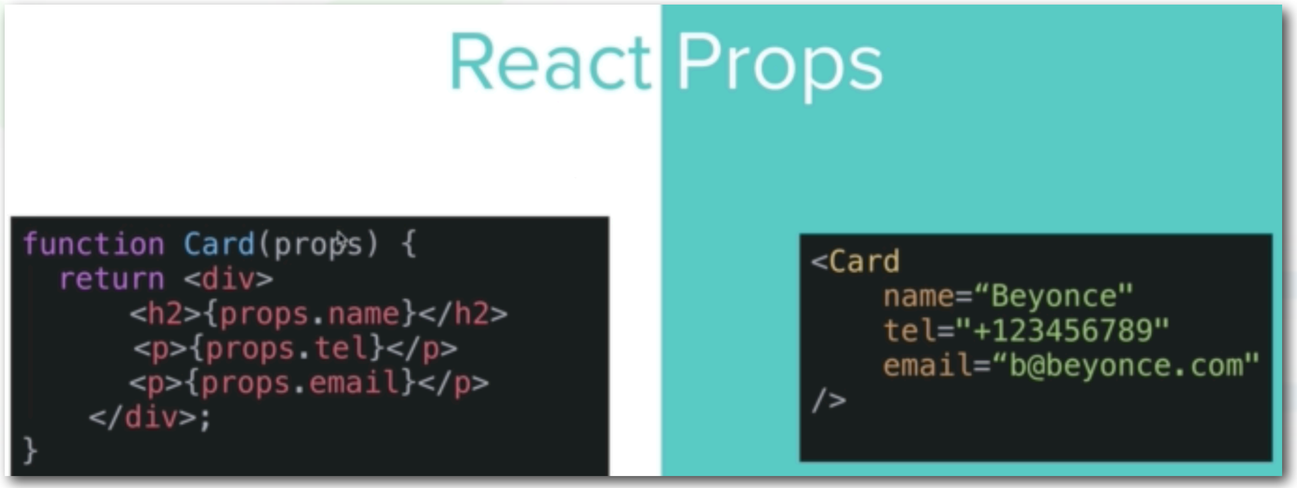
React Props:

We, developers, define the custom properties(props)

Custom components that can be named anything we want, **Not fixed**

When a new Component created, such as Card component, we can pass custom data in the html element using { } as js object, after we create that

The whole object is sent as props object, then we use props.key object to get the value of the custom property and render elements



HTML ATTRIBUTES VS REACT PROPS

HTML attributes:

-class, id, style, src ,..

-ahmet -> No ahmet attribute in HTML.

And I cannot create custom ahmet attribute.

Because HTML attributes are fixed

React Props:

-There is no list for Reach props

-We can create custom props(attributes), such as, ahmet, brand, name, phoneNumber,…

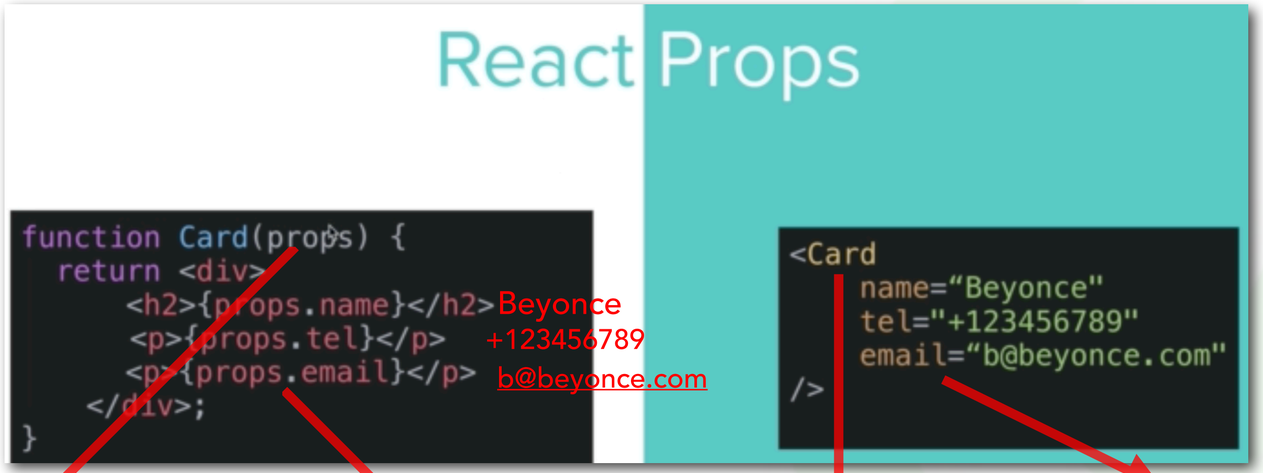
-Props are used to get rid of repetitive items

-Props are used to send the data from one component to the other

-Props are used to create dynamic reusable component

PROPS

REACT PROPS: We cannot apply html classes(such as className, style,...) in the components



Catching
React props

Using
React props

Component

Creating React props.
Name, email, tel props

React props and html attributes looks same. But don't mix because usage is different.

```
function App(){
    return (<div>
        <Header />
        <Footer />
        <Note className="note"/>
    </div>)
}
```

HTML/JSX ATTRIBUTES: We can add style to the html elements. Two of the examples are below.

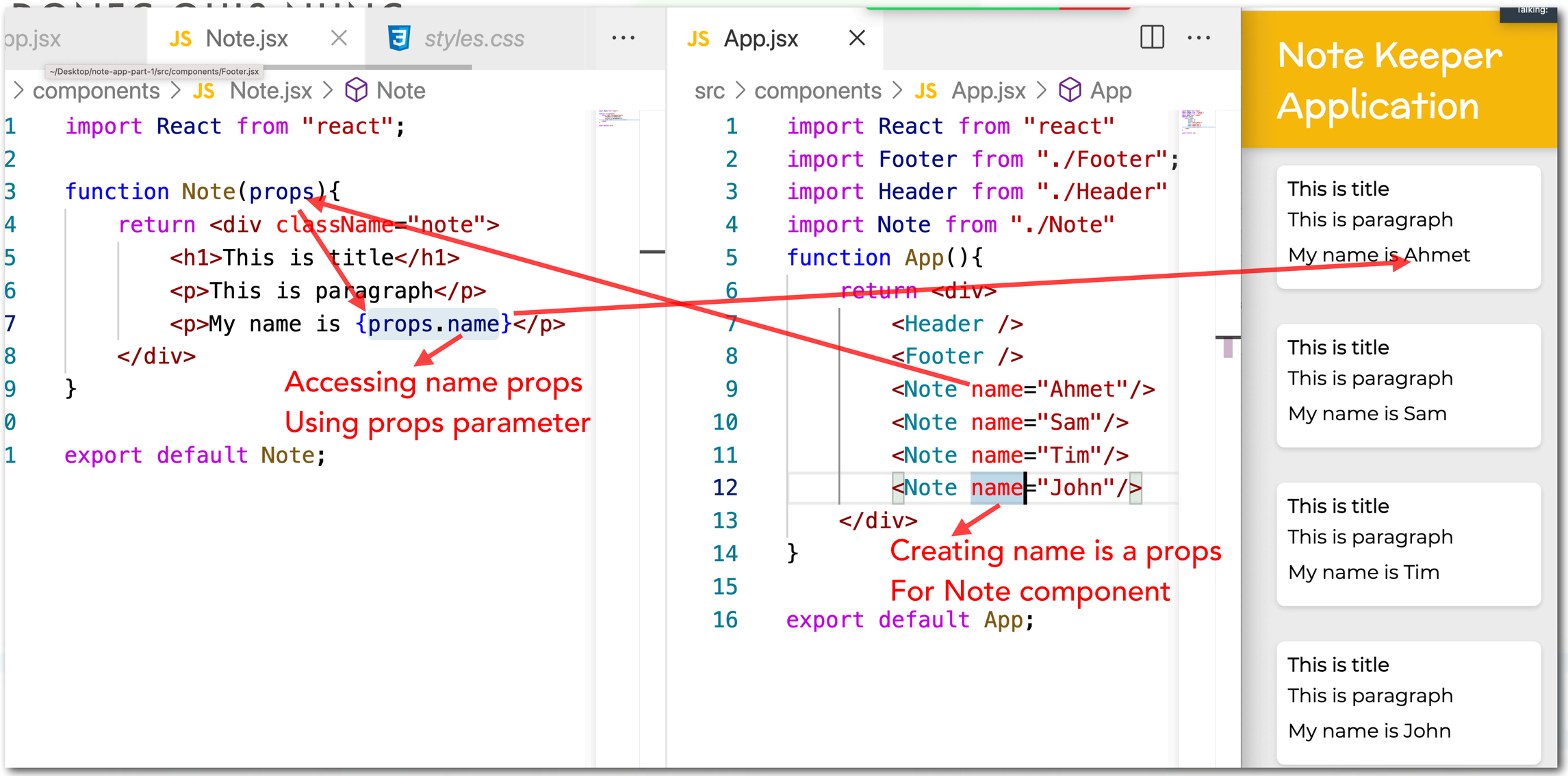
```
return (
  <header>
    <h1 className="note">Keeper</h1>
  </header>
);
```

HTML - JSX element

HTML - JSX property.
This is ready properties. Only ready proprties can be used

```
return <div className="note">
    <h1 > This is the note title </h1>
    <p> This is the note content </p>
</div>
```

REACT PROPS INTRO



ADDING MULTIPLE PROPS

Add phone number and info props on Note component and display on the UI dynamically

NOTE that there are some repetition that will shorten in in the upcoming lessons

p.jsx | Note.jsx | styles.css | Footer.jsx

components > Note.jsx > Note

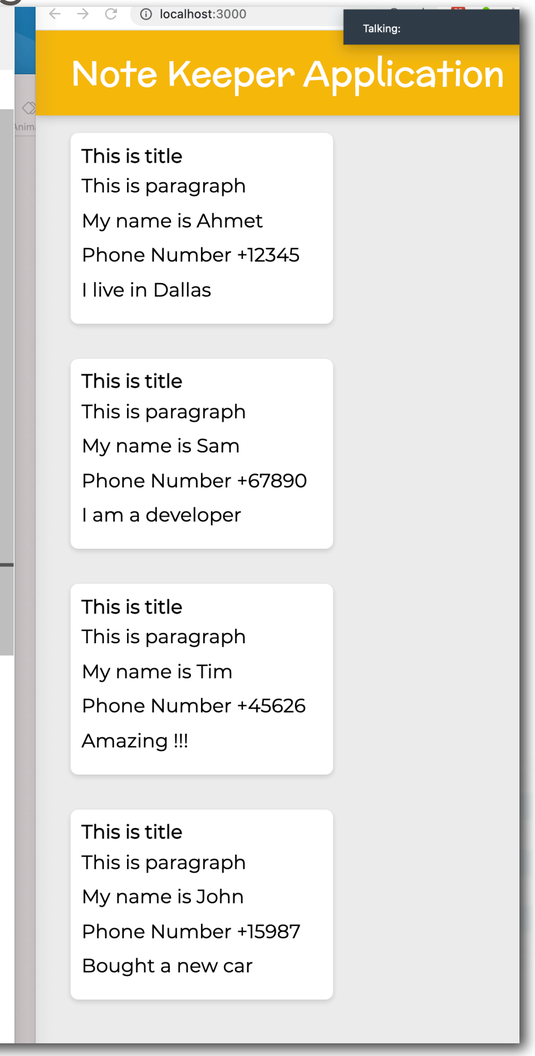
```jsx
import React from "react";

function Note(props){
    return <div className="note">
        <h1>This is title</h1>
        <p>This is paragraph</p>
        <p>My name is {props.name}</p>
        <p>Phone Number {props.phoneNumber}</p>
        <p>{props.info}</p>

    </div>
}

export default Note;
```

App.jsx

You are screen sharing | Stop Share

src > components > App.jsx > ...

```jsx
import React from "react"
import Footer from "./Footer";
import Header from "./Header"
import Note from "./Note"
function App(){
    return <div>
        <Header />
        <Footer />
        <Note
        name="Ahmet"
        phoneNumber="+12345"
        info="I live in Dallas"/>
        <Note
        name="Sam"
        phoneNumber="+67890"
        info="I am a developer"/>
        <Note
        name="Tim"
        phoneNumber="+45626"
        info="Amazing !!!"/>
        <Note
        name="John"
        phoneNumber="+15987"
        info="Bought a new car"/>
    </div>
}

export default App;
```



REACT PROPS

Import react-props starting folder

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

In the index.js we have repetitive code so we can make have a better code.

We will create a component(function) and send different custom data in that component, so each function will render differently

1.In the index.js Create Card() component in the index.js that return a div .

2. Put the first profile information in that div.

3. Put <Card /> inside the render function to render the Card component.Page works same

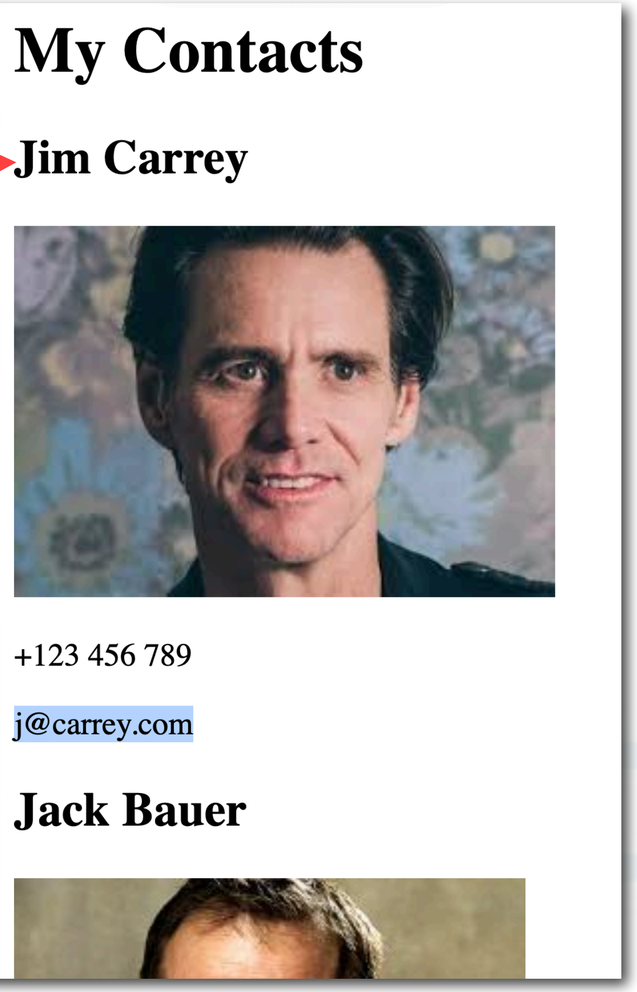
FIRST COMPONENT

Created a card component

Created props for name, img, tel, email and used them in the Card component for Jim Carrey

JS index.js > ...

```
function Card(props){
  console.log(props)
  return <div>
      <h2>{props.name}</h2>
      <img
        src={props.img}
        alt="avatar_img" />
      <p>{props.tel}</p>
      <p>{props.email}</p>
  </div>
}

ReactDOM.render(
  <div>
    <h1>My Contacts</h1>

    <Card
    name="Jim Carrey"
    img="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSVl6f7CWwdVCgb_bEHUT9ZuoY8wz
    tel="+123 456 789"
    email="j@carrey.com"/>
```



2ND AND 3RD

1. Created 2nd component for Jack

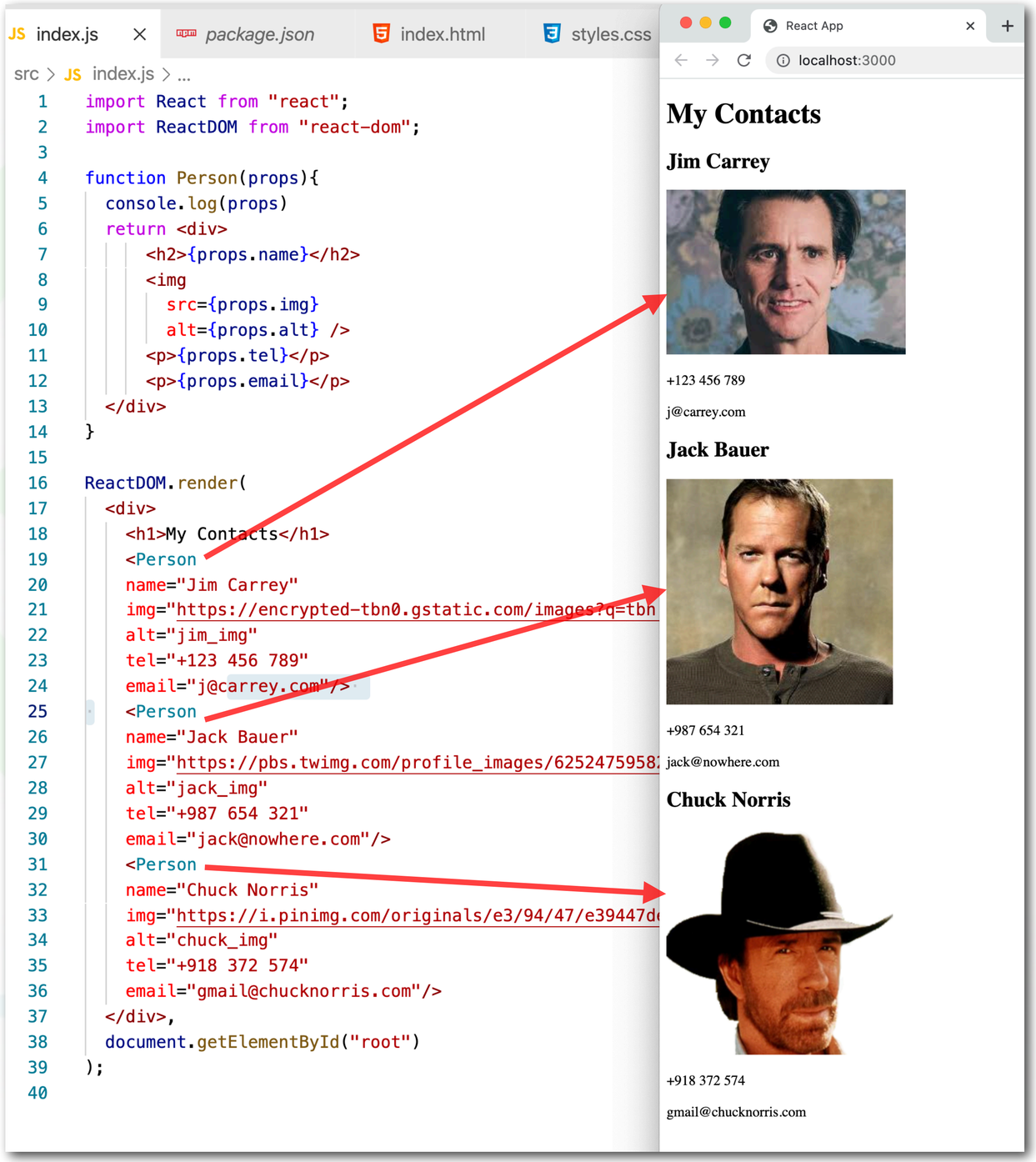
2.Created 3rd component for Chuck

Note: All we did was to copy the Card component and change the data

SINCE THE CUSTOM COMPONENT IS DYNAMIC, I CAN REUSE IT FOR OTHER ELEMENTS

Simply copy and paste the first Card and change the values of the custom properties:

```
<Card
    name="Jack Bauer"
    img="https://pbs.twimg.com/
profile_images/625247595825246208/
X3XLea04_400x400.jpg"
    tel="+987 654 321"
    email="gmail@chucknorris.com"/>

  <Card
    name="Chuck Norris"
    img="https://i.pinimg.com/
originals/e3/94/47/
e39447de921955826b1e498ccf9a39af.png"
    tel="+918 372 574"
    email="gmail@chucknorris.com"/>
```

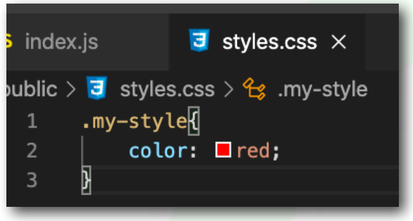


REACT PROPS- STYLING REACH PROPS

FINALLY TO STYLE, WE MUST USE THE style ELEMENT

Add a style in style.css:

```
.my-style{
    color: red;
}
```



How can I apply this style to the texts?? USING HTML ATTRIBUTES!

WE ADD STYLES TO THE HTML ATTRIBUTES, NOT IN THE COMPONENTS, CAUSE REACT WILL THINK THEAT IT IS A CUTOM PROPERTY(PROPS) NOT AN ATTRIBUTE

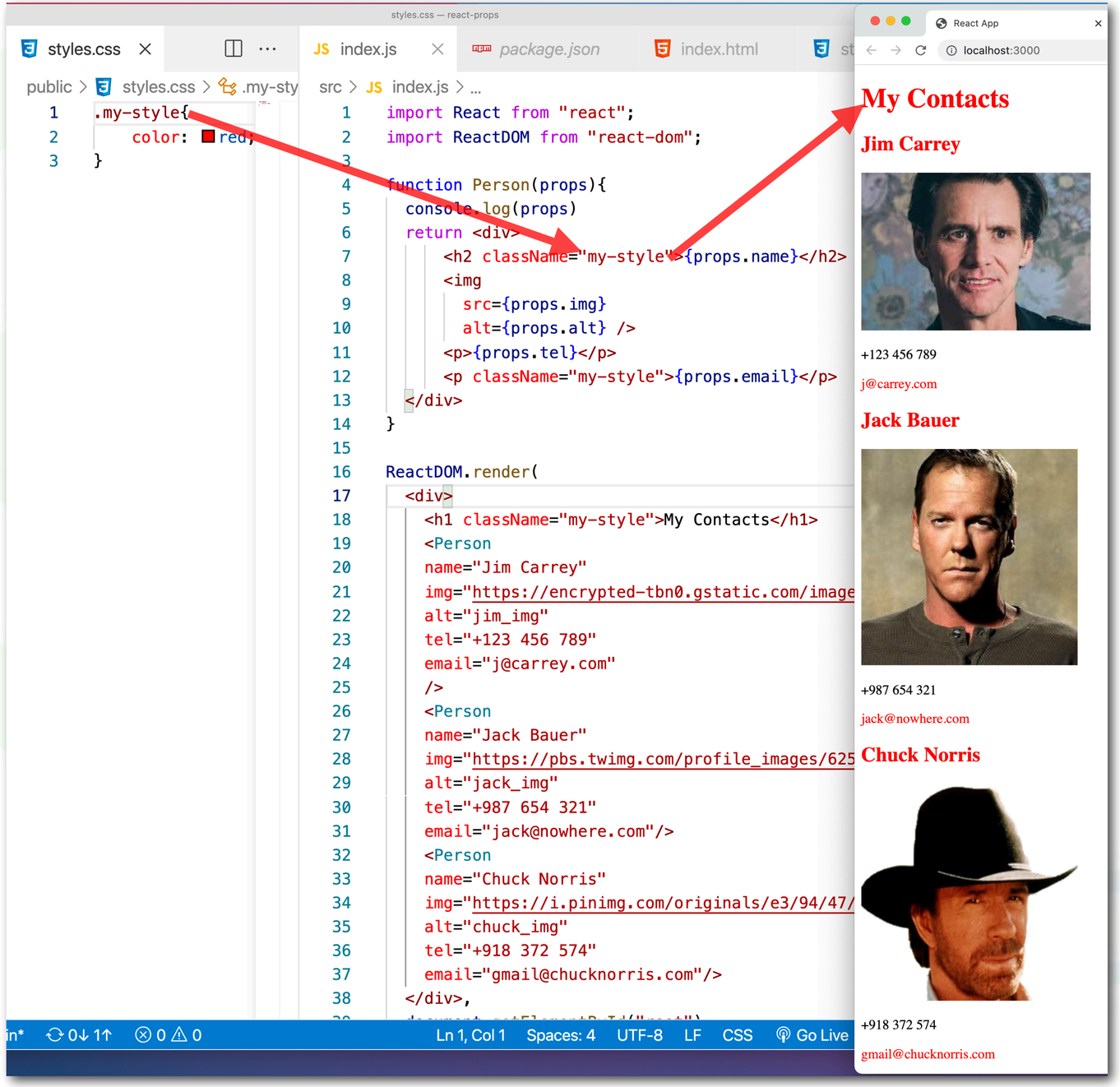
IN THE CARD COMPONENT RATHER THAN PUTTING IN THE <Card />

```
return <div className="my-style">
```

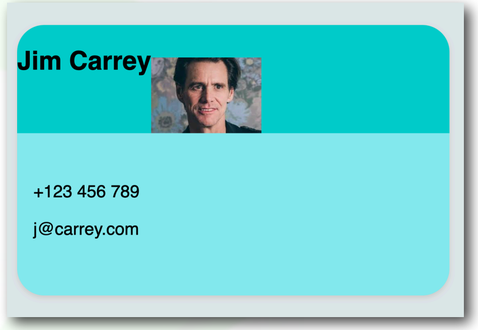
WORKS

```
<Card className="card-style"
```

WONT WORK. REACT WILL THINK THIS IS A CUSTOM COMPONENT, NOT STYLING ATTRIBUTE



REACT PROPS- PRACTICE



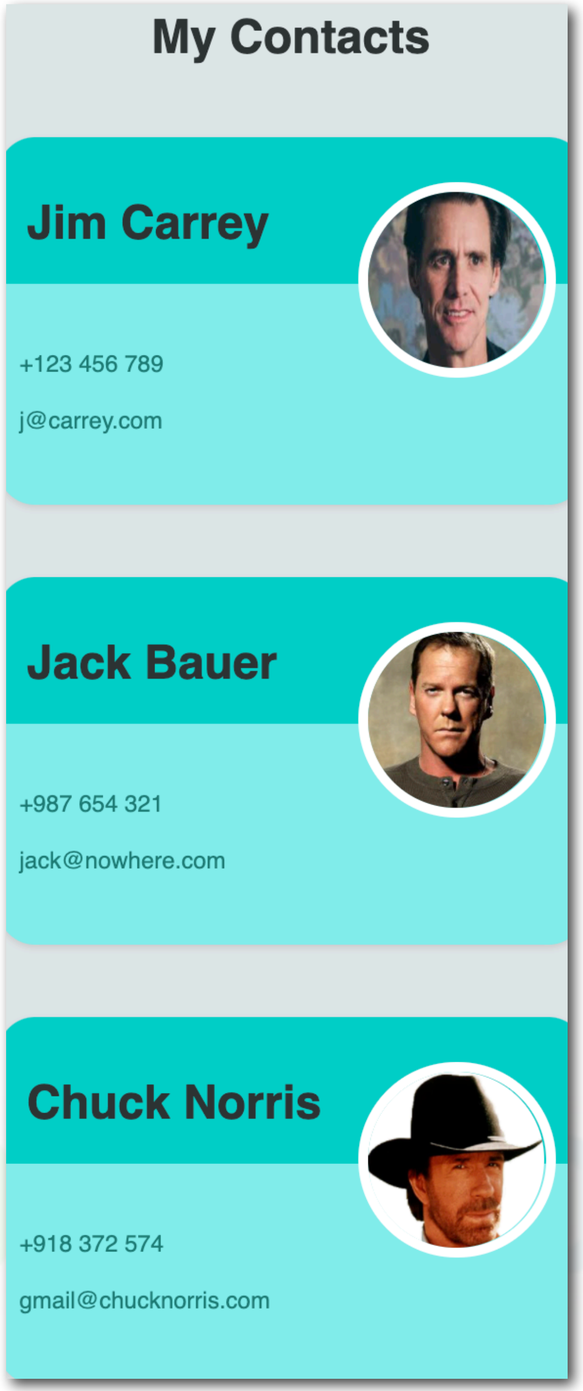
Import react-props-practice

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

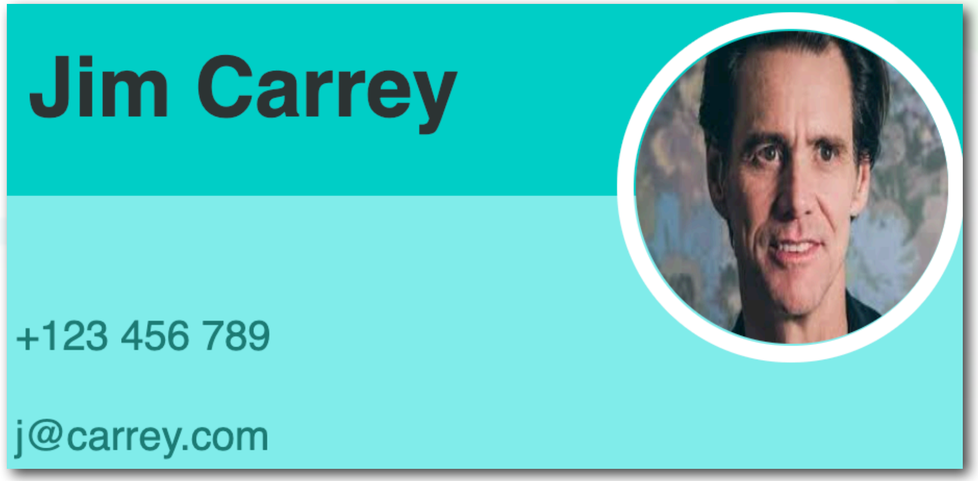
```
//1. Apply CSS styles to App.jsx component to match the appearance
```

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

JS App.jsx × JS Card.jsx JS *contacts.js*

src > components > JS 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;
```

JS Card.jsx × JS *contacts.js*

components > JS 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.c
       tel="+123 456 789"
       email="j@carrey.com"
```

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

```
export default contacts;
```
2

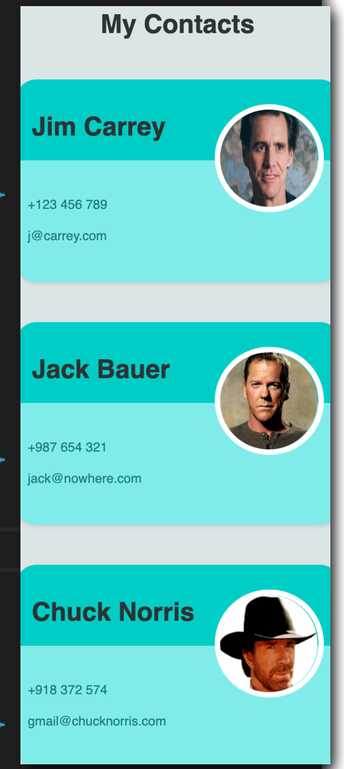
3. Import contact in App.jsx: import contacts from "../contacts"; **!!! Use .. /for path**

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
```

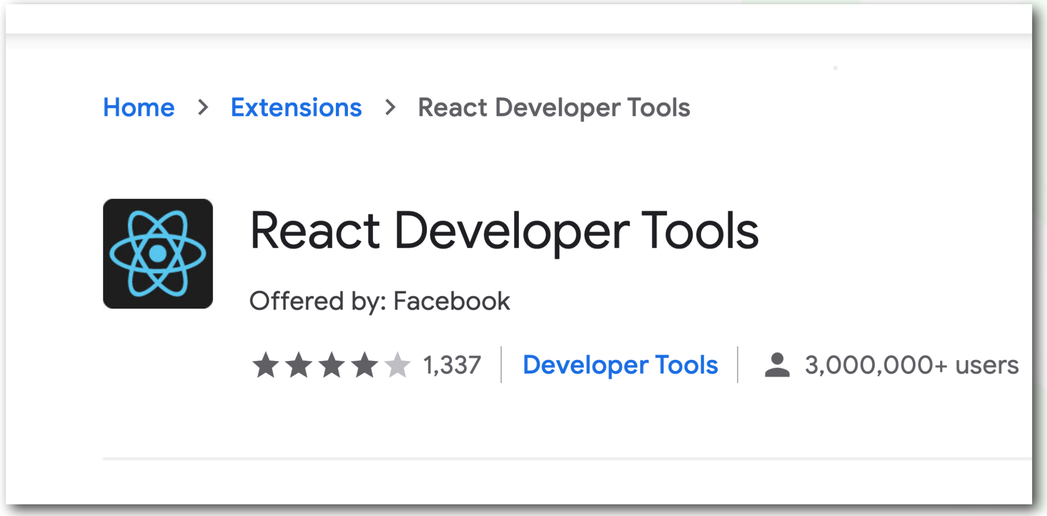
```
import contacts from "../contacts";
//Testing if i am able to read contact component
// console.log(contacts)
// console.log(contacts[0].name)

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

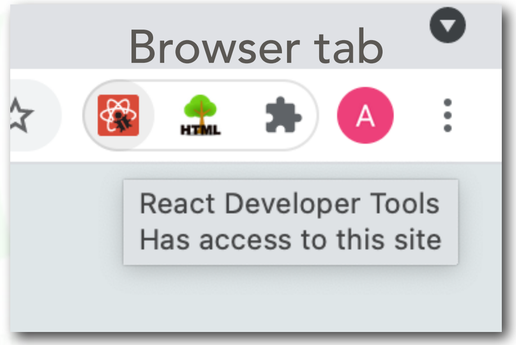
4

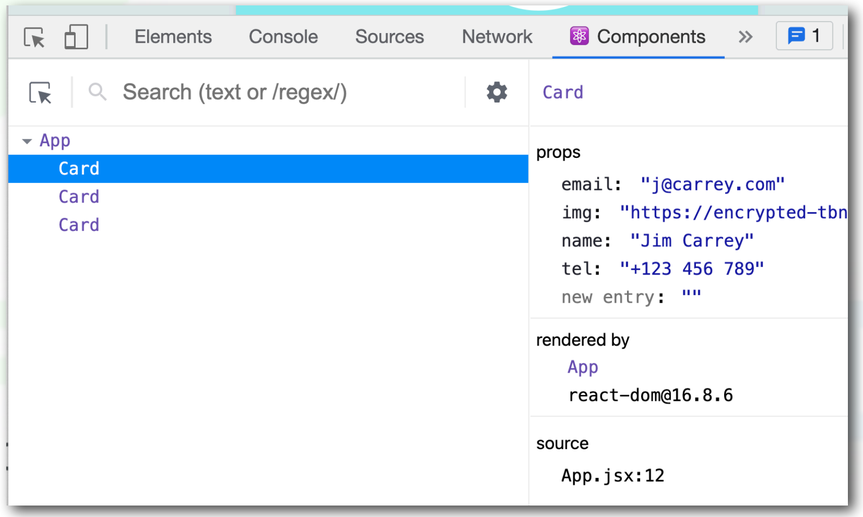


INSTALL REACT DEV TOOL

1. Search for react developer tools on chrome
2. Install the extension







React Dev tool help us see the the component tree

Props, hooks, etc.

REACT DEV TOOL

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

Avatar component will be a child of Card 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

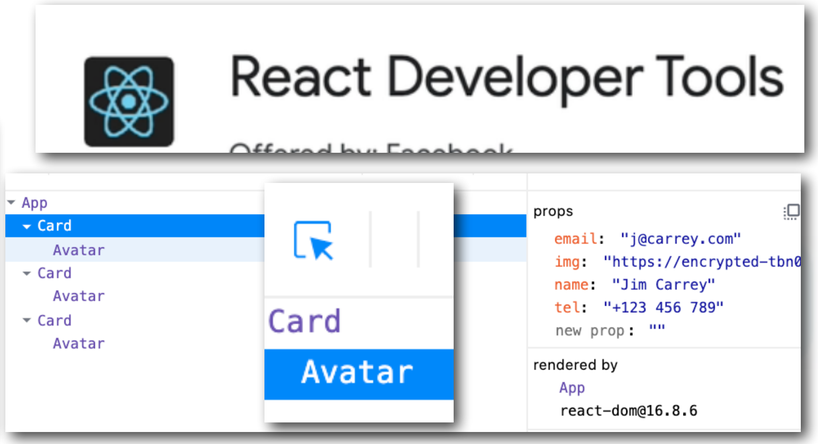
Card.jsx

```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>
```

Failed to compile

```
./src/components/Avatar.jsx
  Line 5:10:  'props' is not defined  no-undef

Search for the keywords to learn more about each error.
```

This error occurred during the build time and cannot be dismissed.



SOLUTION

1. Create Avatar component and return img tag that is related to avatar

2. Use props parameter to catch the props from Avatar that is inside the Card component

3. In the Avatar that is inside the Card component, create img = {props.img} to catch the img the Card that is inside the App component

Passes the img in this order: **contact.js => App => Card** => Avatar

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

App.jsx

```
import React from "react";
import Card from "./Card";
import contacts from "../contacts";
// console.log(contacts)
// console.log(contacts[0]);
// console.log(contacts[0].name);//Jim C

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

Card.jsx

```
import React from "react";
import Avatar from "./Avatar";
function Card(props){
    return <div className="card">
    <div className="top">
      <h2 className="heading">{props.nam
      <Avatar img={props.img}/>3
    </div>
    <div className="bottom">
      <p className="info">{props.tel}</p
      <p className="info">{props.email}<
    </div>
  </div>
}
export default Card;
```

Avatar.jsx

```
import React from "react";

function Avatar(props){
    return <img className="circle-img"
    src={props.img} 2
    alt="avatar_img"
  />
}

export default Avatar;
```

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
https://media.wfaa.com/assets/WFAA/images/6ccb7e54-8529-4c8f-98f8-5a9ce6592482/6ccb7e54-8529-4c8f-98f8-5a9ce6592482_1920x1080.jpg

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

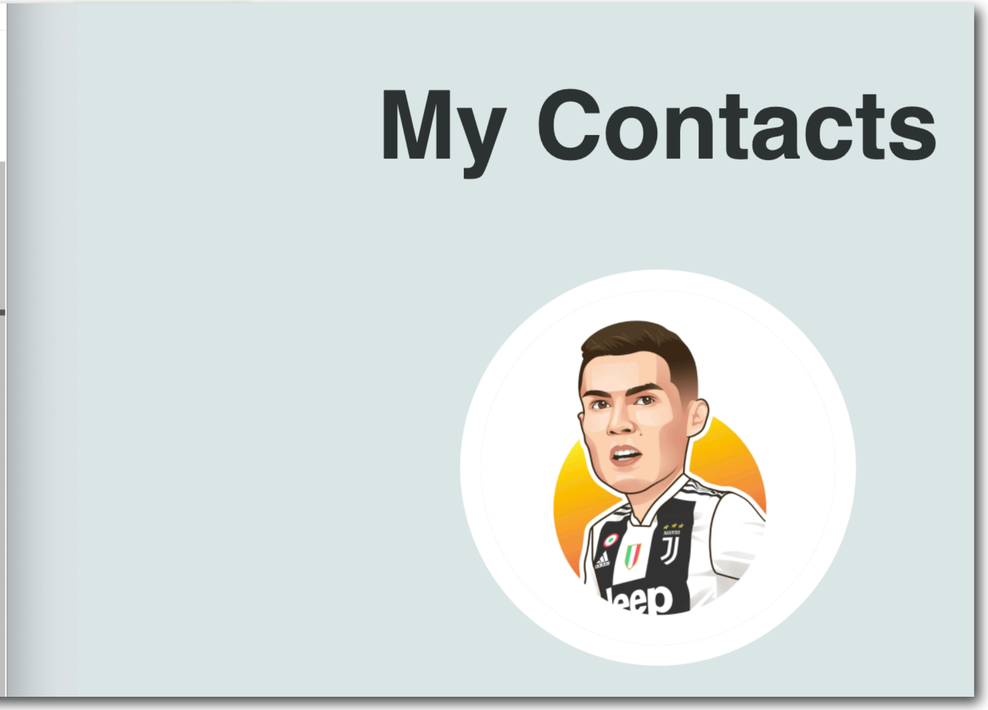
So just using the simple Avatar component, added an Avatar

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

SOLUTION

https://i.pinimg.com/originals/b2/3e/a3/b23ea34c18999b1bddb2f49199cfd871.jpg

```
// console.log(contacts[0].name);//Jim Carrey

function App() {
  return (
    <div>
      <h1 className="heading">My Contacts</h1>
      <Avatar img="https://i.pinimg.com/originals
      <Card
        name={contacts[0].name}
        img={contacts[0].imgURL}
        tel={contacts[0].phone}
        email={contacts[0].email}
```



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

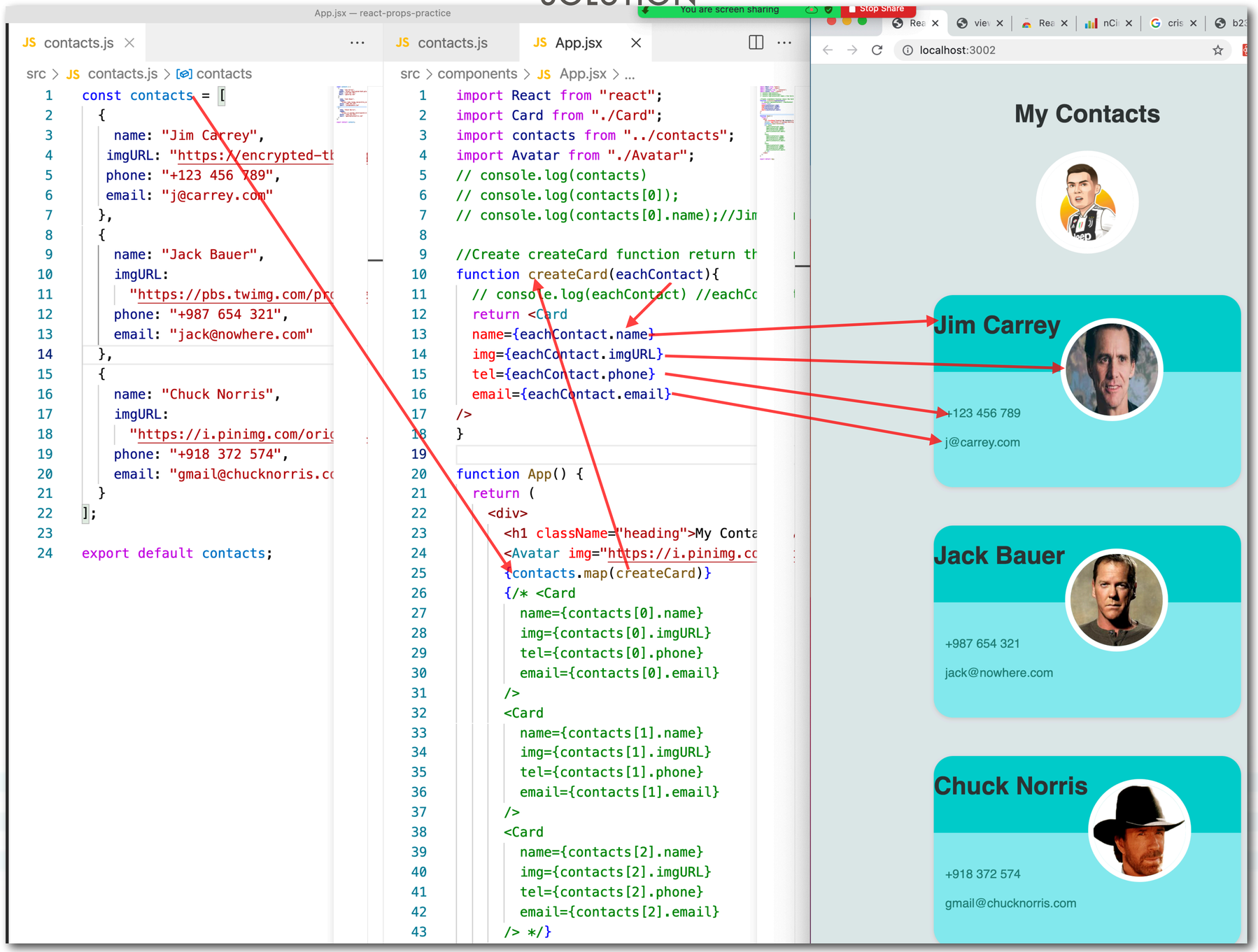
```
function createCard(contact){
  return <Card
  name={contact.name}
  />
}
```

3

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

4

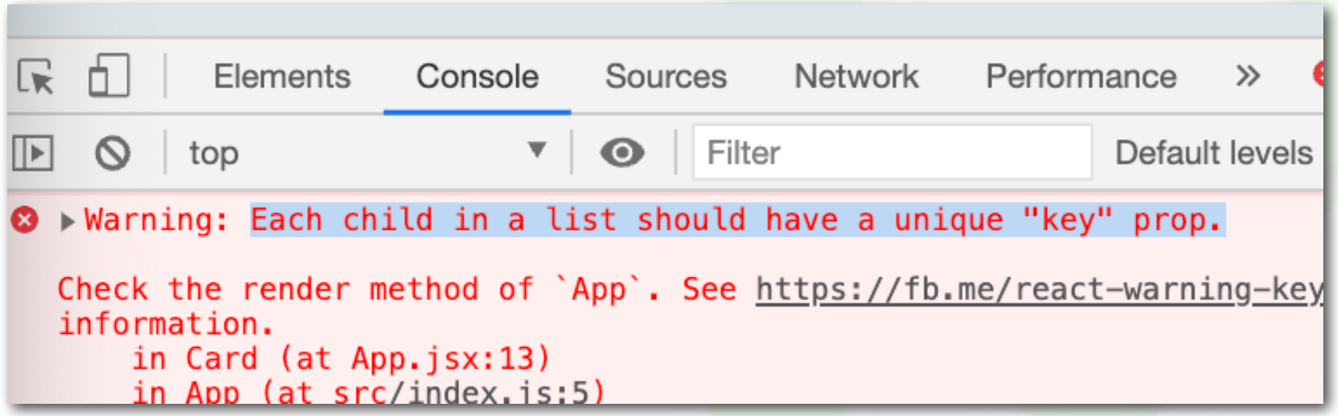
SOLUTION


App.jx — react-props-practice
You are screen sharing
Stop Share
contacts.js
src > JS contacts.js > contacts
const contacts = [
{
name: "Jim Carrey",
imgURL: "https://encrypted-tb
phone: "+123 456 789",
email: "j@carrey.com"
},
{
name: "Jack Bauer",
imgURL:
"https://pbs.twimg.com/pr
phone: "+987 654 321",
email: "jack@nowhere.com"
},
{
name: "Chuck Norris",
imgURL:
"https://i.pinimg.com/orig
phone: "+918 372 574",
email: "gmail@chucknorris.co
}
];
export default contacts;
contacts.js
App.jsx
src > components > JS App.jx > ...
import React from "react";
import Card from "./Card";
import contacts from "../contacts";
import Avatar from "./Avatar";
// console.log(contacts)
// console.log(contacts[0]);
// console.log(contacts[0].name);//Jim
//Create createCard function return th
function createCard(eachContact){
// console.log(eachContact) //eachCo
return <Card
name={eachContact.name}
img={eachContact.imgURL}
tel={eachContact.phone}
email={eachContact.email}
/>
}
function App() {
return (

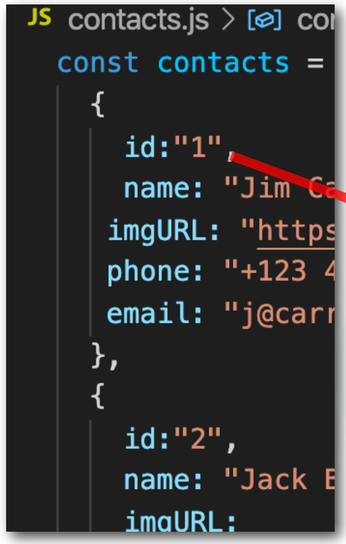
<h1 className="heading">My Conta
<Avatar img="https://i.pinimg.co
{contacts.map(createCard)}
{/* <Card
name={contacts[0].name}
img={contacts[0].imgURL}
tel={contacts[0].phone}
email={contacts[0].email}
/>
<Card
name={contacts[1].name}
img={contacts[1].imgURL}
tel={contacts[1].phone}
email={contacts[1].email}
/>
<Card
name={contacts[2].name}
img={contacts[2].imgURL}
tel={contacts[2].phone}
email={contacts[2].email}
/> */}
localhost:3002
My Contacts
Jim Carrey
+123 456 789
j@carrey.com
Jack Bauer
+987 654 321
jack@nowhere.com
Chuck Norris
+918 372 574
gmail@chucknorris.com

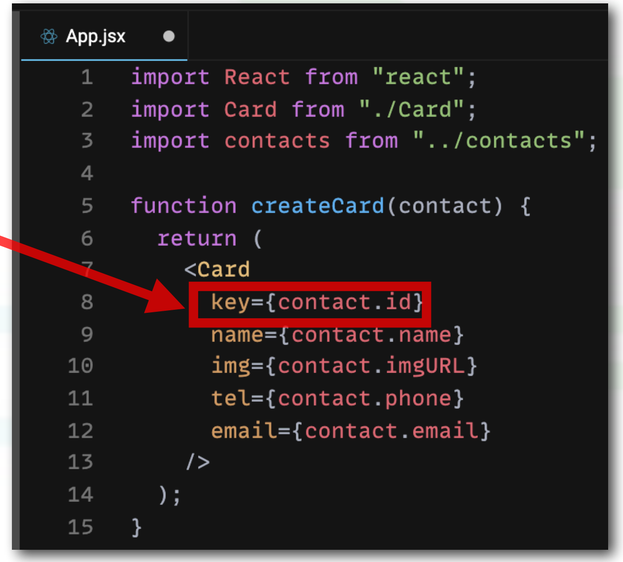
MAPPING DATA TO COMPONENTS AND KEY PROPERTY

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.

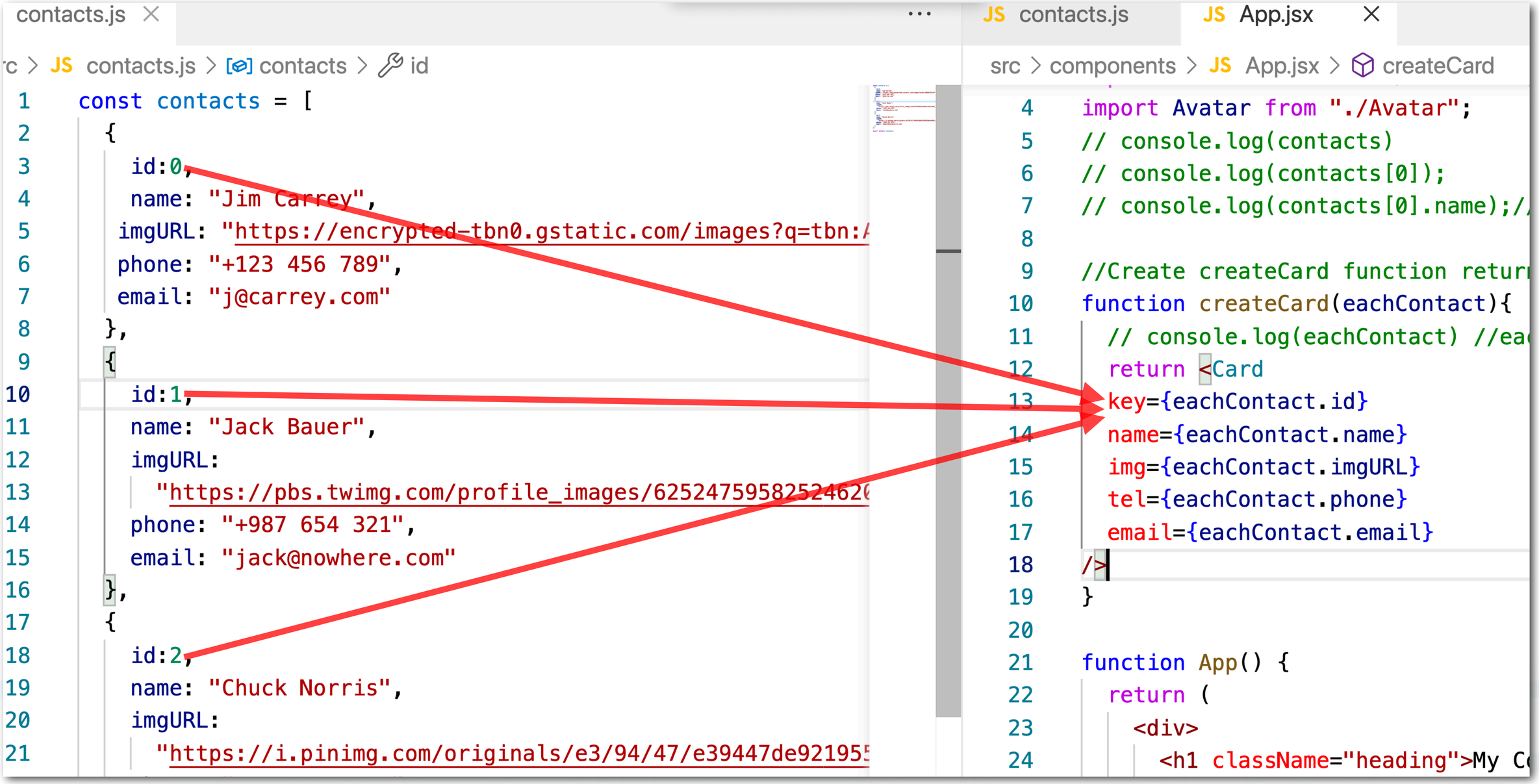
It will help identify each different component

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

KEY PROPERTY FOR IDENTIFYING UNIQUE LISTS



MAPPING DATA TO COMPONENTS PRACTICE

Import mapping components practice .

npm install, npm install react-scripts start, 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

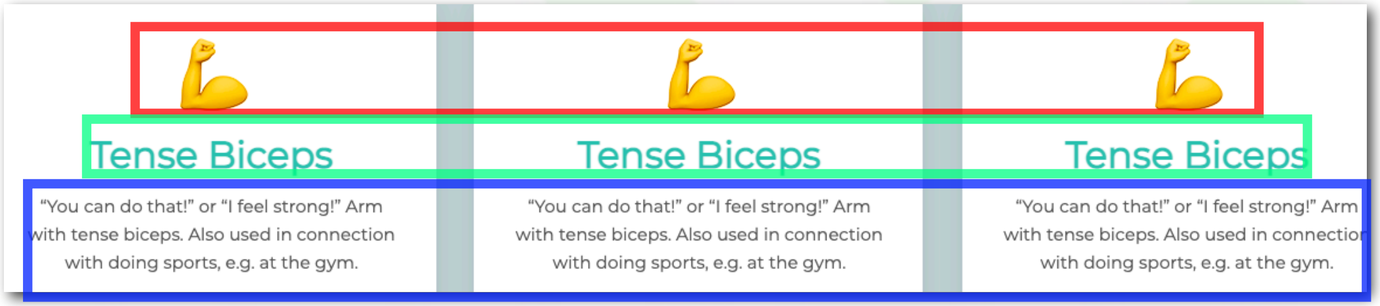
3. Map through the emojipedia Entry.jsx array and render Entry components

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

op.jsx | JS Entry.jsx ×

components > JS Entry.jsx > default

```jsx
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.

JS emojipedia.js | App.jsx × | Entry

src > components > App.jsx > App

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

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

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

      </dl>
    </div>
  );
}

export default App;
```

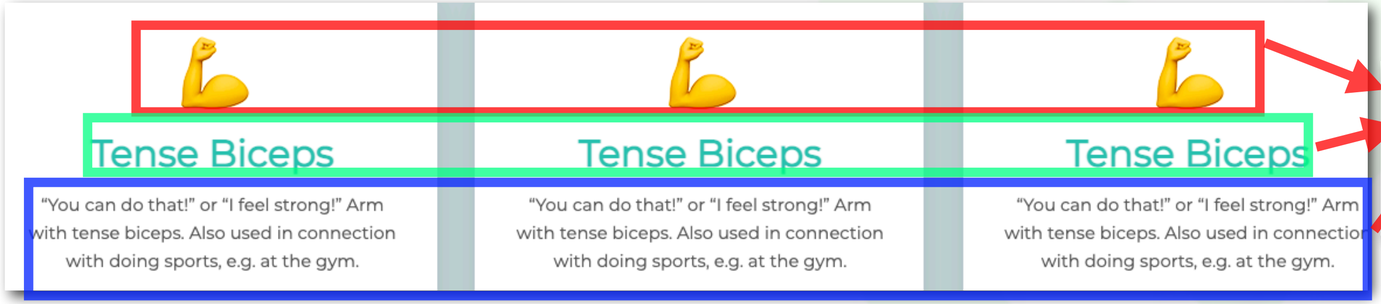
MAPPING DATA TO COMPONENTS PRACTICE-2. CREATE PROPS TO REPLACE THE HARD CODED PART

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

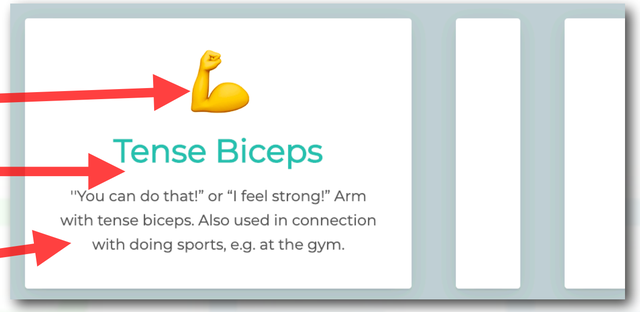
2. In Entry, pass props, and send props.emoji, props.name, props.descriptions




```
<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:

```
23  ];
24  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

```
1   import React from "react";
2   import Entry from "./Entry";
3   import emojipedia from "../emojipedia";
4
5   function createEntry(emojiTerm){
6   return <Entry
7       key={emojiTerm.id}
8       emoji={emojiTerm.emoji}
9       name={emojiTerm.name}
10      description={emojiTerm.description}
11  />
12  }
13
14  function App() {
15    return (
16      <div>
17        <h1>
18          <span>emojipedia</span>
19        </h1>
20
21        <dl className="dictionary">
22        {emojipedia.map(createEntry)}
23
24        </dl>
25      </div>
26    );
27  }
28
29  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;
```

KEEPER APP PROJECT PART 2

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

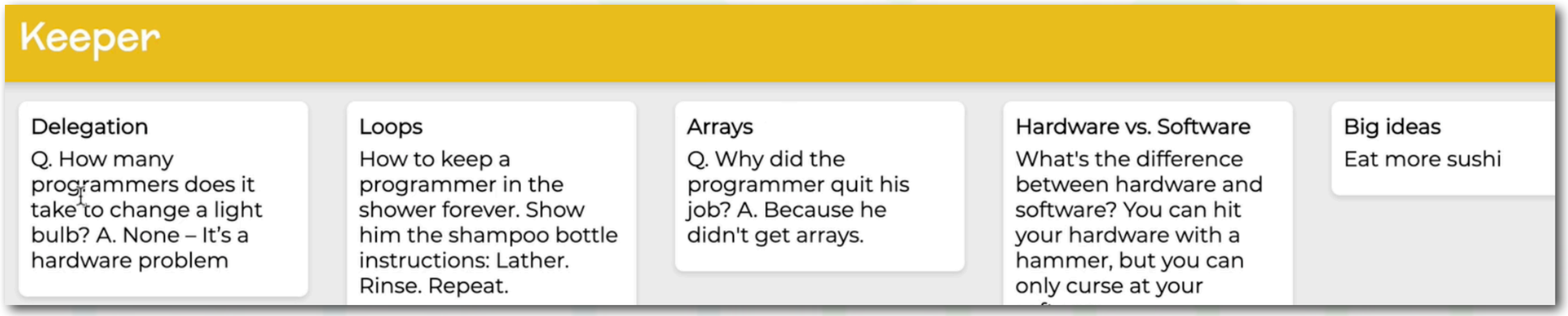
TASK://Render all the notes inside notes.js as a seperate Note component.

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.



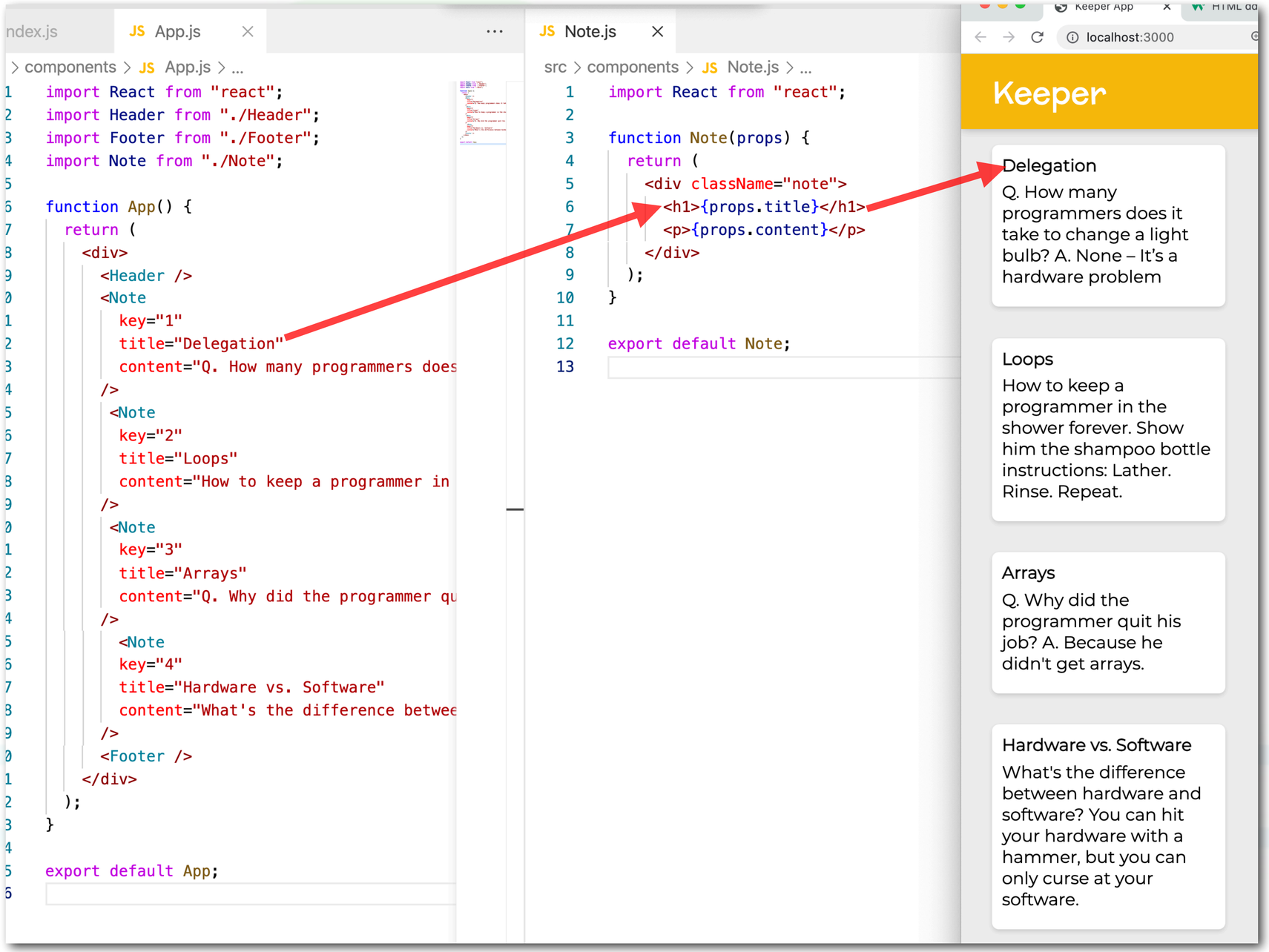
SOLUTION WITHOUT MAP FUNCTION

This works

But there are

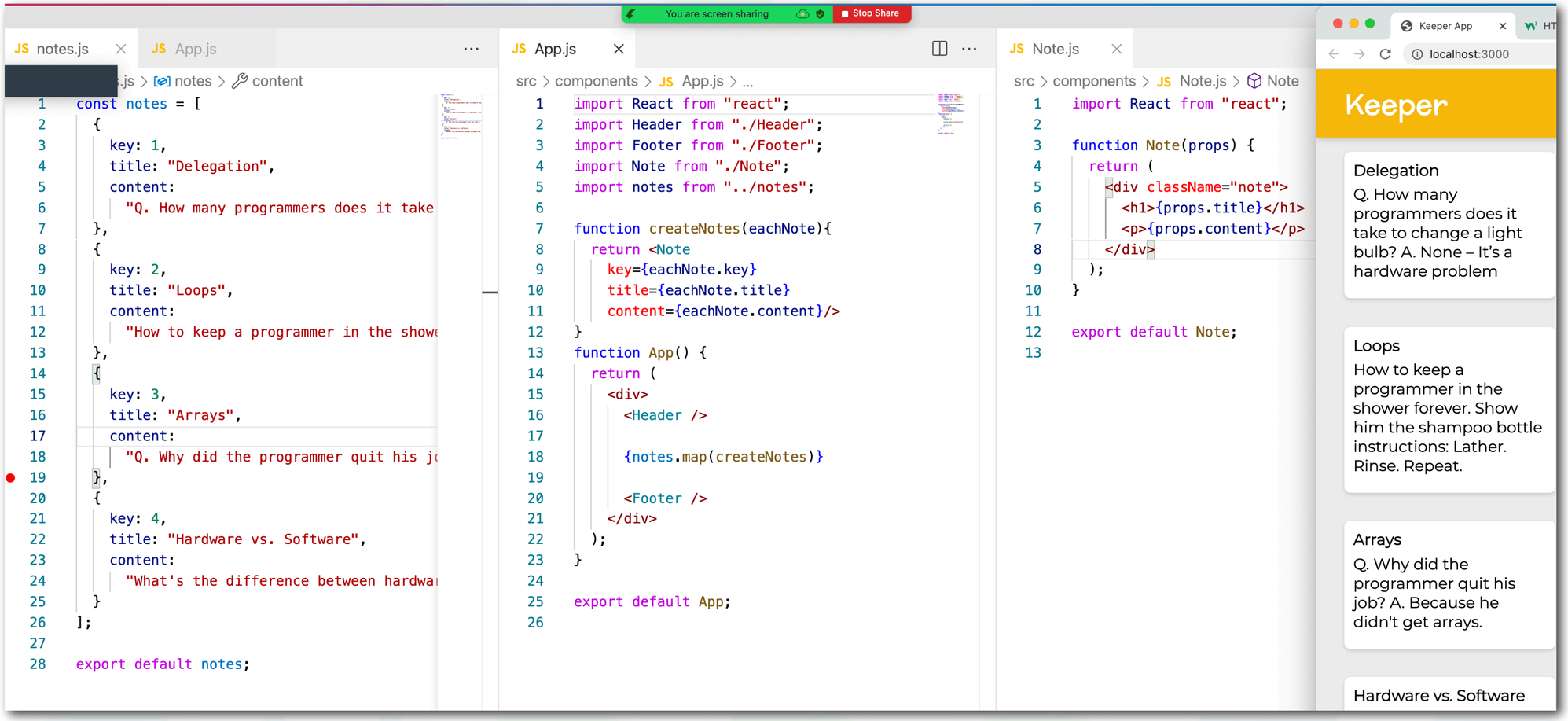
Repeatitions

So we use maps



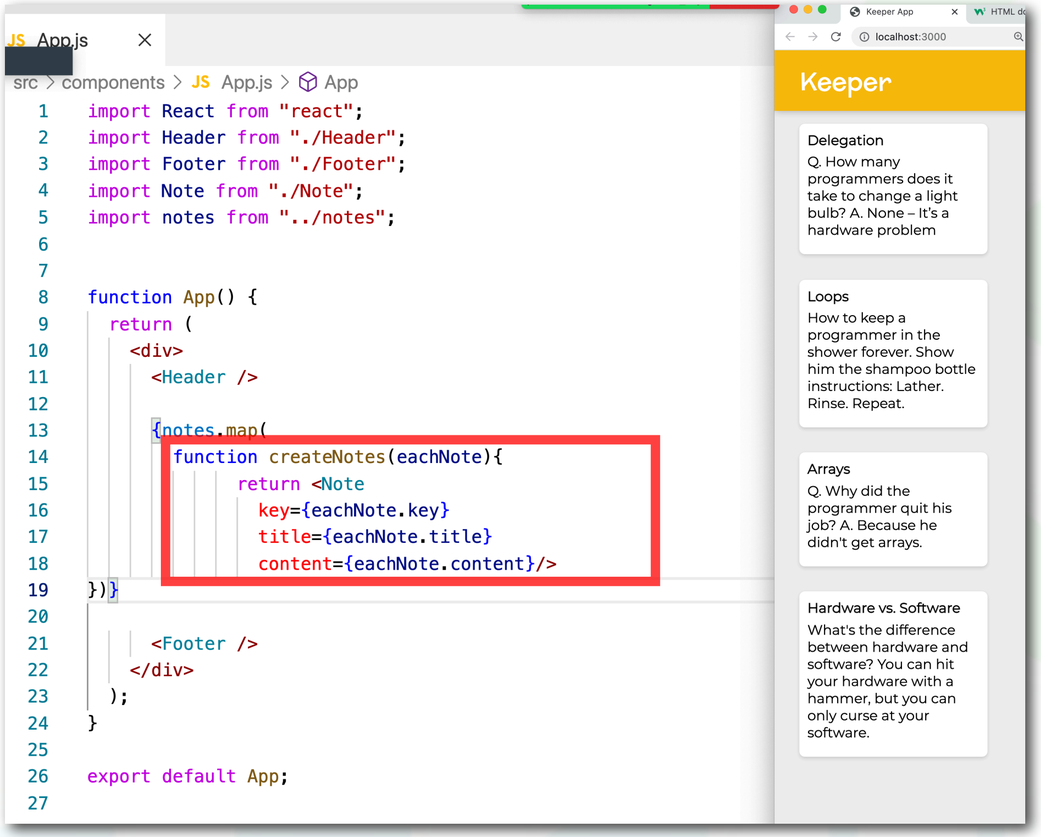
STEPS

1. Export notes
2. Import notes in App
3. Get the date from notes array {notes.map(createNotes)}
4. Create createNotes function that will return Note component
5. Make sure to add properties to show the title and content on the UI

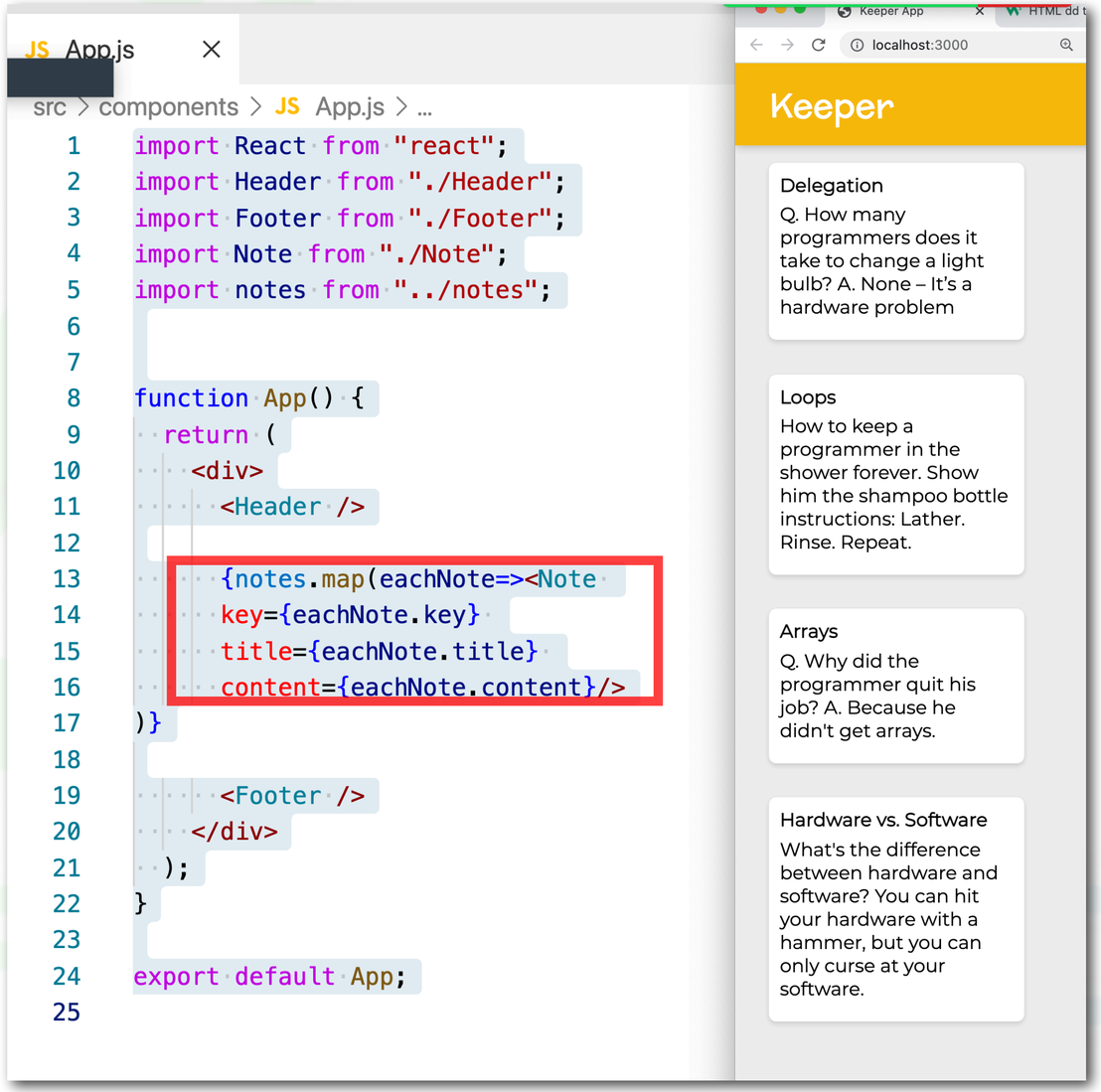


ALTERNATIVE WAY OF PASSING THE FUNCTION

1. We can use createNotes function directly inside the map function



1. We can use arrow function and simplify the code



KEEPER APP PROJECT PART 2- ALL SOLUTIONS IN ONE PAGE

TASK://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
];

export default notes;
```

```
// App.js
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:

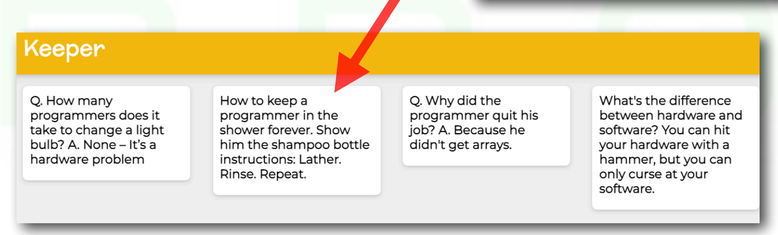
```
function createNotes(noteItem){
  return <Note
  key={noteItem.key}
  title = {noteItem.title}
  content = {noteItem.content}/>
}
```

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

```
// Note.js
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
import React from "react";
import Header from "./Header";
import Footer from "./Footer";
import Note from "./Note";
import notes from "../notes";

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"
      /> */}
      {notes.map(createNotes)}
      <Footer />
    </div>
  );
}

export default App;
```

```
// notes.js
const notes = [
  {
    key: 1,
```



KEEPER APP PROJECT PART 2

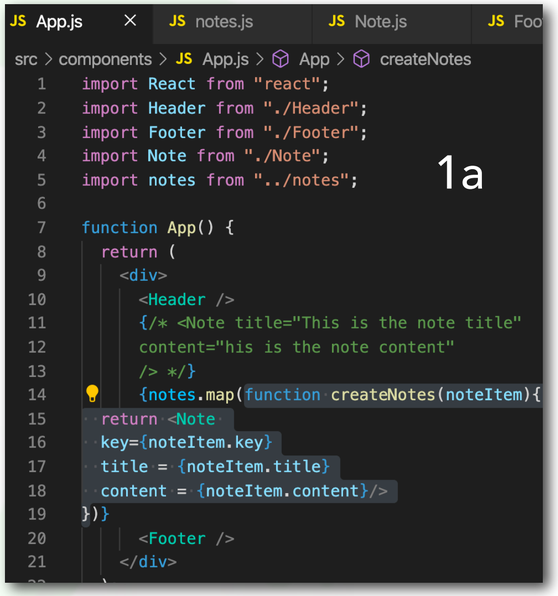
We can write this code in different ways:

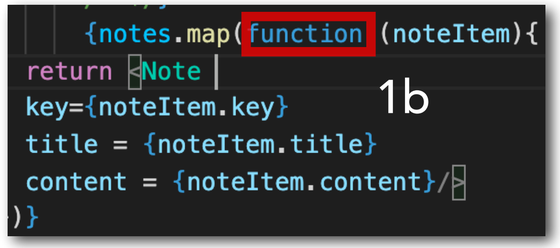
1a. We can put createNote function directly in the map function: So cut createNotes function and paste inside the map:

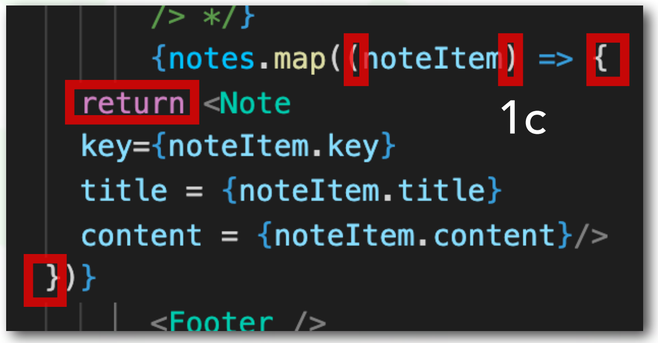
1b. I can make it anonymous ruction by deleting name:Delete createNotes

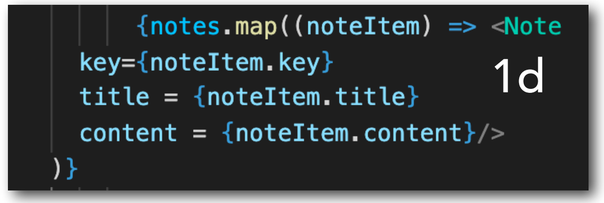
1c. Or Also remember we can use arrow functions, so we can simplify arrow to make it looks shorter: Delete function keyword, use =>

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









CONDITIONAL RENDERING WITH TERNARY OPERATOR AND & OPERATOR

Think about log in screen. We will create a log in flow

When user is not log in, they should see the log in form, otherwise see the login version of the web

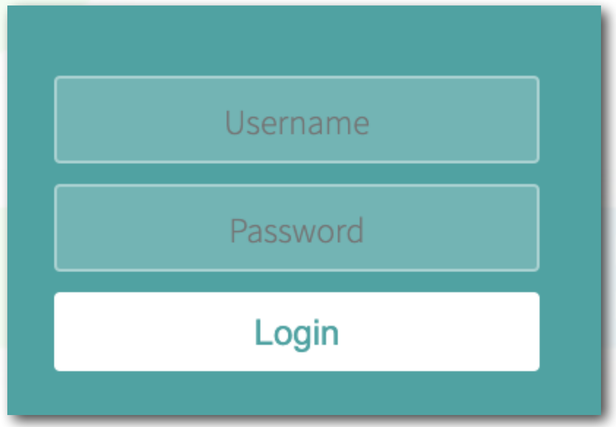
Import **conditional-rendering**

npm install react-scripts start

npm install

npm start

TASK: We will render Hello only is user is logged on, otherwise show login form



CONDITIONAL RENDERING WITH TERNARY OPERATOR @ AND OPERATOR

1. In to App.js Create a var isLoggedIn=true; This will be used to check if we loved in or not We will render Hello is isLoggedIn true, else render the form

2. In App.jsx, Create a function renderConditionally(). If user logs in, This will return <h1>Hello</h1>,otherwise this will return the log in form elements

3. Inside the App( ) function call renderConditionall function as js object:

4. Test your code: change isLogIn false and true, refresh page, and see it it works

```
import React from "react";

var isLoggedIn=false;
function renderConditionally(){
  if(isLoggedIn===true){
    return <h1>Hello</h1>;
    }else{
      return (
        <form className="form">
          <input type="text" placeholder="Username" />
          <input type="password" placeholder="Password" />
          <button type="submit">Login</button>
        </form>
      );}}

function App() {
  return  <div className="container">{renderConditionally()} </div>
}
export default App;
```

This is one way of doing that rendering conditionally

But Each component should have single responsibility

So we will create Login component for the entire form component

CONDITIONAL RENDERING WITH TERNARY OPERATOR @ AND OPERATOR

Create Login.jsx and and add the form from App.jsx

And render <Login/> component in the renderConditionally function

Login.jsx

components > JS Login.jsx > Login

```jsx
import React from "react";

function Login(){
    return <form className="form">
    <h1>Enter Credentials</h1>
    <input type="text" placeholder="Username" />
    <input type="password" placeholder="Password" />
    <button type="submit">Login</button>
  </form>
}

export default Login;
```

App.jsx

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

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

//logic
var isLoggedIn=false;

function renderConditionally(){
  if(isLoggedIn===true){
    return <h1>Hello! You successfully logged in!</h1>
  }else{
    return <Login />
  }
}

function App() {
  return (
    <div className="container">
      {renderConditionally()}
    </div>
  );
}

export default App;
```

CONDITIONAL RENDERING WITH TERNARY OPERATOR @ AND OPERATOR

NOW EXTRACT THE INPUT COMPONENT

1. Create **Input.jsx component,** import react, Create Input function, return the input elements

2. In Login.jsx, import Input, and return <Input />

3. We need two props, one of username, one for password in <Input />

```
function Login(){
    return <form className="form">
   <Input type = "text"
      placeholder="Username"/>
    <Input type = "password"
      placeholder="Password"/>
    <button type="submit">Login</b
```

4. In Input.jsx To get eta values of the username and password properties in the Input components, We need to use props:

```
function Input(props){
    return <div>
            <input
            type={props.type}
            placeholder={props.placeholder} />
    </div>
```

5. Now the website looks good!!!

6. We CAN MAKE THIS CODE SHORTER??

Input.jsx

```
import React from "react";

function Input(props){
    return <input type={props.type}
    placeholder={props.placeholder}/>
}
export default Input;
```

Login.jsx

```
1   import React from "react";
2   import Input from "./Input";
3   function Login(){
4       return <form className="form">
5       <h1>Enter Credentials</h1>
6
7       <Input
8       type="text"
9       placeholder="Username"/>
10
11      <Input
12      type="password"
13      placeholder="Password"/>
14
15      {/* <input type="text" placeholder
16      <input type="password" placeholder
17
18      <button type="submit">Login</butto
19    </form>
20  }
21
22  export default Login;
```

CONDITIONAL RENDERING WITH TERNARY OPERATOR @ AND OPERATOR

Now instead of creating function and putting in the curly braces in App. Function, we will directly create the function inside

In jsx everything inside the curly braces must be an expression (ternary) NOT A STATEMENT(loops, switch, if else statement).

So inserting the renderConditionally function will not work. But we can use ternary

1. In App.jsx, Use ternary to render <h1>Hello</h1> if true, <Login/> if false. We can delete renderConditionally function and object now.

App.jsx × JS Login.jsx

c > components > JS App.jsx > App

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

//logic
var isLoggedIn=false;

function App() {
  return (
    <div className="container">
      {
        function renderConditionally(){
    if(isLoggedIn===true){
      return <h1>Hello! You successfully logged in!</h1>
    }else{
      return <Login />
    }}
      }
    </div>
  );
}

export default App;
```

I cannot use statements inside {}.
So if statement will not work here

JS App.jsx × JS Login.jsx Talking:

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

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

//logic
var isLoggedIn=false;

function App() {
  return (
    <div className="container">
      { isLoggedIn===true ? <h1>Hello! You successfully logged in!</h1> : <Login />}
    </div>
  );
}

export default App;
```

Statements cannot be used but expressions can be used
We can use ternary to render conditionally

CONDITIONAL RENDERING WITH TERNARY OPERATOR @ AND OPERATOR

If currentTime > 12, then show Why are you still working? otherwise DON'T SHOW ANYTHING

App.jsx | Login.jsx

src > components > App.jsx > App

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

//logic
var isLoggedIn=false;

// var currentTime = new Date().getHours();
var currentTime = new Date(2021,11,11,10).getHours();
console.log((currentTime));
function App() {
  return (
    <div className="container">
      { isLoggedIn===true ? <h1>Hello! You successfully logged in!</h1> : <Login />}

      {/* if current time > 12 then render message, otherwise don't do anything */}
      {currentTime > 12 ? <h1>Why are you still working? </h1> : null }
    </div>
  );
}

export default App;
```

With and operator && What we are saying is if left side is true, render the right side, else don't render anything

WORKS FINE. THERE IS && OPERATON IN JS WE CAN USE TO MAKE THE CODE EVEN SHORTER

CONDITIONAL RENDERING WITH TERNARY OPERATOR @ AND OPERATOR

With and operator && What we are saying is if left side is true, render the right side, else don't render anything

```
(EXPRESSION && EXPRESSION)
```

IF BOTH TRUE THEN RETURN TRUE ELSE RETURN FALSE

THIS MEANS IF 1st EXPRESSION IS FALSE, THEN THIS RETURNS FALSE

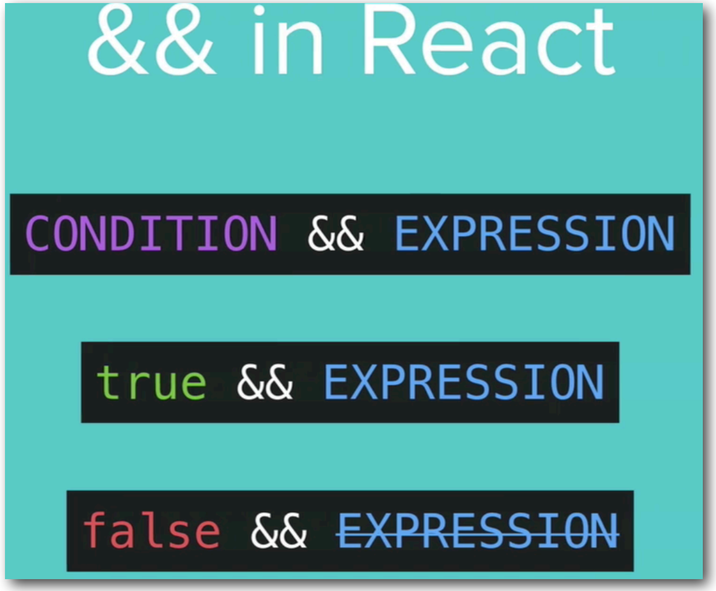
1. We can DELETE NULL and use && instead of ? operator

```
currentTime > 12 && <h1>Why are you still working?</h1>
```

This makes sense because if currentTime>12 left won't be checked

IN CONCLUSION, AS DEVELOPER, WE HAVE CHOICES AND WRITE DIFFERENT CODES THAT DOES SAME THINGS

DONE!!!!!!



App.jsx | Input.jsx | Login.jsx | index.html

src > components > App.jsx > App

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

var isLoggedIn=false;
const currentTime = new Date(2020,11,12,15).getHours();
console.log(currentTime);

function App() {
  return  <div className="container">
      {/* {isLoggedIn===true ? <h1>Hello</h1> : <Login />} */}
      {/* we can remove ===true */}
      {/* {isLoggedIn ? <h1>Hello</h1> : <Login />} */}
      {/* let's render a conditional message if time is greater tha
      { currentTime > 12 ? <h1>It is too late</h1> : null}
      {/* Below and operator does the same thing. In short but adva
      { currentTime > 12 && <h1>It is too late</h1>}
    </div>
}
export default App;
```

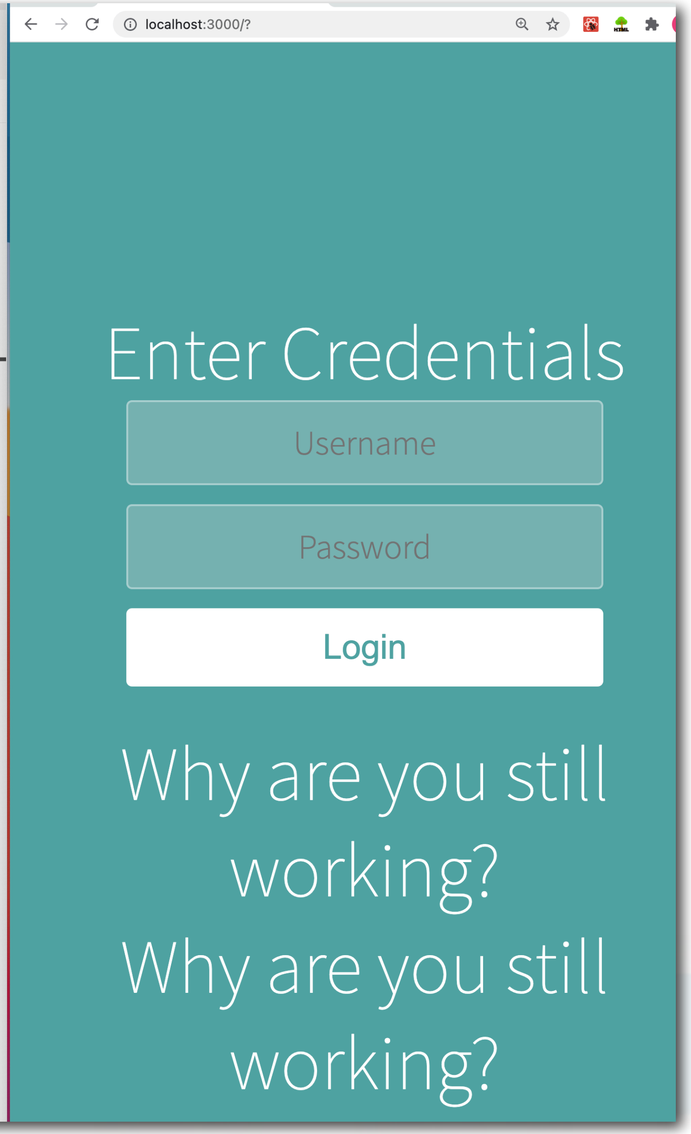
CONDITIONAL RENDERING USING && OPERATOR

JS App.jsx × JS Login.jsx

src > components > JS App.jsx > App

```
1   import React from "react";
2   import Login from "./Login";
3
4   //logic
5   var isLoggedIn=false;
6
7   // var currentTime = new Date().getHours();
8   var currentTime = new Date(2021,11,11,14).getHours();
9   console.log((currentTime));
10  function App() {
11    return (
12      <div className="container">
13        { isLoggedIn===true ? <h1>Hello! You successfully logged in!</h1> : <Login />}
14
15        {/* if current time > 12 then render message, otherwise don't do anything */}
16        {currentTime > 12 ? <h1>Why are you still working? </h1> : null }
17
18        {/* We can use && operator to do conditional rendering
19        If first is true, then render 2nd, otherwise don't render anything
20        Bolow means, if time>12 then render <h1>Why are you still working? </h1>, otherwise do
21        {currentTime > 12 && <h1>Why are you still working? </h1> }
22      </div>
23    );
24  }
25
26  export default App;
27
```

&& operators checks only the left side,
Since the right side has an expression
Which is always true
So if left side is true,
T&&T will render the right side
If left is false, F&& ANYTHING will not render anything



CONDITIONAL RENDERING HOMEWORK

```
//Challenge: Without moving the userIsRegistered variable,
//1. Show Login as the button text if userIsRegistered is true.
//Show Register as the button text if userIsRegistered is false.
//2. Only show the Confirm Password input if userIsRegistered is false.
//Don't show it if userIsRegistered is true.
```

Import conditional-rendering-practice

npm install react-scripts start

npm install

npm start

If user registered , show login screen

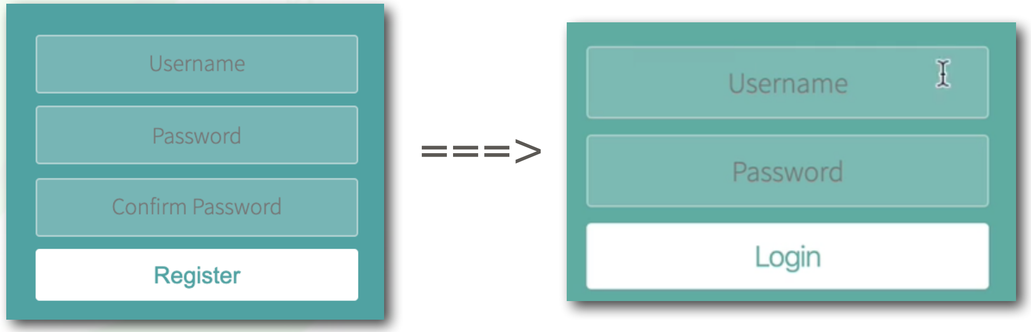
If not registered, show register screen

If on the register screen , show confirm password screen

If on the log in screen, do not show the confirm password screen

In Appjsx, when userIsRegistered = false we see username, password, confirm password

Change userIsRegistered=true, then you SHOULD SEE username, password, LogIn



CONDITIONAL RENDERING PRACTICE

//1. Show Login as the button text if userIsRegistered is true. Now it always shows Register button. Change the text “Register” that is inside App.jsx dependent of the value of userIsRegistered true or false. We will use props.

1. In App.jsx in add the props in the Form **<Form isRegistered={userIsRegistered}/>**

2. In Form.jsx, create props, use that props do conditional rendering of Register Text :

<button type="submit">{props.isRegistered ? "Login" : “Register"}</button>

NOW UI SHOULD RENDER LOGIN OR REGISTER BASED ON isRegister value

//2. Only show the Confirm Password input if userIsRegistered is false.//Don't show it if userIsRegistered is true.

Basically, when on Login screen, don’t render confirm password input, When in the Register screen render confirm password

3. In Form.jsx add this where placeholder=“Confirm Password is”

{props.isRegistered===false && (<input type="password" placeholder="Confirm Password" />) }

What this means is FIRST PART IS TRUE(props.isRegistered===false) is true render right side. DONT RENDER IT OTHERWISE

NOW THIS SHOULD WORK. CHANGE THE useIsRegistered true otr false to test

4. We can use ternary in the last part!.TEST IT.

```
{props.isRegistered === false ? (
  <input type="password" placeholder="Confirm Password" />
) : null}
```

5. FINALLY INSTEAD OF props.isRegistered===false CAN USE !props.isRegistered. TEST IT

DONE

App.jsx | Form.jsx | index.js

src > components > App.jsx > ...

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

var userIsRegistered = true;

function App() {
  return (
    <div className="container">
      <Form isRegistered={userIsRegistered}/>
    </div>
  );
}

export default App;
```

Form.jsx — conditional-rendering-practice

App.jsx | Form.jsx | index.js

src > components > Form.jsx > ...

```jsx
import React from "react";

function Form(props) {
  return (
    <form className="form">
      <input type="text" placeholder="Username" />
      <input type="password" placeholder="Password" />
      {/* {props.isRegistered===false && (<input type="password" placeho
      {/* BELOW IS SAME SLIGHTLY SHORTER */}
      {!props.isRegistered && (<input type="password" placeholder="Confi
      {/* BELOW IS HT ESECOND WAY */}
      {/* {props.isRegistered===false ? (<input type="password" placehol
      <button type="submit">{props.isRegistered ? "Login" : "Register"}<
    </form>
  );
}

export default Form;
```

REACT HOOKS -USESTATE

We will learn how to make our apps more interactive.

To do that we will learn the concept of STATE.

State is one of the fundamental concept in react

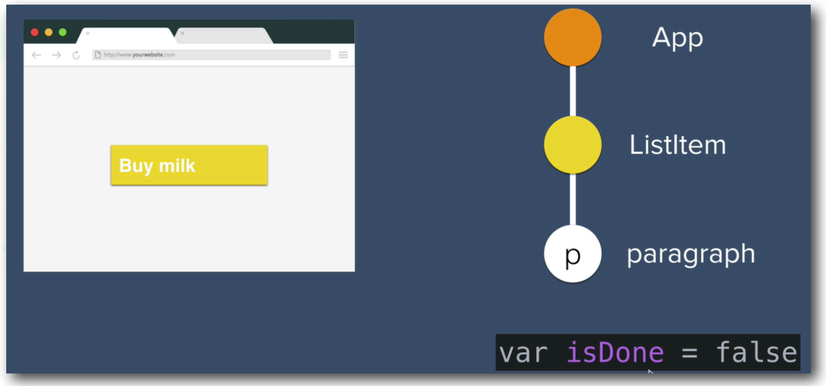
UI is actually a function of the state

When the state of a UI changes, we see different behavior of the app.

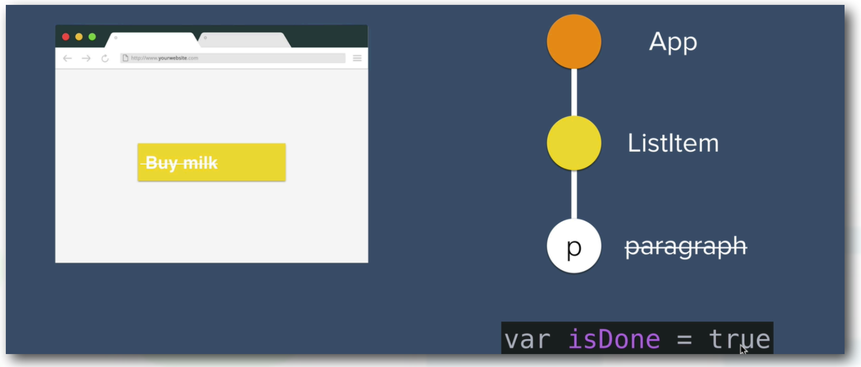
One of the way to change the state is, kept track of the condition of a variable

We will change the sate of the element depending on the isDone variables state.

In this practice we as we click on a button, we will increase the value on the number



USER CLICKS THEN CONDITION CHANGES



REACT HOOKS -USESTATE

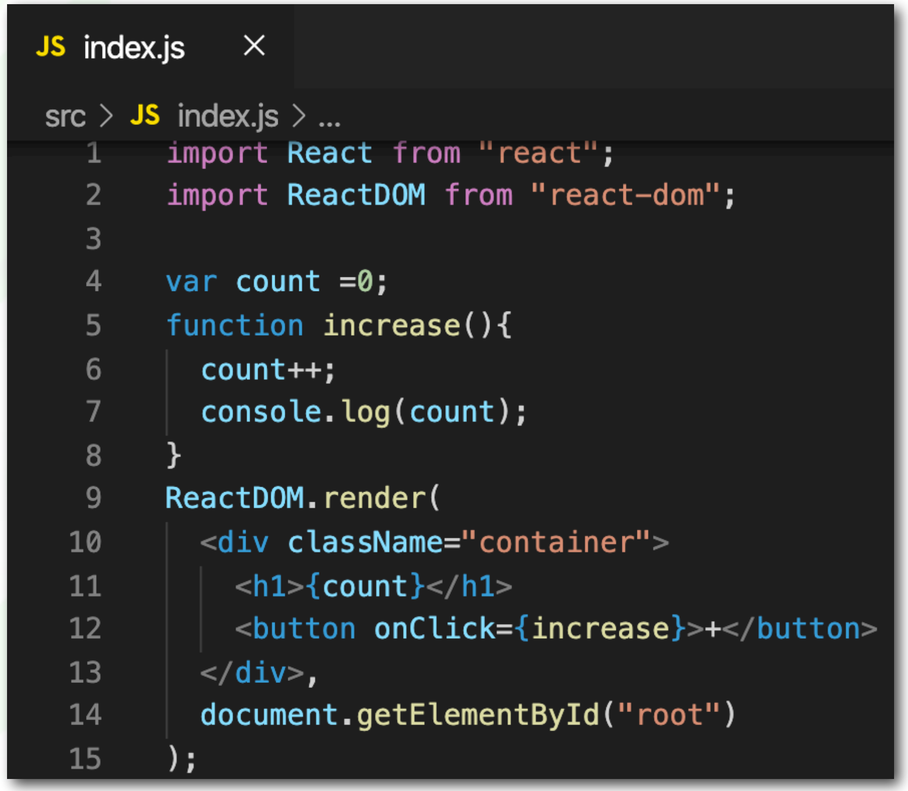
Import usestate-hook,npm install react-scripts start,npm install,npm start

Now I got everything inside my index.js and rending a single div with h1 and button

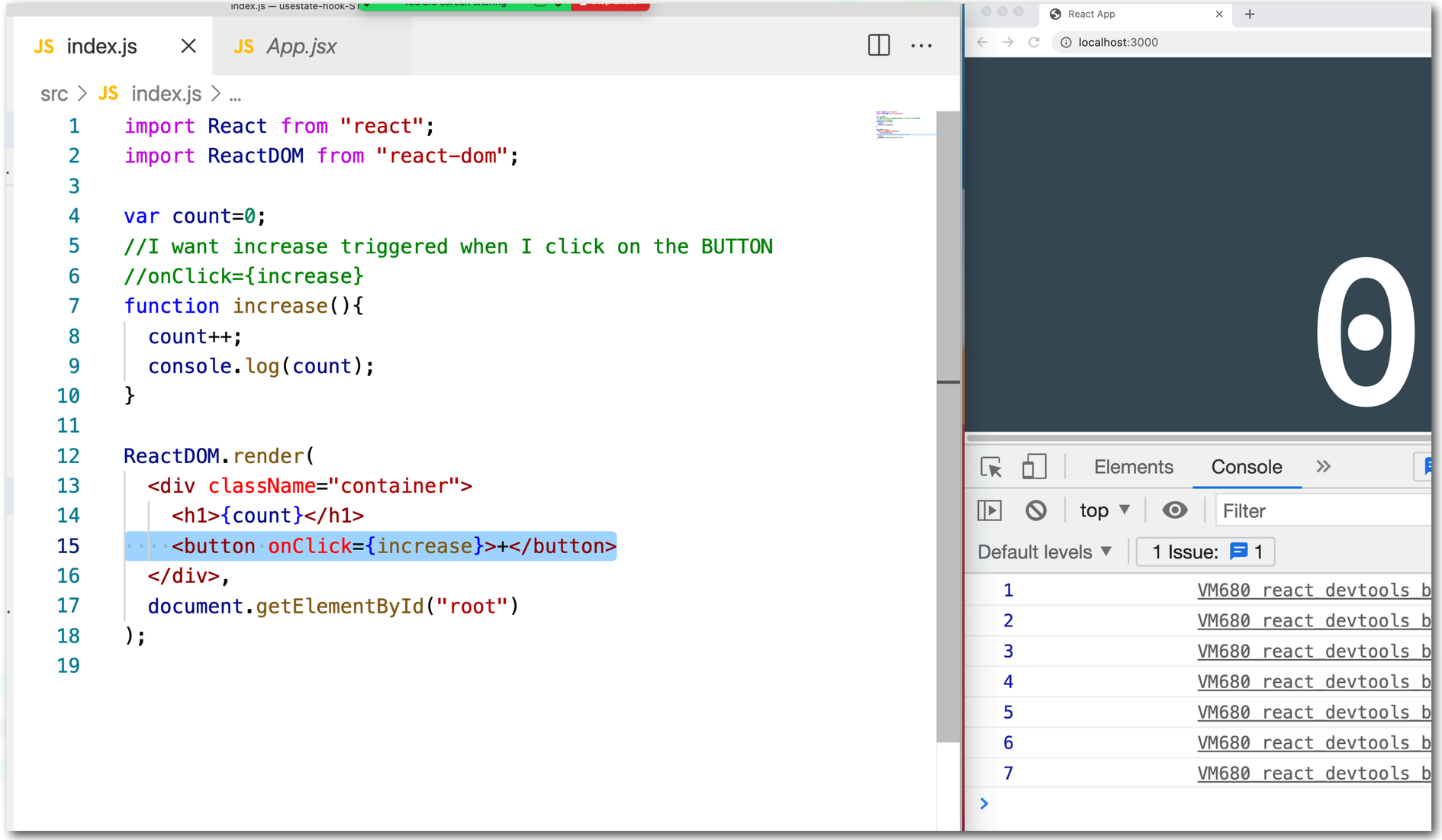
Our goal is the button should trigger and update the number. We want count change so create a count variable

We want to trigger + button. In HTML, button on-click attribute to trigger click action BUT in JS it is onClick.

So use **onClick** attribute for the button and assign a function called increase. Increate function will increate the value of count by 1 when we clock on the + button. onClick will trigger an action and update the element



WHEN WE CLICK ON THE BUTTON, COUNT IS BEING INCREASED BY ONE BUT THE CHANGE IS NOT DISPLAYED ON THE UI



REACT HOOKS -PUT DATA FROM INDEX.JS TO APP COMPONENT

In App.jsx.js,

1. Create App component and return the elements.

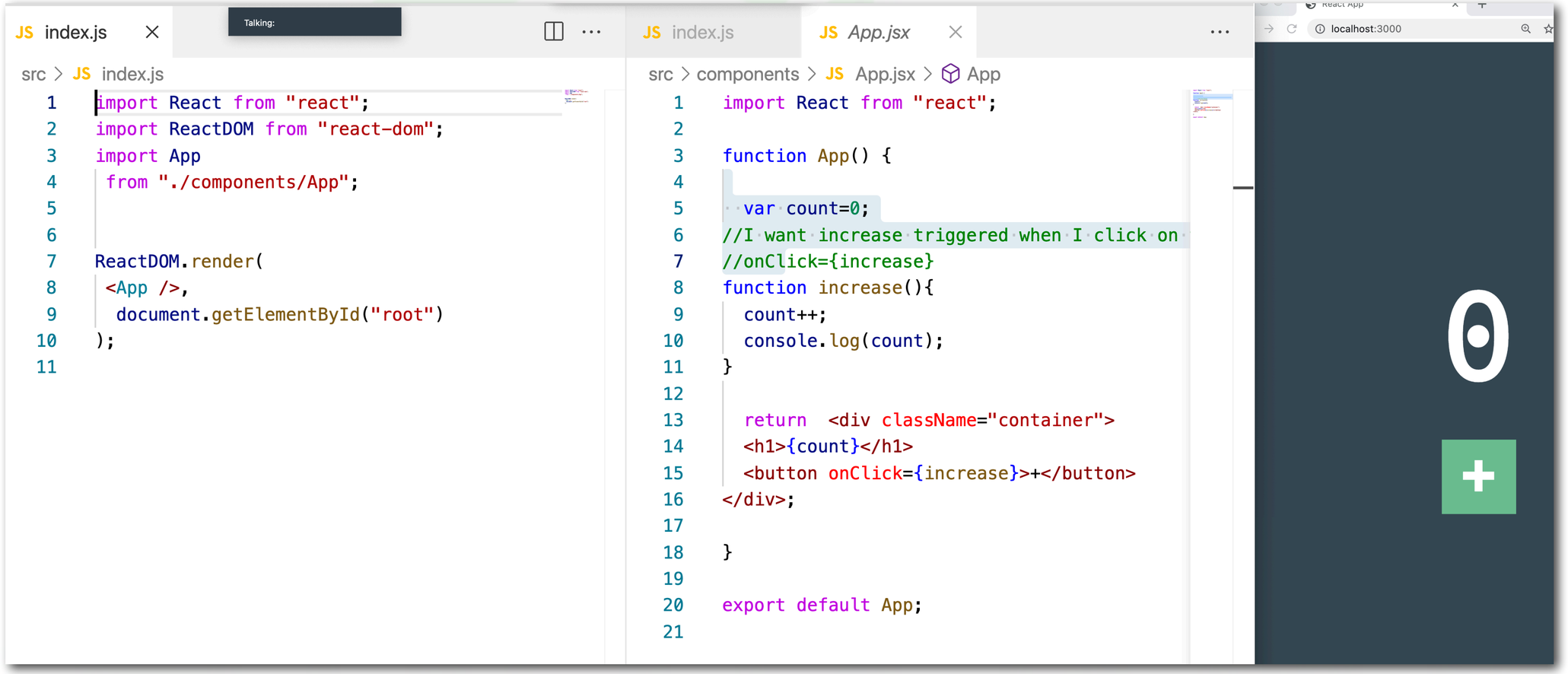
2. Create a variable count, initialize and use it in the h1 as js object

3. Use onClick attribute in the button and set it to a function called increase

4. Create increase function to increment the count

In index.js, call the App component

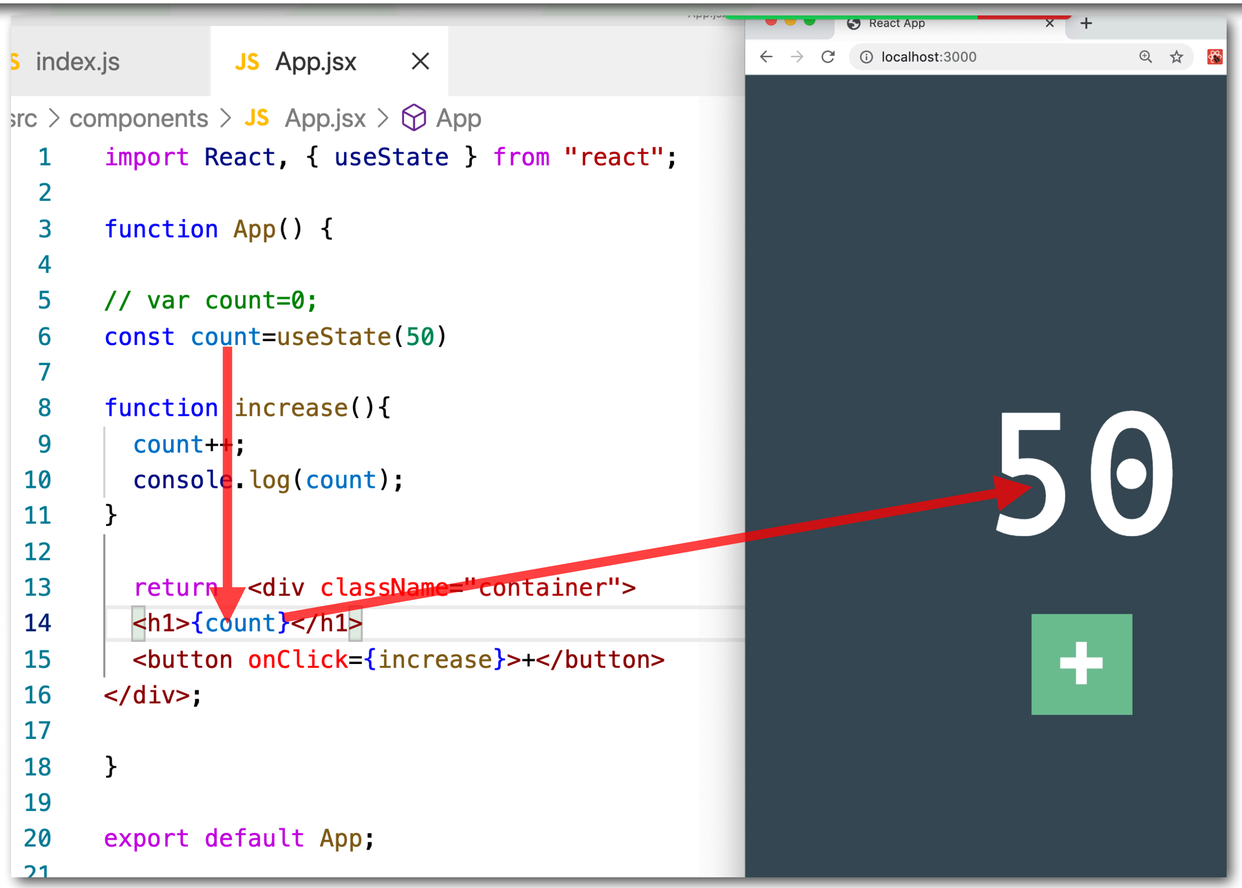
Now as we click the button, we should see the count is increasing on the console, But it is not rendering on the UI.



USE STATE HOOK

USE STATE IS USED TO ASSIGN AN INITIAL VALUE
AND UPDATE THAT INITIAL VALUE
THEN SHOW T IN THE UI
SECOND PARAMETER IS OPTIONAL

```
useState("initial value", "FUNCTION TO UPDATE THE INITIAL VALUE")
```



REACT HOOKS-DISPLAY UPDATED COUNT ON THE UI

1. In App.jsx, in the App function, Instead of var count = 0; we use useState hook

```
const state = useState(123);// 123 is starting state
console.log(state[0]);//return 123 on the console
```

2. import React, { useState } from "react";

3. To display on the <h1 element: <h1>{state[0]}</h1>. This means Whenever useState updated the ui automatically be updated

4. This is has coded. We will Destructure arrays and objects

5. To destrucre useState, let's map useState initial value to an array value:Below will render the initial value:

```
const [count] = useState(123);//assign 123 for count initial
<h1>{count}</h1>//Render count in the h1.UI SHOULD RENDER SAME
```

6. HOW DO WE INCREASE COUNT AS WE CLICK ON COUNT BUTTON???

7. **useState(initialValue,function)**

```
const [count,setCount] = useState(123);
function increase(){
  setCount(count+1)//increate the initial value one by one
}
<button onClick ={increase}>+</button>
```

6.

dex.js | JS App.jsx

components > JS App.jsx > ...

```
import React, { useState } from "react";

// let count=0;
// function increase(){
//   count++;
//   console.log(count);
// }

function App() {
  const [count,setCount] = useState(123);
  function increase(){
    setCount(count+1)
  }
  return  <div className="container">
  <h1>{count}</h1>
  <button onClick={increase}>+</button>
</div>;
}

export default App;
```

REACH HOOKS-USESTATE PRACTICE

YOUR TURN

Add a - sign and decrease the function

SOLUTION

ADDING BUTTON TO DECREASE THE COUNT AND DISPLAY UPDATED VALUE ON THE UI

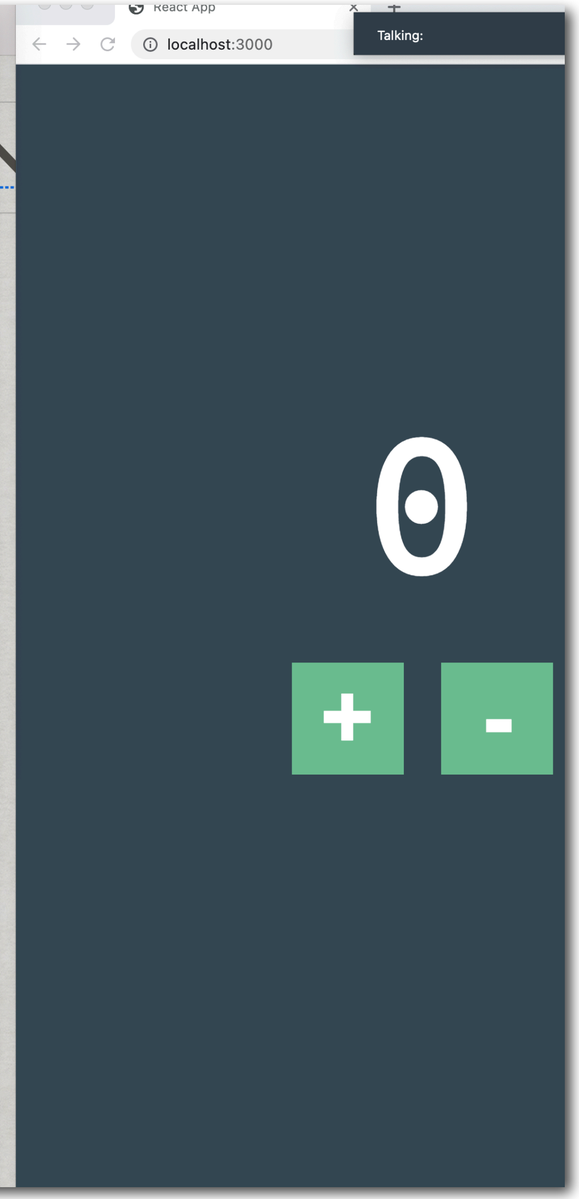
VARIABLE:
Initial value that comes from useState

FUNCTION:
2nd parameter is always a function
That can can be used to update
The first parameter, which is count in this case

App.jsx | index.html | styles.css

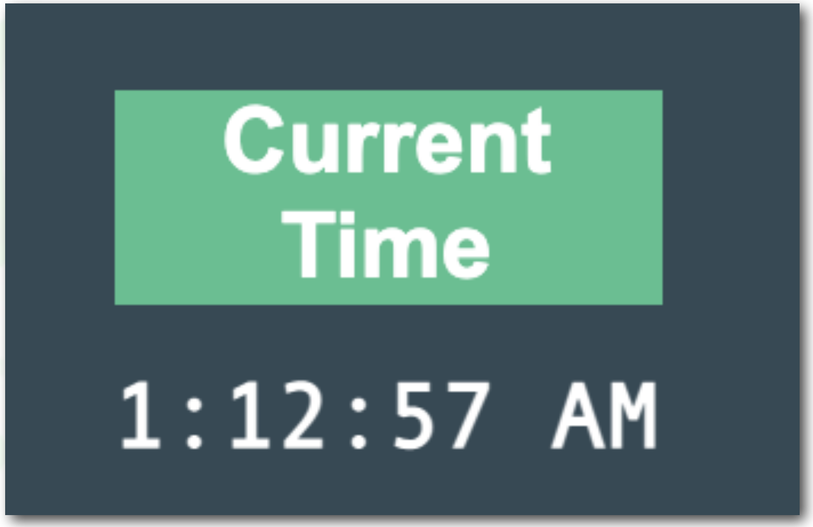
src > components > JS App.jsx > App

```
7    //count=0, setCount is the function that is use
8    //setCount will be used on the increase functio
9    const [count,setCount]=useState(0)
10
11   function increase(){
12     // count++;
13     setCount(count+1)
14   }
15
16   function decrease(){
17     setCount(count-1)
18   }
19
20     return  <div className="container">
21     <h1>{count}</h1>
22     <button onClick={increase}>+</button>
23     <button onClick={decrease}>-</button>
24   </div>;
25   }
26
27   export default App;
```



REACH HOOKS-USESTATE PRACTICE

RENDER THE CURRENT local TIME ON THE UI



Steps :

1. Create a button with text: Current Time
2. Style the button to match the style
3. Create an h1 and to get the current time.
4. Make sure to use create needed objects and the clock is ticking

NOTE : USE setInterval(function) TO SET RENDER THE UPDATE TIMES

REACT HOOKS-USESTATE PRACTICE

1. Create a button with text: Current Time

1. Style the button to match the style

```
<button style = {
  {fontSize:"30px",marginTop:"20%",width:"60%"}
  }>Current Time</button>
```

2. Create an h1 and to get the current time.

3.

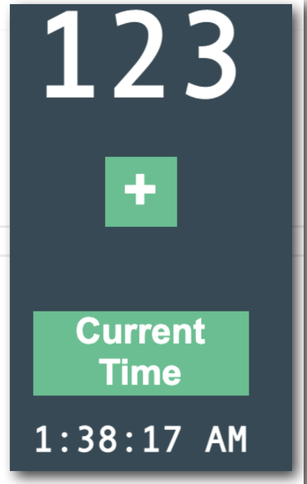
```
const now = new Date().toLocaleTimeString();
console.log(now);
<h1 style = {{fontSize:"30px"}}>{now}</h1>
```

```
//OR
const [time,setTime] = useState(new Date().toLocaleTimeString());
console.log(time);
```

```
//Creating a setInterval function that will update setTime every 1000 milisecond
//setInterval(setTime, 1000);//=> State Hook functions doesnt work in setInterval.
//That is why, we need a new function
setInterval(updateTime, 1000);
function updateTime(){
  const newTime = new Date().toLocaleTimeString();
  setTime(newTime);
}
<h1 style = {{fontSize:"30px"}}>{time}</h1>
```

components > JS App.jsx > ...

```
import React, { useState } from "react";

// let count=0;
// function increase(){
//   count++;
//   console.log(count);
// }
function App() {
  const [count,setCount] = useState(123);
  function increase(){
    setCount(count+1)
  }
  // const now = new Date().toLocaleTimeString();
  // console.log(now);
  // const [time,setTime] = useState(now);
  //OR
  const [time,setTime] = useState(new Date().toLocaleTimeString());
  console.log(time);
  //Creating a setInterval function that will update setTime every 1000 milisecond
  //setInterval(setTime, 1000);//=> State Hook functions doesnt work in setInterval.
  //That is why, we need a new function
  setInterval(updateTime, 1000);
  function updateTime(){
    const newTime = new Date().toLocaleTimeString();
    setTime(newTime);}

  return  <div className="container">
  <h1>{count}</h1>
  <button onClick={increase}>+</button>
  <br />
  <button style = {
    {fontSize:"30px",marginTop:"20%",width:"60%"}
    }>Current Time</button>
  <h1 style = {{fontSize:"30px"}}>{time}</h1>
</div>;}

export default App;
```



WORK FLOW

Created currentTime variable with initial value = current local time

We created setCurrentTime in the second parameter. This is used to update currentTime

We need to use another function (updateTime) to use setCurrentTime

```
const [currentTime,setCurrentTime] = useState(new Date().toLocaleTimeStr
```

We showed the currentTime time on the screen as JS object

```
<h1 style={{fontSize:"25px"}}> {currentTime} </h1>
```

We are creating updateTime to use setCurrentTime.

This will automatically pass newTime to currentTime

Whenever setCurrentTime is called, This assignment is executed: currentTime=newTime

```
function updateTime(){
  const newTime=new Date().toLocaleTimeString();
  setCurrentTime(newTime);
}
```

Using setInterval JS function to update updateTime function every 1000 millisecond=1second.

This will enable to see updated currentTime in every second

```
setInterval(updateTime,1000);
```

USESTATE HOOK PRACTICE-HOMEWORK

Import eleven-usestate-hook-practice

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

Or goal to to render the clock on the page and see the clock is running

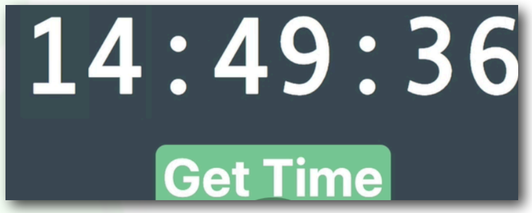
We break down the steps to learn the steps

TASK:

Part 1 : when we click on Get Time we should see the latest updated time

Part 2: we should always the updated time like a clock ticking

NOTE : USE setInterval(function) TO SET RENDER THE UPDATE TIMES




```
//Challenge:
//1. Given that you can get the current time
using:
let time = new Date().toLocaleTimeString();
console.log(time);
//Show the latest time in the <h1> when the Get
Time button
//is pressed.

//2. Given that you can get code to be called
every second
//using the setInterval method.
//Can you get the time in your <h1> to update
every second?

//e.g. uncomment the code below to see Hey
printed every second.
// function sayHi() {
//   console.log("Hey");
// }
// setInterval(sayHi, 1000);
```

USESTATE PRACTICE HOMEWORK

1. In App.jsx, creates constt now to get het current time

```
const now = new Date().toLocaleTimeString();
console.log(now);
```

1. In App.jsx, create a new function to destructure useState array that will hold the initial value of the time and a function that will update the time

```
  const [time, setTime]=useState(now);//starting value now. time=now
  console.log({time});
```

3. Use time, which has the update time in the <h1> to render it:

```
<h1>{time}</h1>
```

4. We will update the time when button clicked Use onClick and creat a function to update:

```
<button onClick={updateTime}>Get Time</button>
```

5. Create the updateTime function:

```
  function updateTime(){
    const newTime = new Date().toLocaleTimeString();
    setTime(newTime);
  }
```

5. Now whenever I click on the on Get Time button, I will get he new time. PART1 IS DONE. TEST

6. We will use setInterval function to display int he first line of the App function

```
  setInterval(updateTime, 1000);//MEANS update updateTime function every 1000 millisecond
```

7. It will create a new constant set to the current time and update our time variable that is used in h1 every single second! So we will see dynamic refreshing time

```
import React, { useState } from "react";

function App() {

  setInterval(updateTime, 1000);

  const now = new Date().toLocaleTimeString();
  // a(now);

  const [time, setTime] = useState(now);
  // console.log({time});

  function updateTime(){
    const newTime = new Date().toLocaleTimeString();
    setTime(newTime);
  }

  return (
    <div className="container">
      {/* <h1>TIME</h1> */}
      <h1>{time}</h1>
      <button onClick={updateTime}>Get Time</button>
    </div>
  );
}

export default App;
```

EVENT HANDLING IN REACT

Import event-handling-in-react,

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

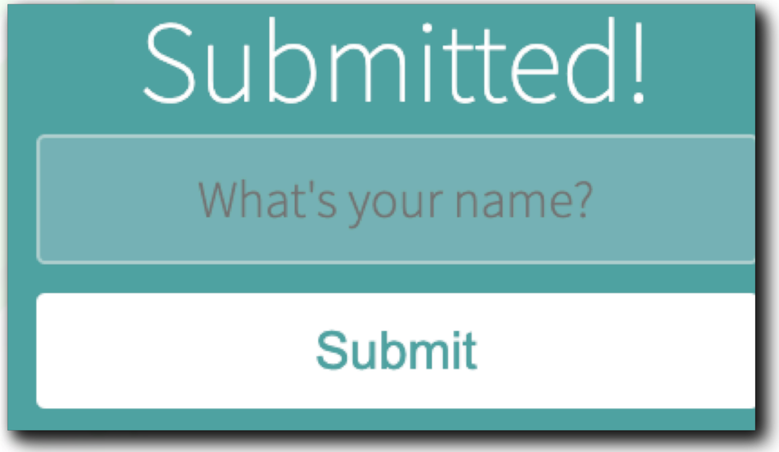
We will learn handling event in react

When the mouse is in the submit button, it turns black, when we move away, it turns white.

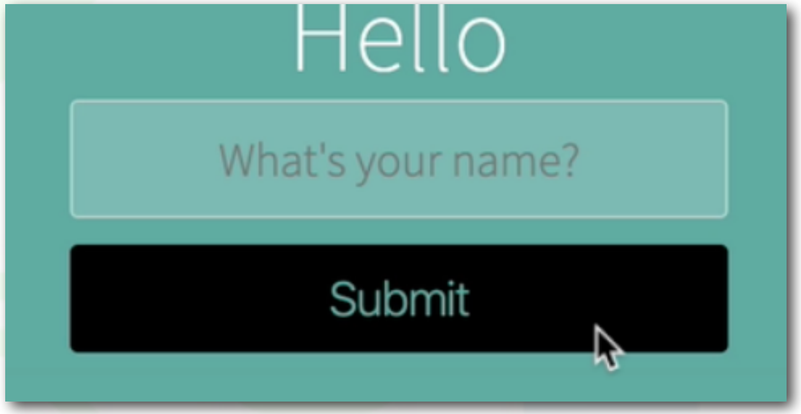
We will do it by handling this mouse over event.

How do you detect when user interact with the app or a component?

How to use react render different things after these user interaction?



We ll use conditional rendering, inline styling, mouse events, managing state in this lesson.



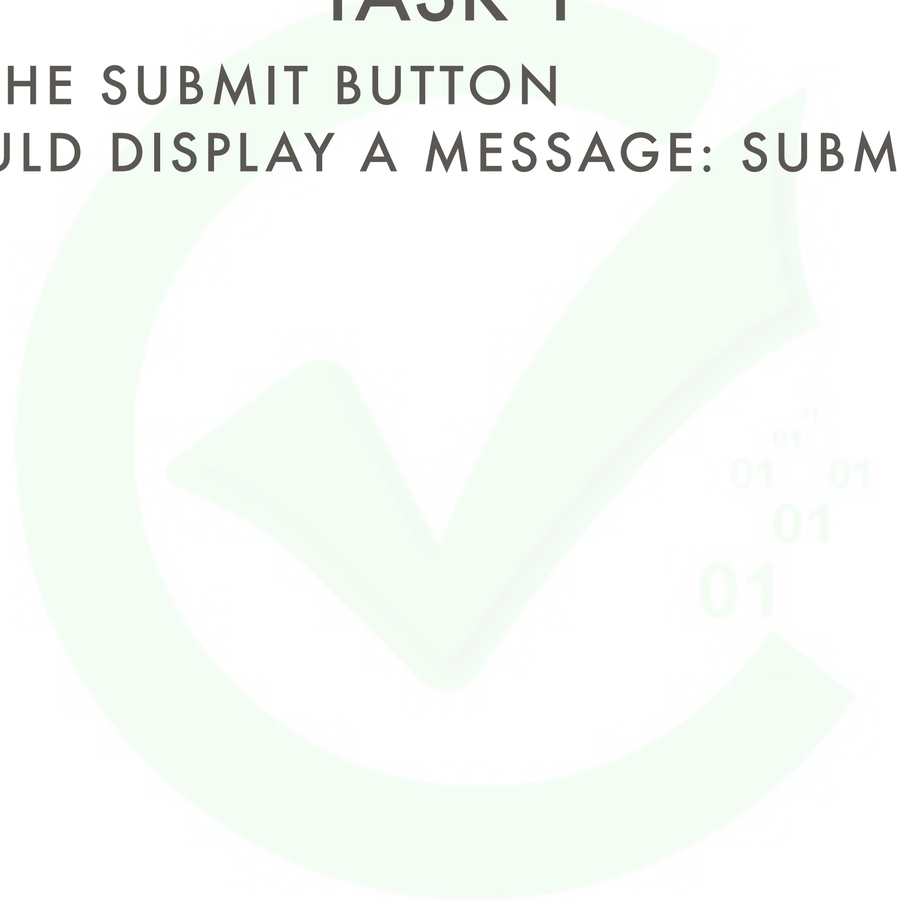
CHECK OUT:

https://reactjs.org/docs/handling-events.html

https://www.w3schools.com/js/js_events.asp

TASK 1

WHEN USER CLICK ON THE SUBMIT BUTTON
THEN H1 ELEMENT SHOULD DISPLAY A MESSAGE: SUBMITTED!



EVENT HANDLING IN REACT

In App,jsx, we just have an h1, input type text, placeholder that is asking for the name to type in, and submit button

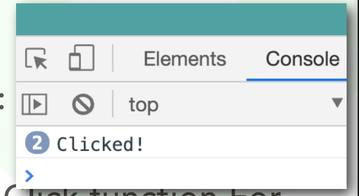
1. Let's start with onClick on the button:Add onClick on the button

```
<button onClick={handleClick}>Submit</button>
```

2.In App function, create the handleClick function to tell the handleClick what to do:

```
function handleClick(){console.log("Clicked!") }
```

Test button. When you click the button, you should now see the log:



3. Now that we can detect button click, we can write code in handleClick function.For example we can change the heading when button is clicked.

```
 <h1> {headingText} </h1>
```

4. Use useState hook, because we have initial value, which is Hello, and a new value, which is "Submitted"

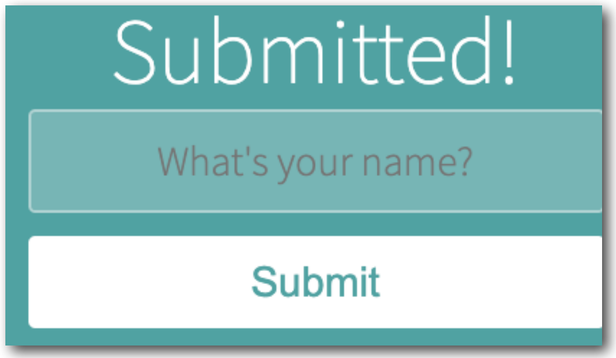
```
const [headingText,setHeadingText]=useState("Hello");
```

5. Add setHeadingText("Submitted" function in the handleClick() function to change the value after click

```
function handleClick(){setHeadingText("Submitted!");
  // To render the text conditionally we can add ternary in the handle click function:
  // headingText === "Hello"?setHeadingText('Clicked'):setHeadingText('Hello')}
```

App.jsx

```jsx
import React,{useState} from "react";

function App() {
  const [headingText,setHeadingText]=useState("Hello");
  function handleClick(){setHeadingText("Submitted!");}
  return (
    <div className="container">
      <h1> {headingText} </h1>
      <input type="text" placeholder="What's your name?" />
      <button onClick={handleClick}>Submit</button>
    </div>
  );
}

export default App;
```



```
//TASK1: When user clicks on Submit button,
//Then Replace Hello with Submitted!
```

FLOW

Created headingText variable = Heading

Created setHeadingText to update the headingText variable

```
const [headingText,setHeadingText]=useState("Heading");
```

Show headingText on the h1 element

```
<h1>{headingText}</h1>
```

Used onClick button handler and passed handleClick function

This means, when Submit button is clicked, handleClick will run because of onClick handler

```
<button onClick={handleClick}>Submit</button>
```

Writing our handleClick to tell what would happen when the button Is clicked.

Here when the button is clicked, we are updating headingText = Submitting

We are able to update the headingText="Submitted" because of the useState hook

```
function handleClick(){
  //Update the heading when Sub
  //console.log("Testing if har
  setHeadingText("Submitted!")/
}
```

TASK 2

WHEN USER MOVES OVER THE BUTTON
THEN COLOR WILL BE BLACK
WHEN USER MOVES AWAY FROM THE BUTTON
THEN COLOR WILL BE BACK TO ORIGINAL

EVENT HANDLING IN REACT

https://www.w3schools.com/tags/ref_eventattributes.asp has the list of events

onmouseover will detect when we hover over the element, onmouseout that will detect when we move away from the element

NOTE : javascript attributes are camelCase . HTML onclick,onmouseover . JS onClick onMouseOver.

1.Remember we can change the appearance of html element using style attribute.style = {{backgroundColor:"black"}}

```
<button style = {{backgroundColor:"black"}}
```

This'll make the button always black. We need to render background color black or red based on a condition, mouse is over or mouse is out, etc.

2. First use onMouseOver to turn the color to black when we hover over the button. Add button:

```
onMouseOver = {handleMouseOver}
```

3. And create the function to tell what this handleMouseOver will do :log something to test

```
function handleMouseOver(){console.log("Mouse Over");}
```

4. Next we will change the color of the button only when we handleMouseOver Function is called.

```
//isMouseOver initial value = false.
//setMouseOver will update the initial value
  const [isMouseOver,setMouseOver]=useState(false);
```

5. Use setMouseOver function inside the handleMouseOver function to set the new value true:

```
function handleMouseOver(){setMouseOver(true);}
```

6. We will change the button background color black when isMouseOver= true. Otherwise change it to white. Use Ternary.

```
<button style = {{backgroundColor:isMouseOver?"black":"white"}}
```

7. When we move mouse over button is still black. So Add on onMouseOut={handleMouseOut} to switch it back:

```
onMouseOut={handleMouseOut}
```

8. Create the handleMouseOut function to set the mouseOver false when mouse is moved out of the button

```
function handleMouseOut(){setMouseOver(false);}
```

App.jsx

src > components > App.jsx > App > handleMouseOut

```jsx
import React,{useState} from "react";

function App() {
//create the state for headingText, with starting = Hello
//setHeadingText will be used to change the text
const [headingText,setHeadingText]=useState("Hello");
//starting value is false cause when we load the site we are not jovered
const [isMouseOver,setMouseOver]=useState(false);

function handleClick(){
  // console.log("Clicked!")
  //change the text to Submitted when we click on the Submit buttum.
setHeadingText("Submitted")
}
function handleMouseOver(){
  //This function will trigger whenever we hover over the button
  // console.log("Mouse Over")
  // Create the variable to hold the state to change the background color
  setMouseOver(true);
}
function handleMouseOut(){
  setMouseOver(false);
}
  return (
    <div className="container">
      {/* <h1>Hello</h1> */}
      <h1>{headingText}</h1>
      <input type="text" placeholder="What's your name?" />
      <button
      style = {{backgroundColor: isMouseOver ? "black":"white"}}
      onMouseOver = {handleMouseOver}
      onMouseOut={handleMouseOut}
      onClick={handleClick}>Submit</button>
    </div>
  );
}

export default App;
```

FLOW

onMouseOver and onMouseOut is detecting when user moves the mouse over or out of the button

```
<button
onClick={handleClick}
onMouseOver={handleMouseOver}
onMouseOut={handleMouseOut}
style={{background:isMouseOver?"black":"white"}}>Subm
```

```
//TASK2: When user moves the mouse
over the Submit button, then change
the background color to black
When user moves the mouse out the
Submit button, then change the
background color to white
```

isMouseOver=false. Thill will be used to render the background color of button conditionally

```
const [isMouseOver,setMouseOver]=useState(false);
```

handleMouseOver.Out functions triggers when user moves the mouse over or out
This will update the value of isMouseOver to true or false

```
function handleMouseOver(){
  //console.log("Mouse Over")//Test
  setMouseOver(true)// => isMouseOv
}
```

```
function handleMouseOut(){
  setMouseOver(false)//=> i
}
```

backGroundColor of button will be black or white depending on isMouseOver Is true or false

```
onMouseOut={handleMouseOut}
style={{background:isMouseOver?"black":"white"}}>S
</div>
```

EVENT HANDLING IN REACT

Now when I move over the button, it turns black, move away it turns white.

And when you click on the submit button hello changes to Submitted

We used conditional rendering using ternary operator

In line styles. Using event listener like onMouseOver onMOuseOut

And setting and using the state!!!

REACT FORMS

Import the new thirteen-event-handlong-in-react, npm install react-scripts start,npm install,npm start

Will learn complex events, like handling form event.

Creating forms are very common in development like log in, registration, contact etc.

We will again set state and use state and **onChange** event for an input:

How we detect the change in the input?

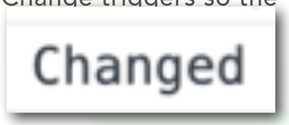
1.Add onChange event in the input and pass a function handleChange:

```
onChange={handleChange}
```

2.Create handleChange function and log to Test

```
function handleChange(){
console.log("Changed");}
```

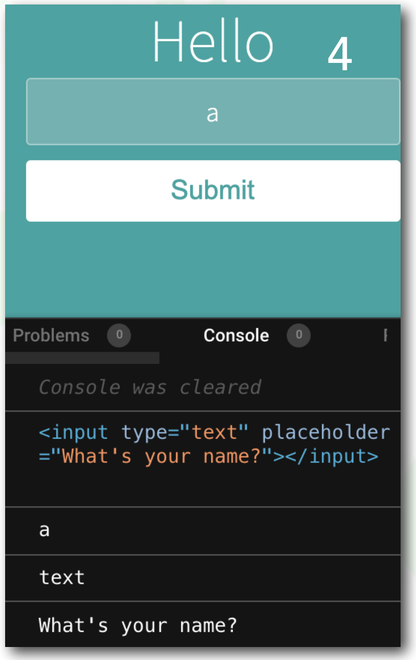
What this does is whenever we enter a data change happens onChange triggers so the handleChange triggers.



3. When input element triggers it passes an object to handleChange. We can catch that object by **event.** Then we can log various thing related to that event:

4. Add event to handleChange function, And use that event to use to log values during input change:

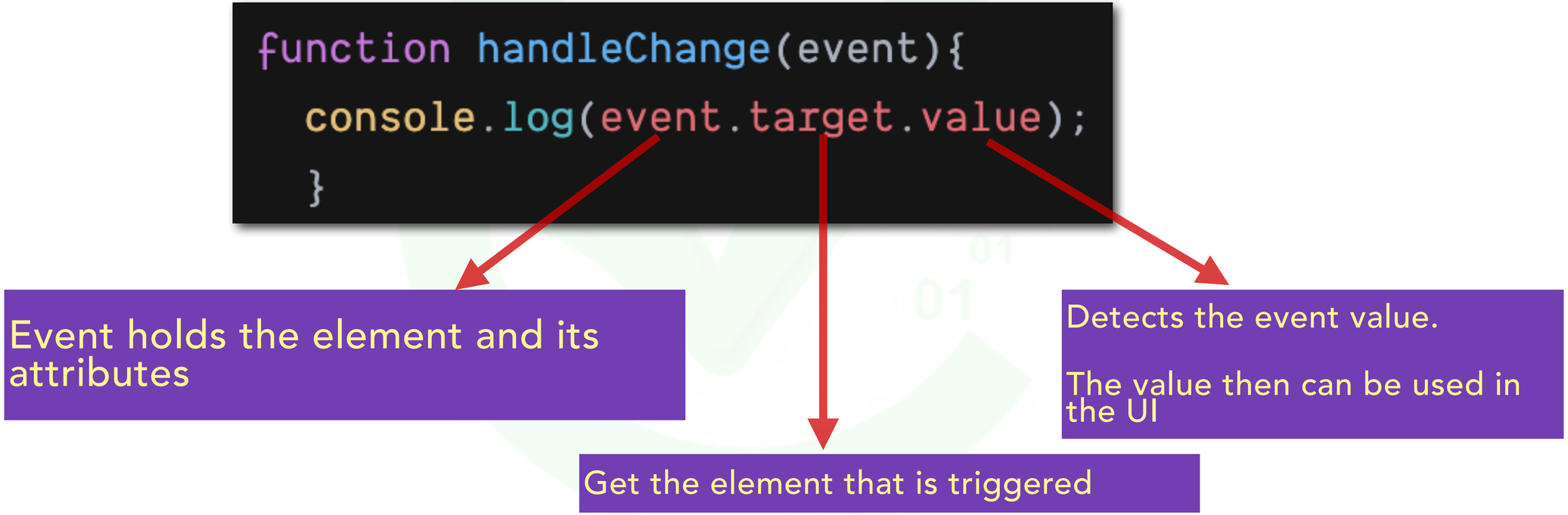
```
function handleChange(event){
    console.log(event.target)
    console.log(event.target.value);
    console.log(event.target.type);
    console.log(event.target.placeholder); }
```



App.jsx

```jsx
import React from "react";

function App() {

  function handleChange(event){
    console.log(event.target)
    console.log(event.target.value);
    console.log(event.target.type);
    console.log(event.target.placeholder);}

  return (
    <div className="container">
      <h1>Hello </h1>
      <input
      onChange={handleChange}
      type="text"
      placeholder="What's your name?" />
      <button>Submit</button>
    </div>
  );
}
export default App;
```

REACT FORMS


function handleChange(event){
console.log(event.target.value);
}
Event holds the element and its attributes
Get the element that is triggered
Detects the event value.
The value then can be used in the UI

REACT FORMS

Now we are getting the data using onChange handler, but how can we use this data on the screen?

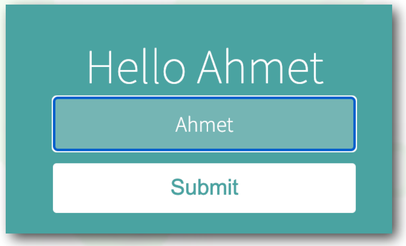
For example as we enter our name in the box, we want to see it on the heading. Again we can use state hook:

1. Create const with initial value is empty string.

```
const [name,setName] = useState("");
```

2. Use setName function to update the name as we update the input

```
function handleChange(event){
 setName(event.target.value);}
```



3. In input, Then pass the name inside the h1 to display the entered updated new value

```
<h1>Hello {name}</h1>
```

So far this works, But Important note about the forms, In html , elements are responsible to handle their own state. We can use value attiribute to set the value with the event.target.value, which is the name variable:

4. In put, So add value property and set the the name(which is event.target.value), so input can always value

```
value={name}
```

This concept is called controlled component https://reactjs.org/docs/forms.html#controlled-components

Again we should update the value attribute with the variable that comes from our useState hooks, which is the updated value

App.jsx

```jsx
import React,{useState} from "react";

function App() {
  const [name,setName] = useState("");

  function handleChange(event){
    console.log(event.target.value);
    setName(event.target.value);}

  return (
    <div className="container">
      <h1>Hello {name}</h1>
      <input
      onChange={handleChange}
      type="text"
      placeholder="What's your name?"
      value={name}/>
      <button>Submit</button>
    </div>
  );
}
export default App;
```

```
Task: When user types in
input,
Then show the text on heading
```

FLOW

Creating name="" and setName

```
const [name,setName]=useState("");
//1. name="" 2. setName will update name
```

Showing name on the screen on h1 element

```
<h1> Hello {name} </h1>
```

Hello

onChange handler, we detect the change. handleChange will change the behavior when there is any change.

```
<input
onChange={handleChange}
```

Updating name using setName function

Catching the input using event, And passing it to the name variable(name=event.target.value)

```
function handleChange(event){
  // console.log(event.target.v
  setName(event.target.value);
}
```

Entered text= javascript

event.target.value=javascript

name=javascript

Hello {name} = Hello javascript

We see Hello javascript on the heading

REACT FORMS

What if I want to type a name , and I don't want to see the instant update in the heading. I only want to see the entered value after I click on the submit button.

HINT1:Detect event in the submit button

HINT2:Need another variable to tract the state

HINT3:And play with the code until it works

1. In button, Add onClick listener on the submit button(We did similar before have)

```
<button onClick={handleClick}>Submit</button>
```

2. Create handleClick function: This will get the input value from event.target.value(name) that came from input and put in the h1. Let's use setHeading to update the name:

```
function handleClick(){
      setHeading(name);}
```

3. We need to create useState hook to hold the initial value and set the new heading value:

```
const [headingText,setHeading] = useState("");
```

4. In h1, add the new heading text . It will only display in the heading after user clicks on submit button. Only after When we Click button, we shiould take the value from event.target.value and put it it in the h1

```
 <h1>Hello {name} {headingText}</h1>
```

App.jsx

```jsx
import React,{useState} from "react";

function App() {
  const [name,setName] = useState("");
  const [headingText,setHeading] = useState("");
  function handleChange(event){
    console.log(event.target.value);
    setName(event.target.value);}
  function handleClick(){
      setHeading(name);}
  return (
    <div className="container">
      <h1>Hello {name} {headingText}</h1>
      <input
      onChange={handleChange}
      type="text"
      placeholder="What's your name?"
      value={name}/>
      <button onClick={handleClick}>Submit</button>
    </div>
  );
}
export default App;
```

Task: When user types in
input AND CLICK SUBMIT BUTTON
Then show the text on heading

FLOW

Creating headingText and updating it using setHeadingText

```
const [headingText,setHeadingText]=useState("");
```

Showing heading text on the h1 element on the UI

```
<h1> Hi {headingText} </h1>
```

Since we want to trigger headingText after button click, we used onClick in the button

```
<button onClick={handleClick}>Submit</button>
```

As soon as button is clicked, setHeadingText function will make the update headingText=name

```
function handleClick(){
  setHeadingText(name)
}
```

REACT FORMS

Lastly, normally we put inputs and submit button, we use html form component

```
<form>
        <input
          onChange={handleChange}
          type="text"
          placeholder="What's your name?"
          value={name}/>
        <button onClick={handleClick}>Submit</button>
</form>
```

2. In form, add onSubmit handler pass handleClick, add type="submit" to the button, Then delete onClick handler from button cause we no longer need it. This will refresh the page under submit button

```
   <form onSubmit={handleClick}>
        <input
          onChange={handleChange}
          type="text"
          placeholder="What's your name?"
          value={name}/>
        <button type="submit">Submit</button>
      </form>
```

3.In handleClick function, Add event.preventDefault(); Then we can use event handler and get the event to prevent the default behavior. Which is when click the button refreshing automactivcallt

```
function handleClick(event) {
    setHeading(name);
    event.preventDefault();}
```

Now after I click on submit, event handler prevents the default behaviors. DONE

App.jsx

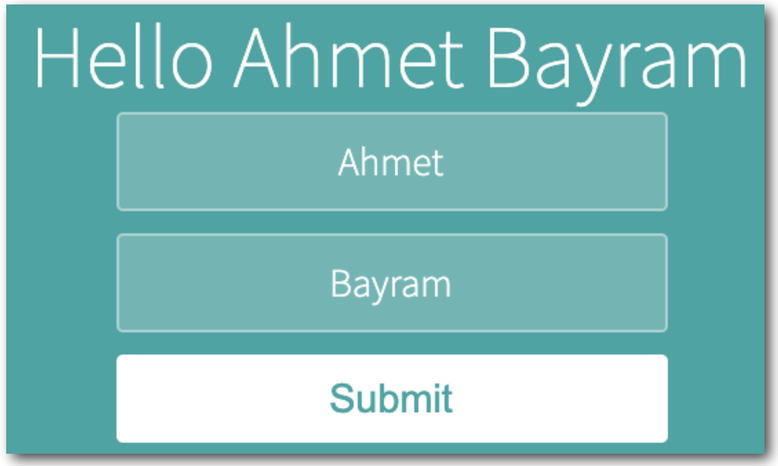
```
1  import React, { useState } from "react";
2
3  function App() {
4    const [name, setName] = useState("");
5    const [headingText, setHeading] = useState("");
6    function handleChange(event) {
7      console.log(event.target.value);
8      setName(event.target.value);
9    }
10   function handleClick(event) {
11     setHeading(name);
12     event.preventDefault();}
13   return (
14     <div className="container">
15       <h1>
16         Hello {name} {headingText}
17       </h1>
18       <form onSubmit={handleClick}>
19         <input
20           onChange={handleChange}
21           type="text"
22           placeholder="What's your name?"
23           value={name}/>
24         <button type="submit">Submit</button>
25       </form>
26     </div>
27   );
28 }
29 export default App;
```

CHANGING COMPLEX STATE

Import changing-complex-state npm install react-scripts

start, npm start

We will learn handling events and managing more complex state.
We will retrieve the value of an object that is working together with another object.



Managing the state of javascript objects WHERE WE HAVE TO FIRST GET THE PREVIOUS VALUE OF THE JS OBJECT

For instance when I type first name then last name(they are different object), the I would see Hello first name last name in the heading

SOLUTION 1

Adding onChange handler to input to detect the change
In the input element

```
<form>
  <input
  onChange={updateFName}
  name="fName"
  placeholder="First Name" />
```

Using useState to initialize fName="", and
create setFName to update fName object with
the user entered data(event.target.value)

```
//displaying fName on the h1 element
const [fName,setFName] = useState("");
function updateFName(event){
  setFName(event.target.value)
}
```

Use that fName variable on the h1 element to display

```
<h1>Hello {fName}</h1>
```

Repeating same steps to display last name
I have to use different variable

```
<input
onChange={updateLName}
name="lName"
placeholder="Last Name" />
```

```
//displaying lName on the h1 element
const [lName,setLName] = useState("");
function updateLName(event){
  setLName(event.target.value)
}
```

```
<h1>Hello {fName} {lName}</h1>
```

SOLUTION 2

```
<input
onChange={handleChange}
name="fName"
placeholder="First Name"
value={fullName.fName}
/>
<input
onChange={handleChange}
name="lName"
placeholder="Last Name"
value={fullName.lName}
/>
```

```
const [fullName,setFullName]=useState({fName:"",lName:""})
```

```
function handleChange(event){
  // const newValue = event.target.value
  // // We need this to understand where the change is happ
  // const inputName=event.target.name
  // //prevValue hold fName and lName values
  // COMPOSING THE ABOVE CODE
  // And updating the setFullName function variables
  const {value,name} =event.target
  // replace newValue for value
  // replace inputName for name

  setFullName(prevValue =>{
    console.log(prevValue);//{fName: 'aby', lName: 'gh'}
    console.log(prevValue.fName)//aby
    console.log(prevValue.lName)//g

    if(name==='fName'){
      // if there is a chagne in fName
      // return fName with latest newValue
      // return lName with the oldest previous value
      return {
        fName: value,
        lName: prevValue.lName
      }
    }
    else if(name==='lName'){
      return {
        fName:prevValue.fName,
        lName: value
      }
    }
  })
}
```

React Hook functions will hold onto the previous values by default
setFullName has prevValue variable which holds
The values inside the fName and lName inputs
We use those values to show on the UI.
If we don't use, react will delete the input
that has no latest change

STEP 1

Created fullName to hold and update user entered values

```
const [fullName,setFullName] = useState({fName:"",lName:""});
console.log(fullName.fName);
console.log(fullName.lName);
```

Render in the h1

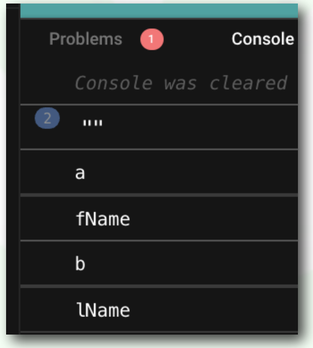
```
<h1>Hello {fullName.fName} {fullName.lName}</h1>
```

To detect the change in the input, we used onChange property

```
<input onChange ={handleChange} value={fullName.fName}/>
<input onChange ={handleChange} value={fullName.lName}/>
```

handleChange is used to hold the event object and use that trigger setFullName function

```
function handleChange(event){
  const newValue = event.target.value;
   console.log(newValue);
  const inputName= event.target.name
   console.log(inputName);}
```



We are only getting the last value, Not the entire values

I need to use setFullName hold on the to the previous value

```
function handleChange(event){
  const newValue = event.target.value;
     console.log(newValue);
  const inputName= event.target.name
     console.log(inputName);
     setFullName(newValue);}
```

STEP2

We need to know which input has the change. We can use event.target.name to check it

```
LOGIC: IS THERE IS A CHANGE IN fName THEN SET THE fName to the newValue BUT
KEEP lName TO ITS  PREVIOUS EXISTING VALUE.
ELSE IF THERE IS A CHANGE IN THE lName, THEN SET lName TO THE NEW VALUE BUT
KEEP THE FLABE TO ITS PREVIOUS EXISTING VALUE.ADD THIS LOGIC IN THE
setFullName function!!!
YOU MIGHT THING IT IS AS SIMPLE AS WRITING IF ELSE LIKE THIS
THIS AUTOMATICALLY RENDERS THE LATEST UPDATED INPUT . IT DELETED THE PREVIOUS
VALUE
```

```
function handleChange(event){
  const newValue = event.target.value;
  const inputName= event.target.name
  // console.log(newValue);
  // console.log(inputName);
  if(inputName==="fName"){
    setFullName({fName: newValue});
  }else if(inputName=="lName"){
    setFullName({lName:newValue});
  }}NOT WORKING AS EXPECTED
```

Next

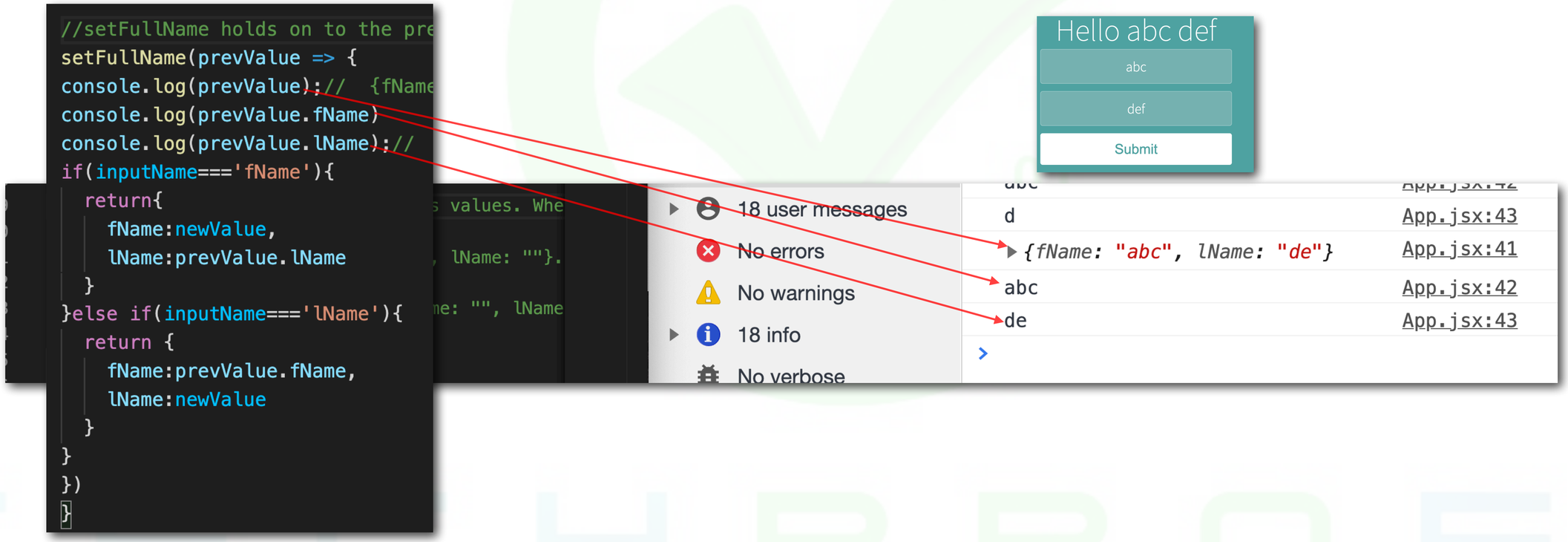
STEP3

```
function handleChange(event){
  const newValue = event.target.value;
  const inputName= event.target.name
setFullName(prevValue=>{//When this function called,
  // we can access the previous value.
  // Let's name it prevValue
  console.log(prevValue);
})}
```

WE MUST USE PREVIOUS VALUE OF THE INPUTS

setFullName is a function of useState. It remembers the previous values of the inputs

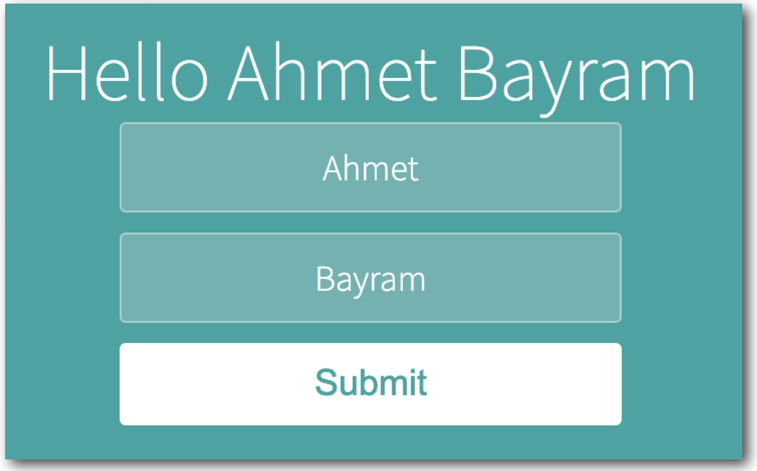
We can use this concept to hold and update the date with previous value if there is no change:



STEP 4

3. We will use if statement to render first name if there is a change in first name, and render last name if there is a change in the last name using event.target.name property by checking the the values of fName and lName

```
LOGIC: IS THERE IS A CHANGE IN fName THEN SET THE fName to the newValue BUT KEEP lName TO ITS  PREVIOUS
EXISTING VALUE.
ELSE IF THERE IS A CHANGE IN THE lName, THEN SET lName TO THE NEW VALUE BUT KEEP THE FLABE TO ITS PREVIOUS
EXISTING VALUE.ADD THIS LOGIC IN THE setFullName function!!!
setFullName can take a function as input. We will write our function inside the setFullName function
```




```
function handleChange(event){
  const newValue = event.target.value;
  const inputName= event.target.name
  setFullName(prevValue=>{

  if (inputName==="fName"){
    return {
      fName: newValue,
      lName: prevValue.lName
    };//IF THE ELSE, WE UPDATE LNAME AS NEW VALUE, AND FNAME WILL NOT CHANGE
  }else if(inputName==="lName"){
    return {
      fName: prevValue.fName,
      lName: newValue
    };
  }})}//DONE THIS NOW WORKS AS EXPECTED TEST
```

prevValue=> React remembers the value of the previous objects. Using arrow function, we can hold that value any use in the new codes! So we used that remembered value depend on this previous value. SO prevValue remembers both fName and lName values because setFullName function has these two javascript objects

WORKFLOW

Creating fullName to show fName and lName

```
const [fullName, setFullName]=useState({
  fName:"",
  lName:""
});
```

Showing fName and lName

```
<h1>Hello {fullName.fName} {fullName.lName}</h1>
```

on Change to detect the change in both inputs

```
<input
onChange={handleChange}
name="fName"
placeholder="First Name"
value={fullName.fName} />
<input
onChange={handleChange}
name="lName"
placeholder="Last Name"
value={fullName.lName}
/>
```

Coding inside handleChange to add the logic

```
function handleChange(event){
  //console.log(event.target.value);//TE
  //When there is a change in the fName
  const newValue=event.target.value;//Re
  console.log(newValue)   any
  const inputName=event.target.name;
  console.log(inputName);//returns fName
```

Updating fName and lName objects.
To prevent some default behaviors, we used prevValue.
If the change is in fName, the easing fName with newValue
And lName = prevValue(if I don't use it, lName will be deleted)

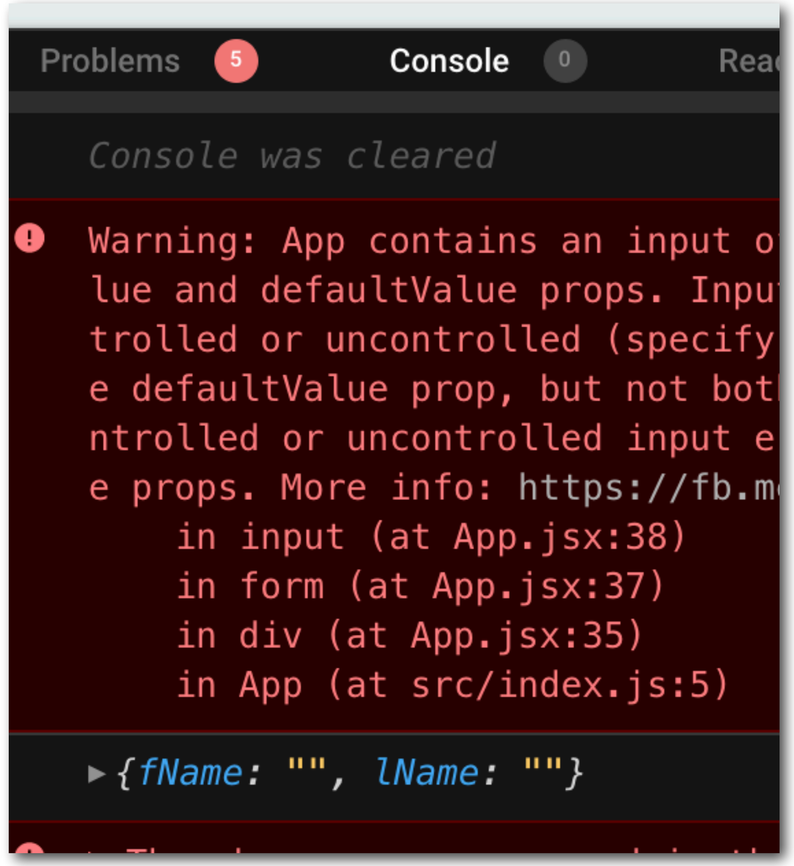
```
//WE WIL HOLD THE PREVIOUS VALUE
setFullName(prevValue=>{
  if (inputName==="fName"){
    return {
      fName: newValue,
      lName: prevValue.lName
    };//IF THE ELSE, WE UPDATE LNAME
  }else if(inputName==="lName"){
    return {
      fName: prevValue.fName,
      lName: newValue
    };
  }})}
```

CHANGING COMPLEX STATE

3. We will use if statement to render first name if there is a change in first name, and render last name if there is a change in the last name using event.target.name property by checking the the values of fName and lName

LOGIC: IS THERE IS A CHANGE IN fName THEN SET THE fName to the newValue BUT KEEP lName TO ITS PREVIOUS EXISTING VALUE.
ELSE IF THERE IS A CHANGE IN THE lName, THEN SET lName TO THE NEW VALUE BUT KEEP THE FLABE TO ITS PREVIOUS EXISTING VALUE.ADD THIS LOGIC IN THE setFullName function!!!
setFullName can take a function as input. We will write our function inside the setFullName function
LETS DELETE THE CODE:

```
function handleChange(event){
  const newValue = event.target.value;
  const inputName= event.target.name
setFullName(prevValue=>{//When this function called,
  // we can access the previous value.
  // Let's name it prevValue
  console.log(prevValue);//Console now gives us both objects
This MEANS REACT REMEMBERS THE OLD VALUES. WE WILL USE THAT REMEMBERED VALUE
NEXT HOW WE CAN WRITE A FUCNTION THAT RENDERS DIFFERENT OUTPUT BASED ON THIS PREVIOUS VALUES
//NEXT SLIDE……USING IF STATEMENT INSIDE THE SETFULLNAME
})}
```



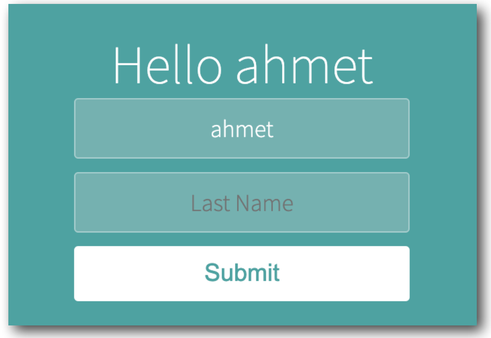
prevValue=> React remembers the value of the previous objects. Using arrow function, we can hold that value any use in the new codes! So we used that remembered value depend on this previous value. SO prevValue remembers both fName and lName values because setFullName function has these two javascript objects

CHANGING COMPLEX STATE

3. We will use if statement to render first name if there is a change in first name, and render last name if there is a change in the last name using event.target.name property by checking the the values of fName and lName

LOGIC: IS THERE IS A CHANGE IN fName THEN SET THE fName to the newValue BUT KEEP lName TO ITS PREVIOUS EXISTING VALUE.
ELSE IF THERE IS A CHANGE IN THE lName, THEN SET lName TO THE NEW VALUE BUT KEEP THE FLABE TO ITS PREVIOUS EXISTING VALUE.ADD THIS LOGIC IN THE setFullName function!!!
setFullName can take a function as input. We will write our function inside the setFullName function

```
function handleChange(event){
  const newValue = event.target.value;
  const inputName= event.target.name
  setFullName(prevValue=>{

  if (inputName==="fName"){
    return {
      fName: newValue,//If there is a change in fName, use entered nw value in there
      lName: prevValue.lName//Keeep lName as it was before. The old value. TEST
    };
  }

})
}
```



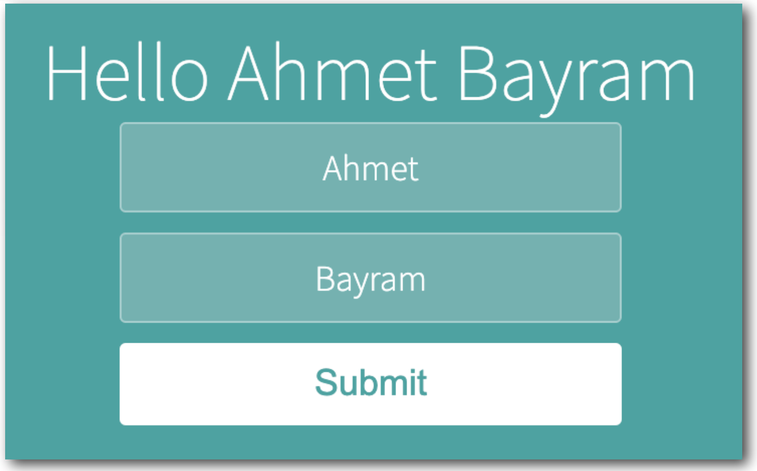
USE SAME CONCEPT TO RENDER LAST NAME. NEXT SLIDE

prevValue=> React remembers the value of the previous objects. Using arrow function, we can hold that value any use in the new codes! So we used that remembered value depend on this previous value. SO prevValue remembers both fName and lName values because setFullName function has these two javascript objects

CHANGING COMPLEX STATE

3. We will use if statement to render first name if there is a change in first name, and render last name if there is a change in the last name using event.target.name property by checking the the values of fName and lName

```
LOGIC: IS THERE IS A CHANGE IN fName THEN SET THE fName to the newValue BUT KEEP lName TO ITS  PREVIOUS
EXISTING VALUE.
ELSE IF THERE IS A CHANGE IN THE lName, THEN SET lName TO THE NEW VALUE BUT KEEP THE FLABE TO ITS PREVIOUS
EXISTING VALUE.ADD THIS LOGIC IN THE setFullName function!!!
setFullName can take a function as input. We will write our function inside the setFullName function
```




```
function handleChange(event){
  const newValue = event.target.value;
  const inputName= event.target.name
  setFullName(prevValue=>{

  if (inputName==="fName"){
    return {
      fName: newValue,
      lName: prevValue.lName
    };//IF THE ELSE, WE UPDATE LNAME AS NEW VALUE, AND FNAME WILL NOT CHANGE
  }else if(inputName==="lName"){
    return {
      fName: prevValue.fName,
      lName: newValue
    };
  }})}//DONE THIS NOW WORKS AS EXPECTED TEST
```

prevValue=> React remembers the value of the previous objects. Using arrow function, we can hold that value any use in the new codes! So we used that remembered value depend on this previous value. SO prevValue remembers both fName and lName values because setFullName function has these two javascript objects

CHANGING COMPLEX STATE

6.WE WROTE THIS CODE STEP BY STEP. BUT WE CAN MAKE THIS CODE A LITTLE SHORTER. WE CAN DO DESTRUCTURING:

1. Event.target has both value and name properties. We can use object destructure to replace newValue and inputName :

```
// const newValue=event.target.value;
// const inputName=event.target.name;
const {value,name}= event.target;
```

2. Now replace inputName ==>name AND newValue ===> value

```
if (name==="fName"){
        return {
          fName: value,
          lName: prevValue.lName
        };
      }else if(name==="lName"){
        return {
          fName: prevValue.fName,
          lName: value
        }//DONE COMPLETELY
```

HERE IS THE LAST WARING. NEVER USE event.target RELATED FUNCTION INSIDE setFunction FOR EXAMPLE, IF YOU USE event.target.name instead of name THEN WE GET ERROR. BECAUSE event IS A SYSTHETIC EVENT WE CAN'T ACCESS IN THE setFunctions

```
if (event.target.name==="fName")

else if(event.target.name==="lName")
```

NEVER USE THIS CAUSE WE CAN'T ACCESS IT IN THE SET STATE FUNCTION

```
import React,{useState} from "react";

function App() {
  const [fullName,setFullName] = useState({fName:"",lName:""});

  function handleChange(event){
    // const newValue=event.target.value;
    // const inputName=event.target.name;
    const {value,name}= event.target;
    setFullName(prevValue =>{
      if (name==="fName"){
        return {
          fName: value,
          lName: prevValue.lName
        };
      }else if(name==="lName"){
        return {
          fName: prevValue.fName,
          lName: value
        }
      }
    });
  }

    return (
    <div className="container">
      <h1>Hello {fullName.fName} {fullName.lName}</h1>
      <form>
        <input onChange ={handleChange} name="fName"
        placeholder="First Name" value={fullName.fName}/>
        <input onChange ={handleChange} name="lName"
        placeholder="Last Name" value={fullName.lName} />
        <button>Submit</button>
      </form>
    </div>
  );
}
export default App;
```

COMPLETED

App.jsx

```jsx
import React,{useState} from "react";

function App() {
  // const [fName,setFName] = useState("");
  // const [lName,setLName] = useState("");
//   function updateFName(event){
//     const firstName=event.target.value;
//     setFName(firstName);}
//  function updateLName(event){
//       const lastName=event.target.value;
//       setLName(lastName);}
const [fullName,setFullName] =useState({fName:"",lName:""});

function handleChange(event){
  // const newValue=event.target.value;
  // const inputName=event.target.name;
  const {value,name}= event.target;
  // newValue => value
  // inputName => name
  setFullName(prevValue =>{
    if (name==="fName"){
      return {
        fName: value,
        lName: prevValue.lName
      };
    }else if(name==="lName"){
      return {
        fName: prevValue.fName,
        lName: value
      }
    }
  });
}

  return (
    <div className="container">
      {/* <h1>Hello {fName} {lName}</h1> */}
      <h1>Hello {fullName.fName} {fullName.lName}</h1>
      <form>
        <input onChange={handleChange} name="fName" placeholder="First Name" value={fullName.fName}/>
        <input onChange={handleChange} name="lName" placeholder="Last Name" value={fullName.lName}/>
        <button>Submit</button>
      </form>
    </div>
  );

}
export default App;
```

CHANGING COMPLEX STATE PRACTICE

In this project, the challenge is to make th ecode work.

Final outcome should generate the input we types :

On the right what should be the end result

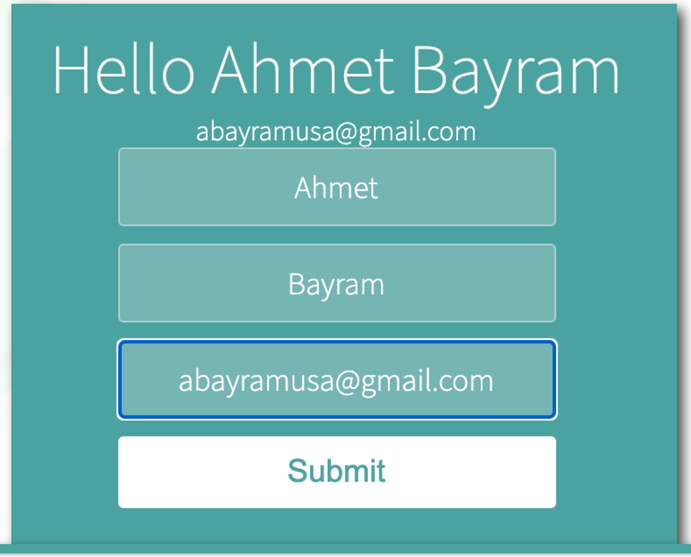
Use developer tool to check what the state look like as you change your code

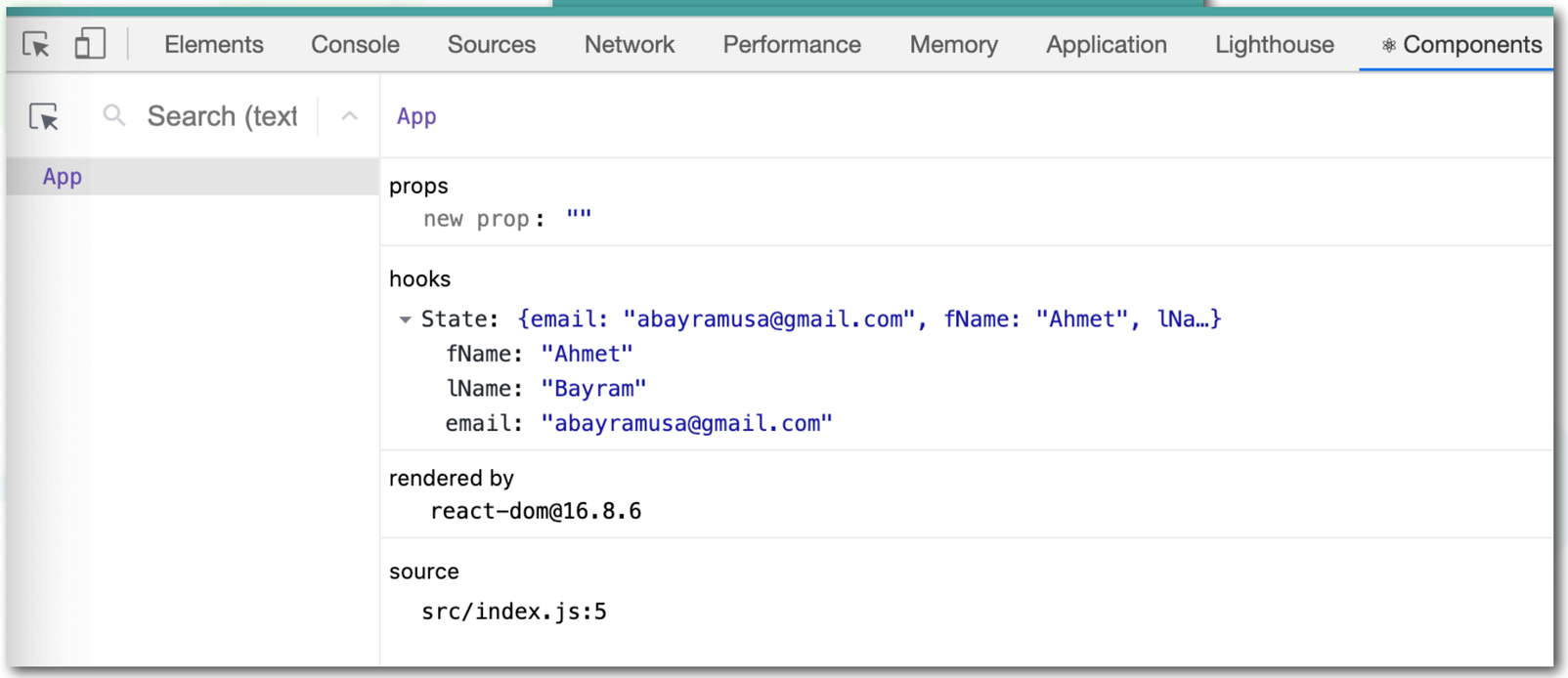
Managing state with js object

Making use of the previous state when updating

Working with forms with real

Handling the events





SOLUTIONS

```
<form>
  <input
  onChange={handleChange}
  name="fName"
  placeholder="First Name"
  value={contact.fName}
  />
  <input
  onChange={handleChange}
  name="lName"
  placeholder="Last Name"
    value={contact.lName}
  />
  <input
  onChange={handleChange}
  name="email"
  placeholder="Email"
  value={contact.email}
  />
```

...prevValue

...prevValue

```
function handleChange(event){
  const {name, value} = event.target
  console.log(name)//fName, or lName, or ema
  console.log(value)//user entered value

  //set contact is used to update the contac
  setContact(prevValue =>{
    if(name==='fName'){
      return {
        fName:value,
        lName:prevValue.lName,
        email:prevValue.email
      }
    }else if(name==='lName'){
      return {
        fName:prevValue.fName,
        lName:value,
        email:prevValue.email
      }
    }else if(name==='email'){
      return {
        fName:prevValue.fName,
        lName:prevValue.lName,
        email:value
      }
    }
  })
}
```

CHANGING COMPLEX STATE PRACTICE

Now we have contact constant with initial value is set to 3 properties and we can update the contact using setContact

When want that update happen when the 3 input changes

1. So lets add onChange attribute on the 3 input elements

```
<input onChange={handleChange} name="fName" placeholder="First Name" />
       <input onChange={handleChange} name="lName" placeholder="Last Name" />
       <input onChange={handleChange} name="email" placeholder="Email" />
```

2. Create handleChange function that will hold the event and use that when each of the input change,

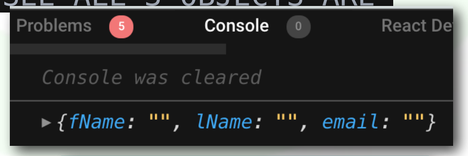
```
function handleChange(event){  }
```

3.destructure evet.target (to reach name, placeholder, value) in the handleChange function.to Use in the function to change the state:

```
  function handleChange(event){
     const {name,value} =event.target;
    console.log(name);
    console.log(value); }
```

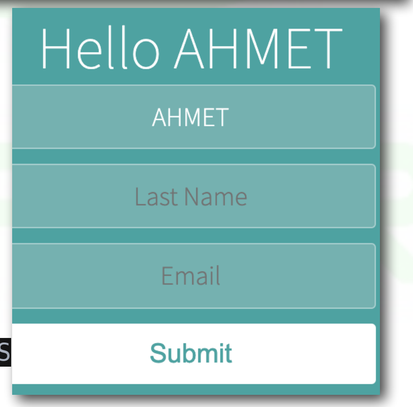
4 Update the contact the object ONLY WHEN THERE IS A CHANGE IN THE RELATED FIELDS. To update contacts use setContact. Remember we will hold of the previous value of the contact. So we can use arrow function:

```
function handleChange(event){
     const {name,value} =event.target;
     setContact(prevValue => {
      console.log(prevValue);//WHEN WE TYPE ANY INPUT, WE WILL SEE ALL 3 OBJECTS ARE REMEMBERED
     })}
```



5. LET'S UPDATE FIRST NAME IF THERE IS ANY CHANGE IN THE fName input field

```
  function handleChange(event){
     const {name,value} =event.target;
     setContact(prevValue => {
      if (name==="fName"){
        return {
          fName:value,
          lName:prevValue.lName,
          email:prevValue.email
        };
      }
     })
   }//TYPE IN FIRST NAME FIELD TO UPDATE FIRST NAME. DO THE REST USING SAME STEPS
```




```
import React, { useState } from "react";

function App() {
  const [contact, setContact] = useState({
    fName: "",
    lName: "",
    email: ""
  });

  function handleChange(event){
    const {name,value} = event.target;
    setContact((prevValue)=>{
      if (name==="fName"){
        return {
          fName:value,
          lName:prevValue.lName,
          email:prevValue.email
        };
      }else if(name ==="lName"){
        return {
        fName:prevValue.fName,
        lName:value,
        email:prevValue.email
        };

      }else if(name==="email"){
        return{
        fName:prevValue.fName,
        lName:prevValue.lName,
        email:value
        }
      }
    })
  }

  return (
    <div className="container">
      <h1>
        Hello {contact.fName} {contact.lName}
      </h1>
      <p>{contact.email}</p>
      <form>
        <input onChange={handleChange} value
={contact.fName} name="fName" placeholder="First Name" />
        <input onChange={handleChange} value
={contact.lName} name="lName" placeholder="Last Name" />
        <input onChange={handleChange} value
={contact.eName} name="email" placeholder="Email" />
        <button>Submit</button>
      </form>
    </div>
  );
}

export default App;
```

CHANGING COMPLEX STATE PRACTICE

Update the contact the object ONLY WHEN THERE IS A CHANGE IN THE RELATED FIELDS. To update contacts use setContact. Remember we will hold of the previous value of the contact. So we can use arrow function:

```
  function handleChange(event){
    const {name,value} = event.target;
    setContact((prevValue)=>{
      if (name==="fName"){
        return {
          fName:value,
          lName:prevValue.lName,
          email:prevValue.email
        };
      }else if(name ==="lName"){
        return {
        fName:prevValue.fName,
        lName:value,
        email:prevValue.email
        };

      }else if(name==="email"){
        return{
        fName:prevValue.fName,
        lName:prevValue.lName,
        email:value
        }
      }
    })}
```

Last this to remember to do is to add the value attribute each of the input and keep them tight to the state

This code is also long, we can shorten these using ES6 spread Operator later

```
ge} value ={contact.fName} nam
ge} value ={contact.lName} nam
ge} value ={contact.eName} nam
```

```
import React, { useState } from "react";

function App() {
  const [contact, setContact] = useState({
    fName: "",
    lName: "",
    email: ""
  });

  function handleChange(event){
    const {name,value} = event.target;
    setContact((prevValue)=>{
      if (name==="fName"){
        return {
          fName:value,
          lName:prevValue.lName,
          email:prevValue.email
        };
      }else if(name ==="lName"){
        return {
        fName:prevValue.fName,
        lName:value,
        email:prevValue.email
        };
else if(name==="email"){
        return{
        fName:prevValue.fName,
        lName:prevValue.lName,
        email:value
        }
      }
    })
  }

  return (
    <div className="container">
      <h1>
        Hello {contact.fName} {contact.lName}
      </h1>
      <p>{contact.email}</p>
      <form>
        <input onChange={handleChange} value ={contact.fName}
name="fName" placeholder="First Name" />
        <input onChange={handleChange} value ={contact.lName}
name="lName" placeholder="Last Name" />
        <input onChange={handleChange} value ={contact.eName}
name="email" placeholder="Email" />
        <button>Submit</button>
      </form>
    </div>
  );
}export default App;
```

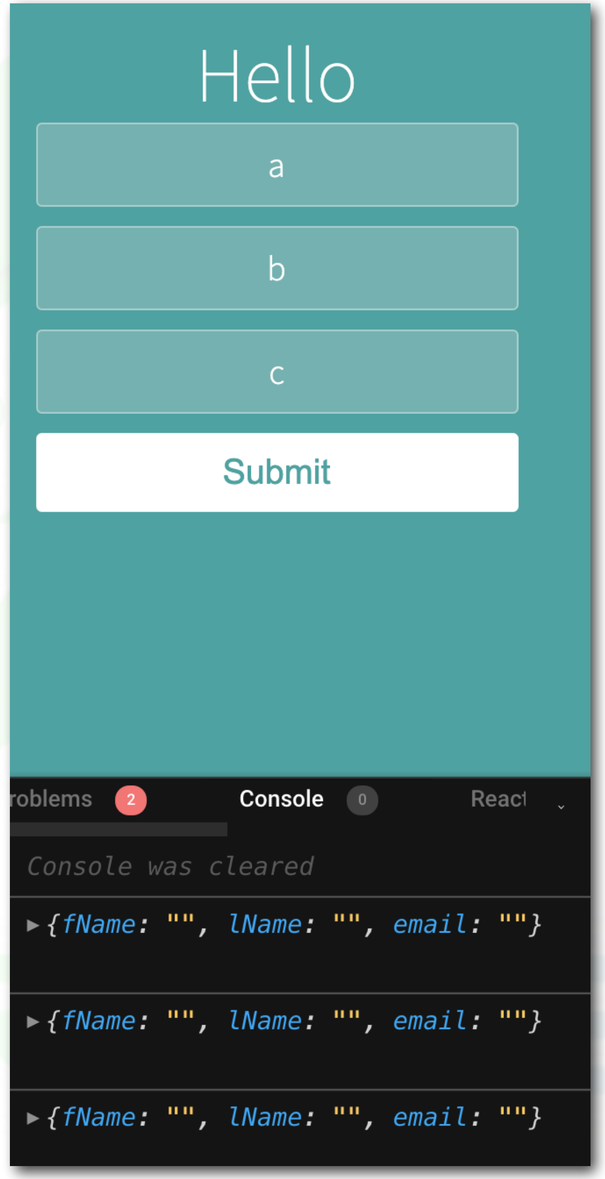
JAVASCRIPT ES6 SPREAD OPERATOR

Currently our handleChange function is too long. Let's use spread operator to add the previous values .DELETE ALL IF STATEMENTS FROM setContact and try to use spread operator

Return an object using spread operator adding the previous value of the objects(name,lastname,email properties), **...prevValue** will do that

```
//DELETE ABOVE AND ADD  BELOW AFTER LEARING SPREAD OPERATOR
//NOTE THAT name:value will be considered as string name
//With ES6 we can use [name] to pass the value of the name
  function handleChange(event){
     const {name,value} =event.target;
     setContact(prevValue => {
      console.log(prevValue)
        return {
          …prevValue//represents fName,lName,email.Taking the previous
values as TEST THIS NOW
//WHE WE TYPE IN ANY INPUT, WE DONT SEE ANY UPDATE BUT IT TRIGGERS THE OBJECT. GETTING
CLOSER
          };
     })}
```

```
//WHEN WE LOG AND TYPE THEN WE SEE THE OBJECTS
RENDERING
setContact(prevValue =>{
  console.log(prevValue);

  return {
    ...prevValue
  }
```

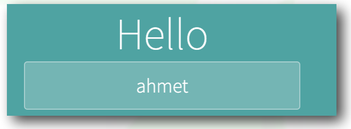


JAVASCRIPT ES6 SPREAD OPERATOR

Then add the new value for inputs that is passed in and the value user typed in **name: value**

```
setContact(prevValue => {
      return {
        …prevValue,//Taking the previous values as
        name: value// THIS WONT WORK CAUSE name is considered as string . Let's test and see.THIS OS NOT
   WE WANT. THIS RENDERS A NEW KEY VALUE PAIR KEY IS name
      };});
```

```
▸{fName: "", lName: "", email: "", name: "ahme"}
▸(1) ["fName"]
ahme
```

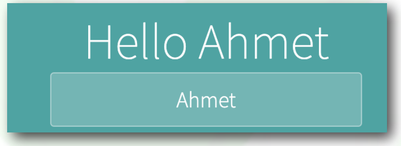



```
==========================================================================
```

TRY TO USE SQUARE BRACKETS

```
setContact(prevValue => {
      return {
        …prevValue,//Taking the previous values as
        [name]: value // adding the new value of an array which is name.User typed in this value.We must
   use array [name] in array. If we just use name: value then name will be considered as string. [name] is
   representing event.target.name but we can't use that is here. AGAIN we have [name] THAT COMES FROM
   EVENT.TARGET.NAME. WE USE IT IN [] TO HOLD THE VALUE IN THE const{name,value}. WE USE ARRAY IN THE KEY, AS
   OPPOSED TO USING JUST A STRING
      };});DONE TEST
```

```
▸{fName: "Ahme", lName: "", email: ""}
▸(1) ["fName"]
Ahmet
```



Setting object key by variable is a common issue. So let's check some common problem and error. Read below to understand why we must use array

https://stackoverflow.com/questions/11508463/javascript-set-object-key-by-variable

```
function handleChange(event) {
  const { name, value } = event.target;

  setContact(prevValue => {
    return {
      ...prevValue,
      [name]: value
    };
  });
}
```

TO DO LIST APP

```
/CHALLENGE: Make this app work by applying what you've learnt.
//1. When new text is written into the input, its state should be saved.
//2. When the add button is pressed, the current data in the input should be
//added to an array.
//3. The <ul> should display all the array items as <li>s
//1. When new text is written into the input, its state should be saved.
```

1. Create a new constant on the App function to give a name and update function :

```
const [inputText,setInputText] = useState("");
```

2. Think about Where to display the inputText. To do that we must add value={inputText} to our input property to control the input and keep the input value:

```
<input type="text" value={inputText}/>
```

3. When do we want to change the inputText?When do we want to call the function: Answer is when the input changes: so add onChange = {handleChange}:

```
<input  onChange = {handleChange} type="text"  value={inputText}/>
```

4. Create the hadleChange function to pass the event:function handleChange(event){}

5.create a constant that will hold the newValue of the input: const newValue=event.target.value;

6. Call setInputText to pass the newValue that Is entered the input field : setInputText(newValue);

```
  function handleChange(event){
    const newValue=event.target.value;
Console.log(newValue)
    setInputText(newValue);
    // NOTE THAT I DONT NEED TO THE PREVIOUS VALUE SO NO NEE TO GET IT}
```

Now we see the state updates the latest state you can see that in the dev console:AS we enter input, state is being updated. Part 1 is DONE

```
import React, { useState } from "react";

function App() {
const [inputText,setInputText] =
useState("");
function handleChange(event){
  const newValue=event.target.value;
  setInputText(newValue);}
  return (
    <div className="container">
      <div className="heading">
        <h1>To-Do List</h1>
      </div>
      <div className="form">
        <input
      onChange = {handleChange}
      type="text"
      value={inputText}/>
        <button>
          <span>Add</span>
        </button>
      </div>
      <div>
        <ul>
          <li>A Item </li>
        </ul>
      </div>
    </div>
  );
}
export default App;
```

TO DO LIST APP

```
//2. When the add button is pressed, the current data in the input should be added to an array.
```

Goal it to Keep tract all of the items that user entered inside an array

1. Create a new constant that will hold array of items, and a function that will set these items. The initial value is an empty array.

```
const [items, setItems] = useState([]);
```

2.Instead of A Item , we ll use map. The array items renders all of the li's. we will can map function that will have another function that will get access of the each item that user enters and store in the array of items. We can pass arrow function inside the map function . With arrow function we get access to each item(todoItem), and it will return li which has todoItem as the text that is in the list items:

```
    {items.map((todoItem)=>{//items represent the entire todoItem array
           return <li>{todoItem}</li>;})} //todoItem represents the single user entered values
```

We can simplify this cause we return single item, so delete return keyword, opening curly brace, closing semicoliun and closing curly brace:

```
 {items.map(todoItem => (
           <li>{todoItem}</li>))}
```

```
<ul>
  {items.map(todoItem =>  <li>{todoItem}</li>)}
</ul>
```

Again this means to Map through all of the items for each todoItem, create an li and put todoItem in that li.

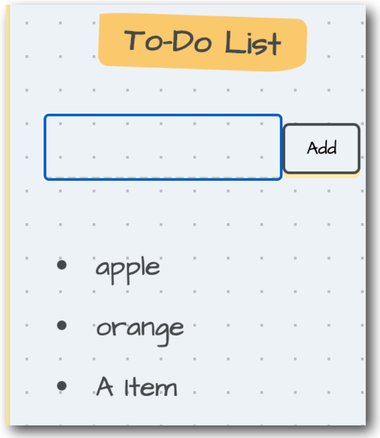
3.Next all we have to do is to call setItems when we click on add button:Use onClick to add a function addItem:

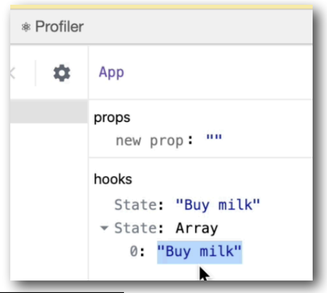
```
<button onClick={addItem}>
```

Create addItem function. It ll add the text inside our input into our items array. TO do that we need to call setItems function to hold the previous function and add the new input add the end of the array as a new item:To do this create a new arrow function and hold prevItems and return a new array using the spread operator(we can spread array, obhjects, buttons,...) to all all of the previous items, then add the new item that is inside the inputText

```
  function addItem(){
   setItems(prevItems => {
     return [...prevItems,inputText];/returning a new array….prevItems means add all of the previous items.inputText represent the new item is being added to the end of the array
   });
 }// DONE. NOW WE SHOULD BE ABLE TO ADD NEW ITEM TO THE ITEMS LIST USING items ARRAY. LAST LET'S CLEAR
```

```
import React, { useState } from "react";

function App() {
const [inputText,setInputText] = useState("");
const [items,setItems] = useState([]);
function handleChange(event){
  const newValue=event.target.value;
  setInputText(newValue);
}
function addItem(){
  setItems(prevItems => {
    return [...prevItems,inputText];
  });
  setInputText("");
}
  return (
    <div className="container">
      <div className="heading">
        <h1>To-Do List</h1>
      </div>
      <div className="form">
        <input  onChange = {handleChange} type="text"
value={inputText}/>
        <button onClick={addItem}>
          <span>Add</span>
        </button>
      </div>
      <div>
        <ul>
          {/* {items.map((todoItem)=>{
            return <li>{todoItem}</li>;
          })} */}
        {/* Above can be simplified: */}
        {/* Map through all of the items for each
todoItem create an li and put todoItem in that li.
This is what this means */}
          {items.map(todoItem=><li>{todoItem}</li>)}
          <li>A Item </li>
        </ul>
      </div>
    </div>
  );
}
```





TO DO LIST APP

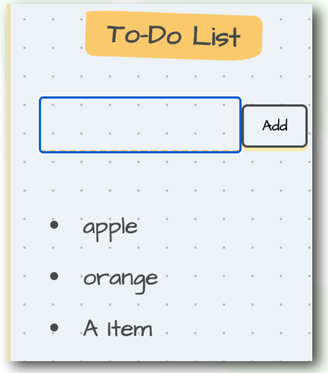
Let's clear the input after your add a new item

In addItem() function, We need to set the input to an empty string to clear it. So use setUnputText to set the input to empty string and pass empty string after the item is added. REMEMBER THE STATE OF THE INPUTTEXT IS BEING ASSIGNED BY setUnputText function

1. In addItem() add setInputText(""); In the addItem function after setItems : This will add new inputText in the array THEN PAS THE EMPTY STRING IN THE INPUT TEXT

```
setInputText("");
```

```
function addItem(){
  setItems(prevItems => {
    return [...prevItems,inputText]
});
  setInputText("");
}
```



DONE

Now when you click on Add button, it will add the last item at the end of the array, then clear the input

```
import React, { useState } from "react";
function App() {
  const [inputText, setInputText] = useState("");
  const [items, setItems] = useState([]);

  function handleChange(event) {
    const newValue = event.target.value;
    setInputText(newValue);}

  function addItem() {
    setItems(prevItems => {
      return [...prevItems, inputText];
    });
    setInputText("");}

  return (
    <div className="container">
      <div className="heading">
        <h1>To-Do List</h1>
      </div>
      <div className="form">
        <input onChange={handleChange} type="text" value={inputText} />
        <button onClick={addItem}>
          <span>Add</span>
        </button>
      </div>
      <div>
        <ul>
{/* {items.map((todoItem)=>{
          return <li>{todoItem}</li>;
        })} */}
        {/* Above can be simplified: */}
        {/* Map through all of the items for each todoItem create an li and put todoItem in that
li. This is what this means */}
          {items.map(todoItem => (
            <li>{todoItem}</li>
          ))}
        </ul>
      </div>
    </div>);}
export default App;
```

TO DO LIST APP

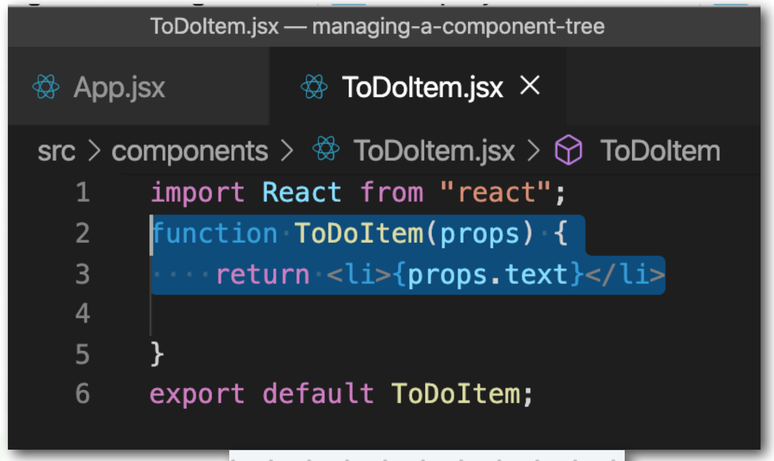
CURRENTLY WE HAVE EVERY COMPONENT IN THE APP COMPOENT. NORMALLY WE HAVE DIFFERENT COMPONENTS AND PASS THOUSE COMPONENT IN THE APP COMPONENT. We will separate the app into separate components line heading, div with input and button, each li's

1. Create new component :ToDoItem.jsx

2. Takeout <li> and create a new component for this:

3. Create ToDoItem function that will return <li> that will contains Text and Export the toDoItem:

```
import React from "react";
function ToDoItem() {
    return <li>Text</li>}
export default ToDoItem;
```



4. In App.jsx, Import ToDoItem in the App.jsx : import ToDoItem from "./ToDoItem";

5. Now we can use this component instead of {todoItem} so replace this li with <ToDoItem />


```
        <ul>
         {items.map(todoItem => (
            <ToDoItem />))}
        </ul>//TEST THIS NOW RENDERS text item when we enter click because of return <li>Text</li>
```

6. To display each of the different to do item each time we map through from the map as text, we should use props:

```
<ToDoItem text={todoItem}/>))}
```

7. Now we can receive this text inside ToDoItem as props: In ToDoItem component:

```
function ToDoItem(props) {
    return <li>{props.text}</li>}//TEST NOW WE SHOULD RETURN toDoItem AS AN li on the UI
//THIS WAS A REVIEW OF MOST OF THE THINGS WE LEARNED SO FAR. LET'S TALK ABOUT THIS ToDoItem a little more
```

MANAGING COMPONENT TREE

```
function ToDoItem(props) {
    return <li>{props.text}</li>}/
```

Here, Note that we can't modify our props. Props are read only. We can't modify props. It is being passed over from App.jsx parent component and returned in the ToDoItem child component.

We can't modify props. But can use useState inside ToDoItem component.

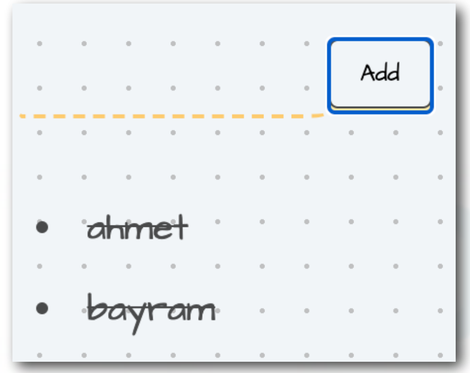
For example, when we click an item, we want to cross out text, put a line through that item.

We can do this using useState

1. In ToDoItem.jsx Test if we can use inline styling to cross out the li item.

```
return <li style ={{textDecoration:"line-through"}}>{props.text}</li>}
```

```
Now all items are being crossed out.
We want to cross out when ONLY WE CLICK ON THE ITEM
We will add onClick to detect the click.
We can add to li or div in
```



HOW TO CHANGE STYLE AFTER A CLICK ON THE ELEMENT

We will add onClick to detect the click on SPECIFIC toDoItem.

In ToDoItem, Lets put the li in a div:

```
return (<div><li style ={{textDecoration:"line-through"}}>{props.text}</li></div>)
```

We can add onClick to li or div.It does't matter. Lets add in the div.

```
<div onClick={handleClick}>
```

Create handle Click function.

```
function handleClick(){}
```

This function will detect if any item is done. Here we need one thing to know: 1. If item isDone or not. If the item is done, we click on that item and cross out that item. So Let's create a constant isDone and a function to update the state of isDone

```
const [isDone,setIsDone]=useState(false);//Item is not initially done, so set it false
```

Now we can tell our handleClick function what to do: It will call setIsDone and update the state of isDone when the button is clicked. It will make it true or false in this case. So when we click, isDone will take the opposite of its previous value. It isDone is true, it will be false, if isDune is false, then it will be true. We can write if else statements but simply we can get eta previous value and return the opposite of prevValue:

```
  function handleClick(){
    setIsDone(prevValue=>{
      return !prevValue;
    })}
```

Last thing we need to do is to use isDone and render `{textDecoration:"line-through"}` conditionally the cross out part

```
<li style ={{textDecoration: isDone ? "line-through":"none"}}>{props.text}</li>
```

MATERIAL UI

```
https://www.npmjs.com/package/@material-ui/core
npm install @material-ui/core

https://www.npmjs.com/package/@material-ui/icons
npm install @material-ui/icons
```

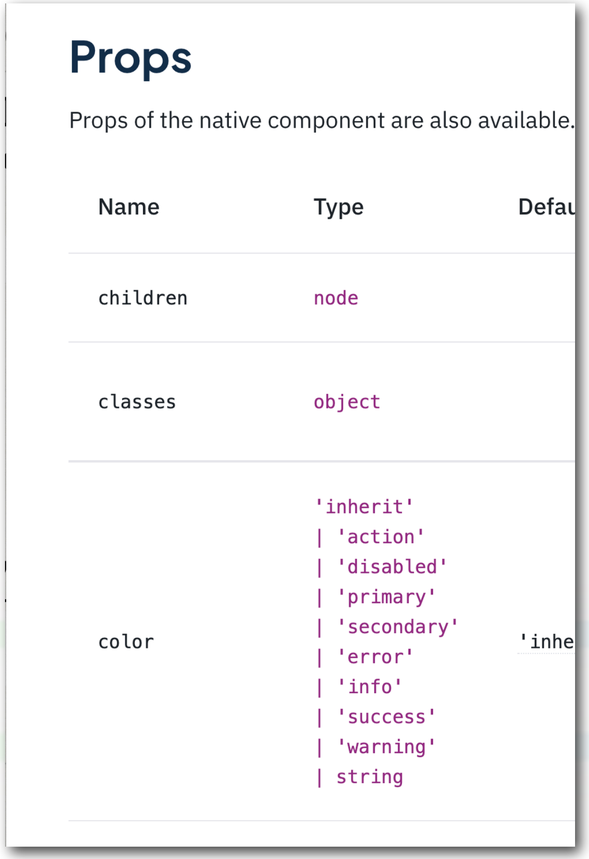
ADD HOME ICON

Import the related material ui package
Use that component in anywhere of the application

```
import HomeIcon from "@material-ui/icons/HomeTwoTone"
```

```
<HomeIcon />
```

STYLING ICONS:
https://mui.com/api/svg-icon/
This api link explain the combinations



Props

Props of the native component are also available.

Name	Type	Defau
children	node	
classes	object	
color	'inherit' | 'action' | 'disabled' | 'primary' | 'secondary' | 'error' | 'info' | 'success' | 'warning' | string	'inhe

ADD HOME ICON

We can search the icons from this link:
https://mui.com/components/material-icons/

Typography

```
import { Typography } from "@material-ui/core";
```

BUTTONS

HTTPS://MUI.COM/COMPONENTS/BUTTONS/#MAIN-CONTENT

BUTTON GROUPS

DONEC QUIS NUNC