

React Overview

I. INTRODUCTION TO REACT

What is React?

- React is a Javascript Library For building frontend UI
- Developed at **Facebook in 2011 by Jordan Walke Made Public in 2013**
- The current Version using React is:16.8 .
- Some of the companies that are using React: Facebook, Instagram, Airbnb, Dropbox, Netflix

JavaScript has many Libraries, why do we use React?

Javascript has many Libraries Like

- JQuery Library
- D3.js Library
- UnderScore Library
- Anime.js library but there are some advantages of React Over Others
- Declarative
- Component-Based
- Learn Once Write EveryWhere
- Single Page Apps

1. Declarative

There are two kinds of programming,

- Declarative Programming
- Imperative Programming

Declarative Programming:

Its main goal is to get desired output without describing how to get it .

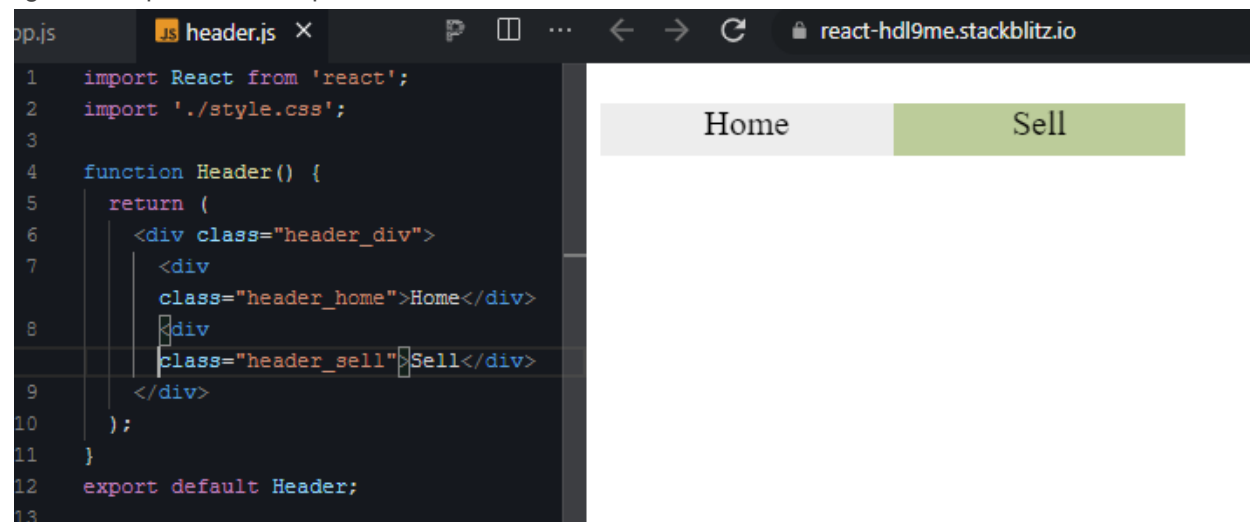
Imperative Programming:

Its main goal is to describe how to get output or accomplish it

React is Declarative in nature.

2.Component-Based

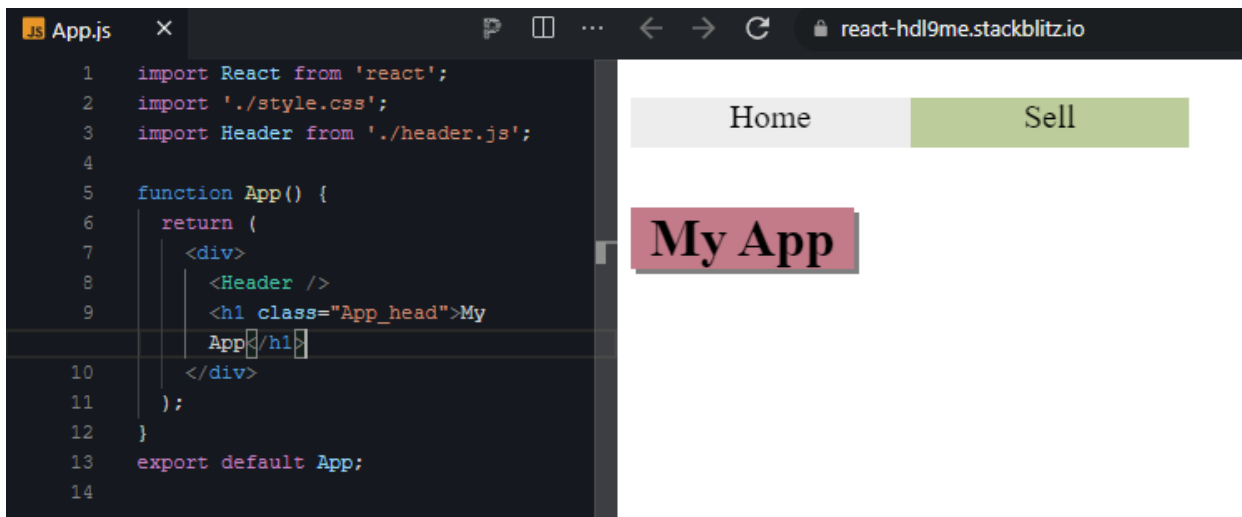
Since React is a component Based When we have to code the same data to different .js files we need not write that code again, Just pass that component to the file



The screenshot shows a code editor with a file named 'header.js' open. The code defines a 'Header' function that returns a JSX element. The JSX element consists of a 'div' with class 'header_div' containing two 'div' elements: one with class 'header_home' containing the text 'Home', and another with class 'header_sell' containing the text 'Sell'. The 'Sell' button is highlighted in green in the rendered output. The browser's address bar shows 'react-hdl9me.stackblitz.io'.

```
1 import React from 'react';
2 import './style.css';
3
4 function Header() {
5   return (
6     <div class="header_div">
7       <div
8         class="header_home">Home</div>
9       <div
10        class="header_sell">Sell</div>
11     </div>
12   );
13 }
14
15 export default Header;
```

Home Sell



3.Learn Once Write Everywhere

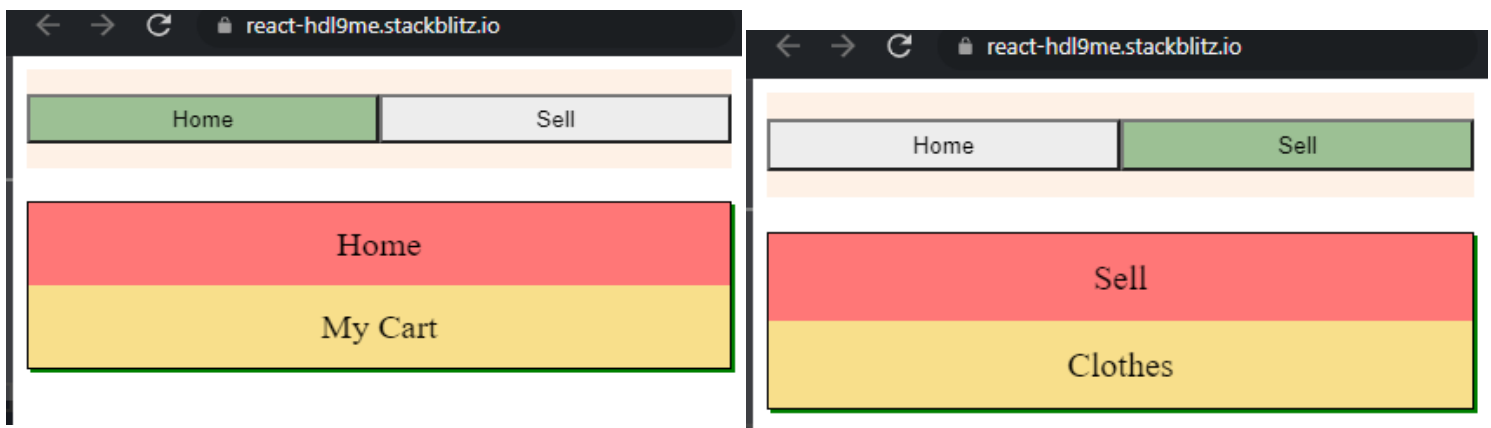
With this approach, an experienced React developer can work on Android and ios App much Faster

Example: If any developer know React js then it's easy for the developer to work on react native and any other Javascript libraries as Well

4.Single Page AppLication

A single-page application is one that doesn't need to reload the page during its use, Since react works on states, React does not need to reload the page, only states change will take you to next page Example :

Case When Home Button is Clicked



Case When Sell Button is Clicked

In the above example,Both Home and Sell pages are loading on the same URL

code of above Example

```

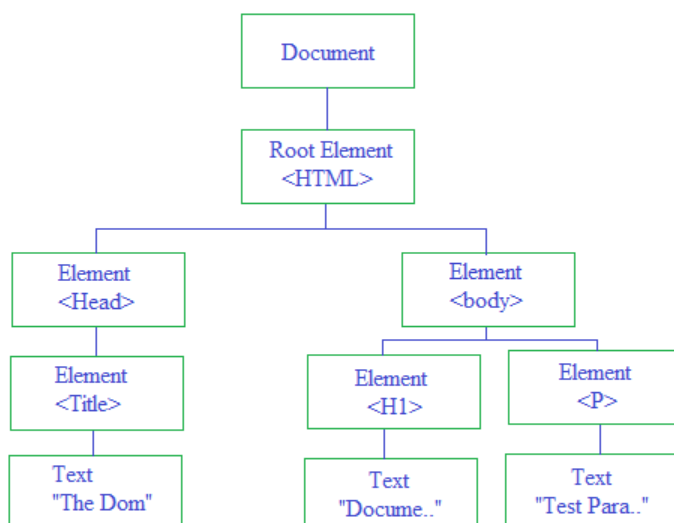
15 app.js > App
1  import React from 'react';
2  import './style.css';
3  import React, { useState } from 'react';
4
5  const App = () => {
6    const [active, setActive] = useState(-1);
7    let homeClass = -1;
8    let sellClass = -1;
9    {active == -1 ? (homeClass = 'App_home') : (homeClass = 'App_sell')}
10   {active == -1 ? (sellClass = 'App_sell') : (sellClass = 'App_home')}
11   return (
12     <>
13     <div class="App_div App_head">
14       <button
15         class={homeClass}
16         onClick={() => {
17           setActive(1);
18         }}
19       >
20         Home
21       </button>
22       <button
23         class={sellClass}
24         onClick={() => {
25           setActive(-1);
26         }}
27       >
28         Sell
29       </button>
30     </div>
31     <div>
32       {active == 1 ? (
33         <>
34           <div class="App_data_to_show">
35             <div class="App_data_to_show_home_item_one">Home</div>
36             <div class="App_data_to_show_home_item_two">My Cart</div>
37           </div>
38         </>
39       ) : (
40         <div class="App_data_to_show">
41           <div class="App_data_to_show_home_item_one">Sell</div>
42           <div class="App_data_to_show_home_item_two">Clothes</div>
43         </div>
44       )}
45     </div>
46   );
47 };
48
49 export default App;

```

II. VIRTUAL DOM

What is DOM?

DOM is a way to represent the webpage in a structured hierarchical way so that it will become easier for programmers and users to glide through the document.



DOM STRUCTURE

The tags `<p></p>` of Html are said to be objects (p object) in DOM

Why do we need DOM?

HTML is used to structure the web pages and Javascript is used to add behavior to that web pages.

When an HTML file is loaded into the browser, the javascript can not understand the HTML document directly So, a corresponding document is created(DOM).

DOM(Document Object Model) is basically the representation of the same HTML document but in a different format(hierarchical)

With the use of objects of DOM, Javascript interprets DOM easily i.e javascript can not understand the html tags(like p tag h2 tag) in HTML document but can understand object p in DOM and hence, Javascript can access each of the objects (p,h2, etc) by using different functions and modify them.

What is virtual DOM?

A virtual DOM object has the same properties as a real DOM object, but it lacks the real thing's power to directly change what's on the screen.

Why do we need Virtual DOM in React, meanwhile it is the blueprint of Real DOM?

<code></code>		<code></code>
<code>A</code>		<code>A</code>
<code>B</code>	-----Changes Made-----	<code>B</code>
<code>C</code>		<code>D</code>
<code></code>		<code></code>

- Real Dom will updates all the list items(i.e. A ,B,C to A B,D respectively)
- Virtual Dom will only update list item C to list item D

The manner of Updating the dom objects is different in both

In virtual DOM making changes to DOM objects is Lightweight, that is effective for a big project hence React prioritize virtual DOM

Why is Virtual DOM faster and how does it work internally?

When any changes are made in the application, every time a new virtual dom is created.

This new Virtual DOM tree is then compared with the previous Virtual DOM tree, and make a note of the changes. After this, it finds the best possible ways to make these changes to the real DOM.

Now only the updated elements will get rendered on the page again.

Diffing: This process of comparing the current Virtual DOM tree with the previous one

How to React Update Real DOM from Virtual DOM?

Once React finds changes through virtual DOM that exactly have changed then it updates those objects only, on real DOM.

React uses something called batch updates to update the real DOM. It just means that the changes to the real DOM are sent in batches instead of sending any update for a single change in the state of a component.This entire process of transforming changes to the real DOM is called **Reconciliation**

III. REACT VS ANGULAR VS VUE

What is the difference between Angular,React and Vue?

Point Of Difference	Angular	React	Vue
Developed	Google in 2010	developed in 2011 at Facebook and made public in 2013	Ex-Google Evan You in 2014
Framework/Library	Framework	Library	Framework
Component Based	yes	yes	yes
Syntax	Split HTML + TypeScript	JSX-JavaScript +HTML	Separate HTML and JS
Performance	No Virtual DOM	Virtual DOM	Virtual DOM
Learning Ease	Comparatively Tough	Easier than Angular	Easier than React and Angular
Popularity	Second Most Popular	Most Popular	Less Popular

What are Pros and Cons of Framework and Library?

Pros Of Framework

1. Easy Debugging Process , Several web frameworks built-in support for debugging and QA (Quality Assurance) testing.
2. Improved Code Efficiency
3. Easy Code Reusability
4. **Accelerated Development:** With a pre-defined template and a coding interface, developers can save hours of strenuous coding effort.
5. Enhanced Security

Cons of Framework

1. The framework's core behavior can't be modified, meaning that when you use a framework, you are forced to respect its limits and work the way it is required .

Pros of Library

1. The control is always with you. Requirement-wise development. (Whatever code we need we just need to install library for that code , one by one according to our needs)
2. Libraries are meant to do some specific tasks only.
3. Ease of Use

Cons of Library

1. **Bugs propagation:** If any bugs are found in a library then it will cause a problem in the production
2. Risk of computer viruses.

IV. COMPONENTS AND JSX

What are Components in React?

Components: A component is one of the core building blocks of React

Component on React tells what should be rendered on the Screen

Components let you split the UI into independent, reusable pieces, and think about each piece in isolation.

There are two types of Components in React

1.Class Component

```

JS App.js > ...
import React, { Component } from "react";

class App extends Component {
  constructor() {
    super();
  }

  render() {
    return <h1>Welcome to React</h1>;
  }
}
export default App;

```

React Class Component

2.Functional Component

```

function App() {
  return (
    <div>
      <h1>Welcome to React</h1>
    </div>
  );
}
export default App;

```

React Functional Component

What is JSX?

- JSX: JSX stands for “**J**avascript **X**ML”
- It is a Syntax extension to JavaScript-based in ES6(new Version of Javascript)
- JSX allows you to write HTML in React by converting HTML into React Components

In which form React code is converted by the compiler and then run that code ?

```

import React from 'react';

function App() {
  return (
    <div>
      <h1>Welcome to React</h1>
    </div>
  );
}

export default App;

//The above code will be converted by compiler in these form,(These is the actual code running on backend)
React.createElement("h1",null,"Welcome to React")

```

What is React and ReactDOM in the imports below?

```

import React from "react";
import ReactDOM from "react-dom";

```

React - The task of React is to create an element

ReactDOM - The task of ReactDOM is to render that element

V. INSTALL NODE AND NPM

What is NPM?

Node Package Manager

Npm is the package manager for the Node Javascript platform

When we download any library Where does that library manage?

After we download the library there is a folder name `node_modules` , library will go inside that particular `node_modules` folder and will come in `package.json`

What are `package.json` and `package-lock.json`?

package.json is used for defining more dependencies - like defining project properties, description, author & license information, scripts, etc.

package-lock.json is solely used to lock dependencies to a specific version number

What is `npm i`?

`npm i` can be used to set up a new or existing npm package.

How to check the version of the node?

`node --version`

VI. INSTALL CREATE REACT APP

What is a boilerPlate in any language?

BoilerPlate refers to standardized text, copy, documents, methods, or procedures that may be used over again without making major changes to the original.

A boilerplate is commonly used for efficiency and to increase standardization in the structure and language of written or digital documents .

What is `create-react-app`?

Create-react-app is a react boilerplate using which we can start writing react code very easily.

Or

Creating a React App is a comfortable environment for learning React and is the best way to start building a new single-page application in React.

What is `npx` ?

`npx` - Node Package Execute

It is an npm package runner that can execute any package that you want from the npm registry without even installing that package.

It comes with the npm, when you installed npm above the 5.2.0 version then automatically `npx` will be installed.

Packages used by `npx` are not globally installed ,while packages Installed by npm are installed globally .

How to install react boilerplate?

Installation

If you have not already created a folder for your project and you want that new folder will be created and boiler plate will be inside that

`npx create-react-app Folder Name`

If you have already created a folder and you want that boilerplate will be installed inside that folder

`npx create-react-app` .

What is the role of `npx` in react boilerplate installation (`npx create-react-app`) ?

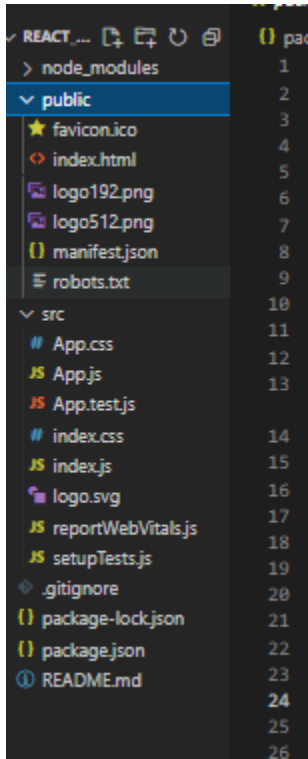
`npx` would download all the dependencies for create-react-app (i.e. it would download all the dependencies of your react boilerplate)

What are node_modules ?

It is a folder in react-boiler-plate.

All the libraries,dependencies and packages we download for our project goes inside node_modules

Boilerplate Structure



What is npm start ?

npm start - with npm start code is compiled, bundle is created and your side starts run

What are the scripts in the package.Json ?

The script is just a set of instructions that you want to run

```
"start": "react-scripts start",
```

 npm-start with the definition in package.json file

```
"build": "react-scripts build",
```

 npm run build with the definition in package.json file

Develop in React.

I. BABEL AND WEBPACK OVERVIEW

What is Localhost?

Localhost just corresponds to the IP address of Your System .

What is PORT?

A Port is a virtual Point where network connections Start and end .

What is Babel?

Babel is a JavaScript compiler

Babel does two task

1. Convert ES6 to ES5 (As many browsers don't support ES6)
2. Convert JSX Syntax to js

What is Webpack?

It's a tool that lets you bundle all your .js Pages.

We created different .js files, webpack does the task of bundling all .js files into a single .js file and that file is known as bundle.js.

OR

Webpack is a static module bundler for JavaScript applications. Webpack enables you to take a fully dynamic application and package it into static files, which you can then upload and deploy to your server.

What is minify css?

Minifying a CSS file implies the removal of unnecessary characters in the source code to reduce the file size and facilitate faster loading of the site. When a user requests a web page, the minified version is sent instead of the full version, resulting in faster response times and lower bandwidth costs .

Input CSS

```
body {
  padding: 0;
  margin: 0;
}

body .pull-right {
  float: right !important;
}

body .pull-left {
  float: left !important;
}

body header,
body [data-view] {
  display: none;
}
```

Minified Output

```
body,html{height:100%;}body{padding:0;margin:0;}body .pull-right{float:right!important;}body .pull-left{float:left!important;}body [data-view],body header{display:none;opacity:0;transition:opacity .7s ease-in;}body [data-view].active{display:block;opacity:1;}body[data-nav=playground] header{display:block;opacity:1;}[data-view=home]{height:100%;}[data-view=home] button{opacity:0;pointer-events:none;transition:opacity 1.5s ease-in-out;}[data-view=home] button.live{opacity:1;pointer-events:all;}[data-view=home] .mdc-layout-grid_cell--span-4.mdc-elevation--z4{padding:1em;background:#fff}
```

What happens when we do npm start ?

1. After doing npm start Webpack confirms that all dependencies are coming, all css files are minified
2. It ensures that bundles are being created and the script is inside index.html files
3. When we do npm start, a server is created

That server needs two things

- Ip address -machine on which it is running
- Port

localhost - localhost corresponds to the IP of your machine

3000 - port on which server is running .

II. MORE ON COMPONENTS

A component is one of the core building blocks of React. Component on React tells what should be rendered on the Screen

What is the robots.txt file in react boilerplate?

A robot. txt file tells search engine crawlers, which URLs the crawler can access on your site

What is the manifest.json file in react boilerplate?

manifest.json provides information about an application (such as name, author, icon, and description) in a JSON text file

Or

It defines how the app should look like and configures the app's behavior on launch. It is an array of objects that are displayed on the home screen .

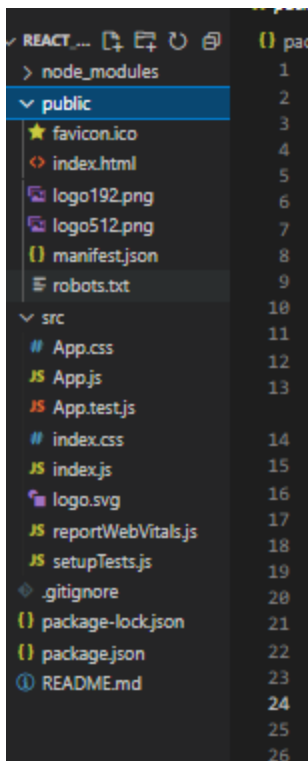
manifest.json is required by Chrome to show the Add to Home Screen prompt

What is the gitignore file?

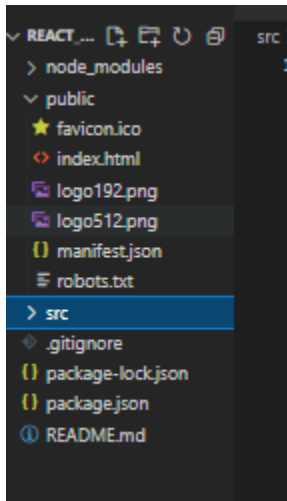
gitignore file is a text file that tells Git which files or folders to ignore in a project.

The structure of your project should be like this, we can delete some files which is not in our use

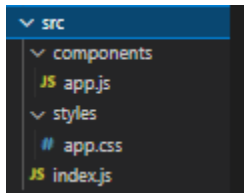
- Structure just after installing react-boiler-plate



Structure after deleting some files which are not in our use



Your src folder structure should be like these



Components - where all your react components will exist
Styles - where all your css files will exist

When we have multiple lines in react to written, why do we use ?

```
<></>
```

```
<div></div>
```

```
<React.Fragment></React.Fragment> ?
```

We can render a single element or multiple elements .

Rendering multiple elements will require

`<div></div>` or `<React.Fragment>` or `</React.Fragment>` or `<></>`

around the content as the render method will only render a single root node inside it at a time.

Why do we prefer

`<React.Fragment>` over `<div></div>`

in react rendering ?

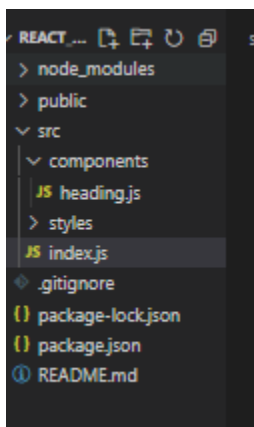
'Div' allows you to wrap or group multiple elements by adding an extra node to the DOM and hence it makes rendering slower

'React. Fragments' allows you to wrap or group multiple elements without adding an extra node to the DOM. and hence It's a tiny bit faster and has less memory usage .

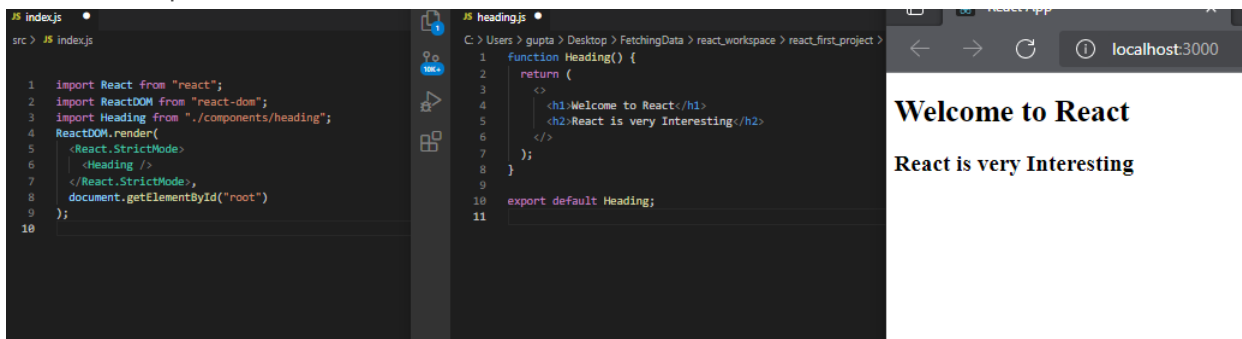
How to render Components in React?

Case 1: when we have a single component to render

Folder structure

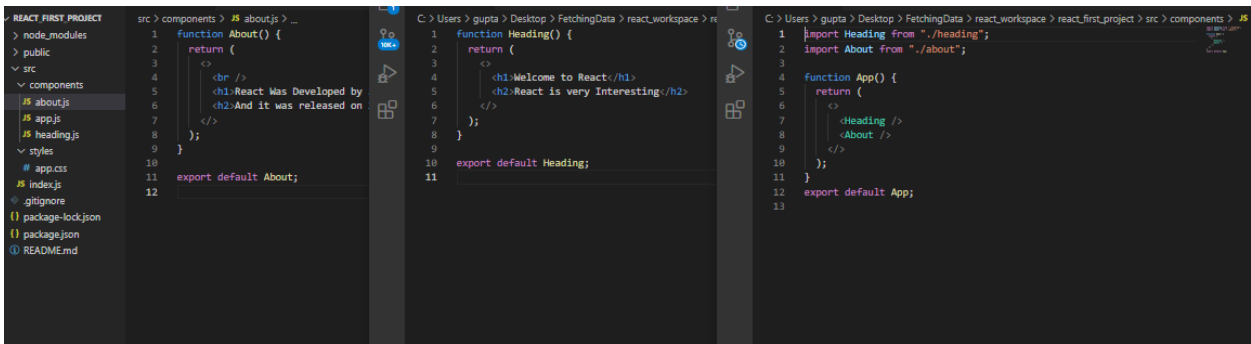


Code and output

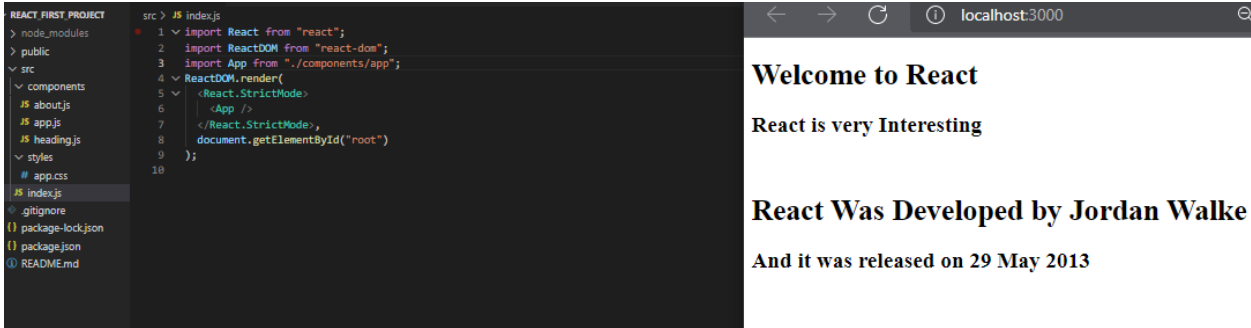


Case 2: when we have Multiple Components to render

about.js , heading.js, app.js files



index.js and output

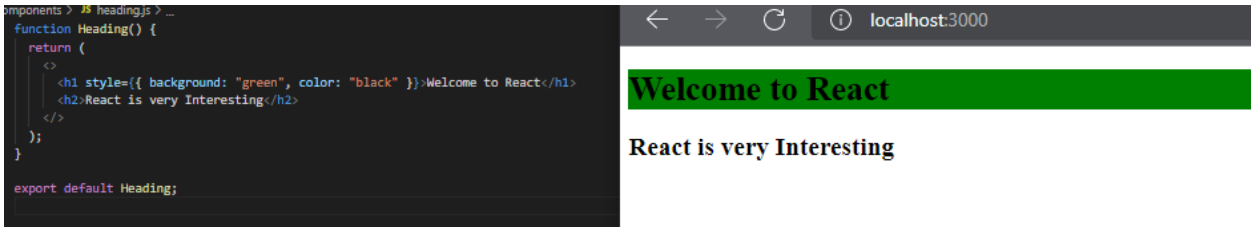


III. ADDING STYLES IN REACT

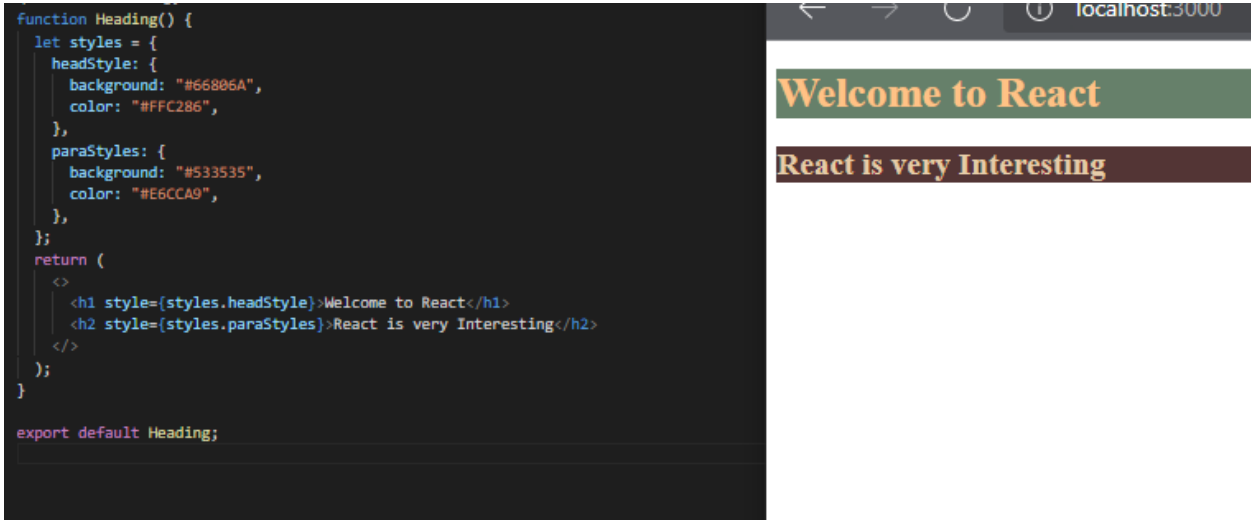
Types of CSS in React

Inline CSS	Internal CSS	External CSS
Css is written inside the element itself	Css is not written inside element but written within the same file	A separate file is created for css (saved with .css extension) And then import that css file in our .js file

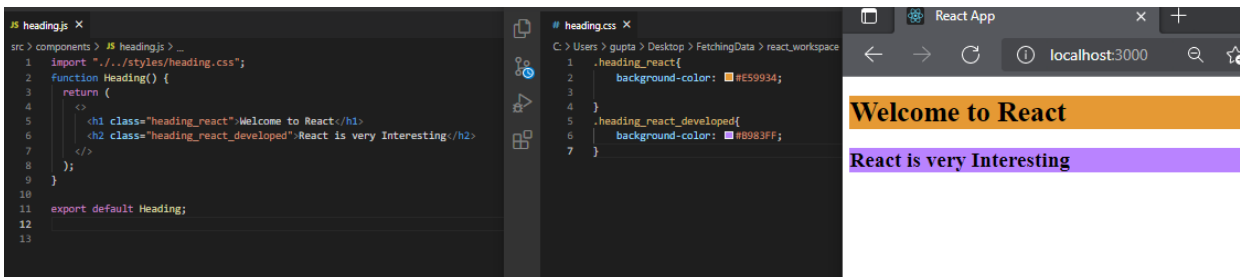
Inline css in React



Internal css In React



External css in React

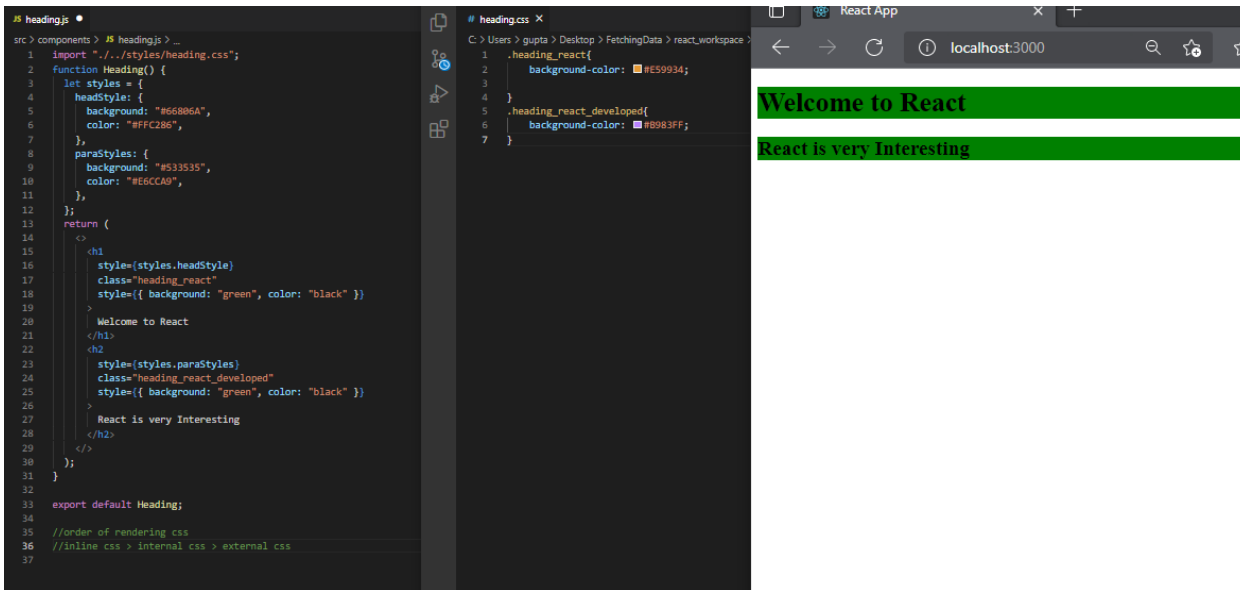


Order of rendering(executing) Inline css Internal css and External css in React

Inline css > Internal css > External css

Inline css will render first then Internal css and then External css

Inline css Internal css and External css



IV. CLASS COMPONENTS

Difference between Class Components and Functional Components?

SNO	Functional Component	Class Component
1	A functional component is just a plain JavaScript function that accepts an argument and returns a React element.	A class component requires you to extend from React. Component and create a render function that returns a React element.
3	Hooks can easily be used in Functional component	Hooks can't be used in Class Component
4	LifeCycle method not present in Functional Component	LifeCycle method like <code>ComponentDidMount</code> present in Class Component
5	this keyword is not present	this keyword present in Class Component

When do we prefer to use a class component over a function component?

If a component needs state or lifecycle methods, we should use the class component; otherwise, use the functional component.

However, after React 16.8, with the addition of Hooks, we can use lifecycle methods, and other features that were only available in the class component in our function component as well .

How to declare class components?

```
> components > JS heading.js > ...
1 import React, { Component } from "react";
2
3 class Heading extends Component{
4
5   constructor(){
6     super()
7     this.state = {
8       count:1
9     }
10  }
11  render(){
12    return(
13      <>
14        <h1>{this.state.count}</h1>
15        <h1>Class Component</h1>
16      </>
17    )
18  }
19 }
20
21 export default Heading
```

localhost:3000

1

Class Component

V. EVENT HANDLING

What are Event Handlers ?

Event handlers determine what action is to be taken whenever an event is fired

or

When an event occurs, you can create an event handler which is a piece of code that will execute to respond to that event

```
<button className="heading_click_btn" onClick={handleClick}>
  Click Me
</button>
```

For example: when the user clicks on the click Me button then the onClick event fired which will call the handleClick function

What is the difference between HTML and React event handling?

Point of Difference	HTML Event Handling	React Event Handling
Event Representation	<pre><button onclick="handleCl</pre> <p>In Html event is represented in lowercase Convention</p>	<pre><button className="heading_click Click Me </button></pre> <p>In React event is represented in camelCase Convention</p>

PreventDefault Behaviour

```
<form onSubmit="console.  
  <button type="submit">  
</form>
```

For preventDefault
behaviour in html we can
return 'false'

```
function handleClick(e) {  
  e.preventDefault();  
  console.log("You clicked btn.");  
}  
  
return (  
  <>  
    <button className="heading_cli  
      Click Me  
    </button>  
  </>  
>);
```

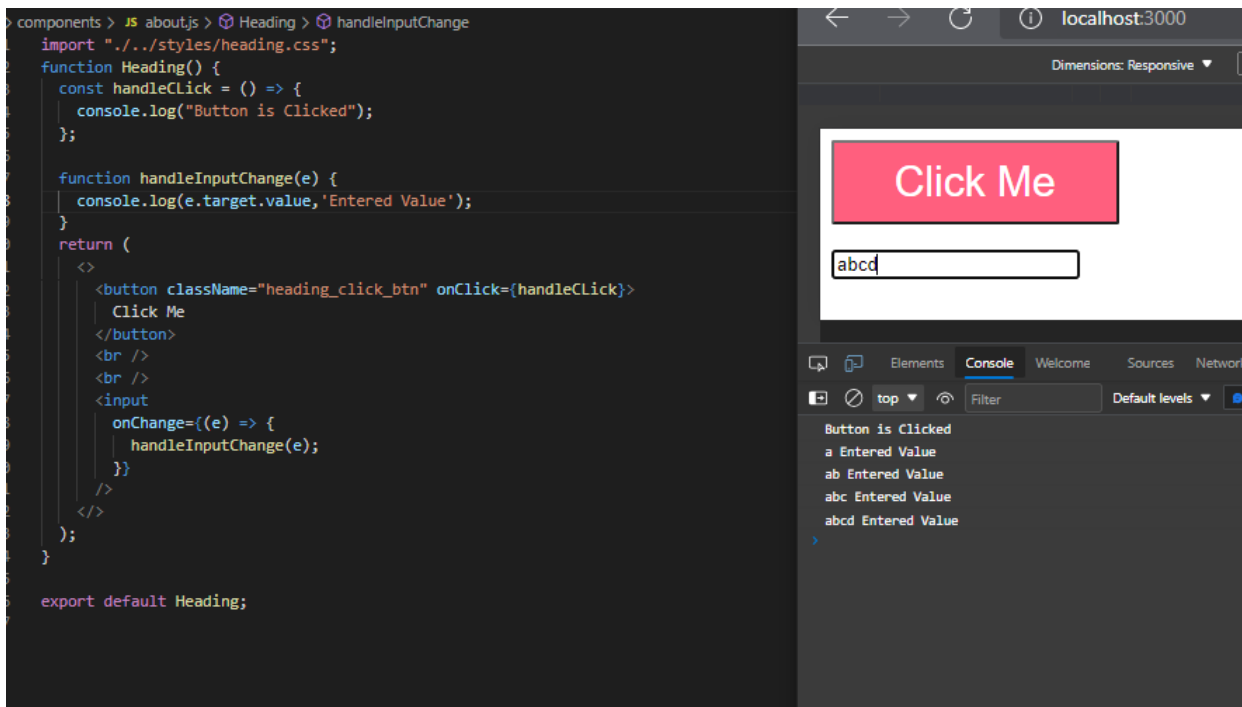
For preventDefault behaviour in
React we must call
event.preventDefault()

List of Events Supported By React ?

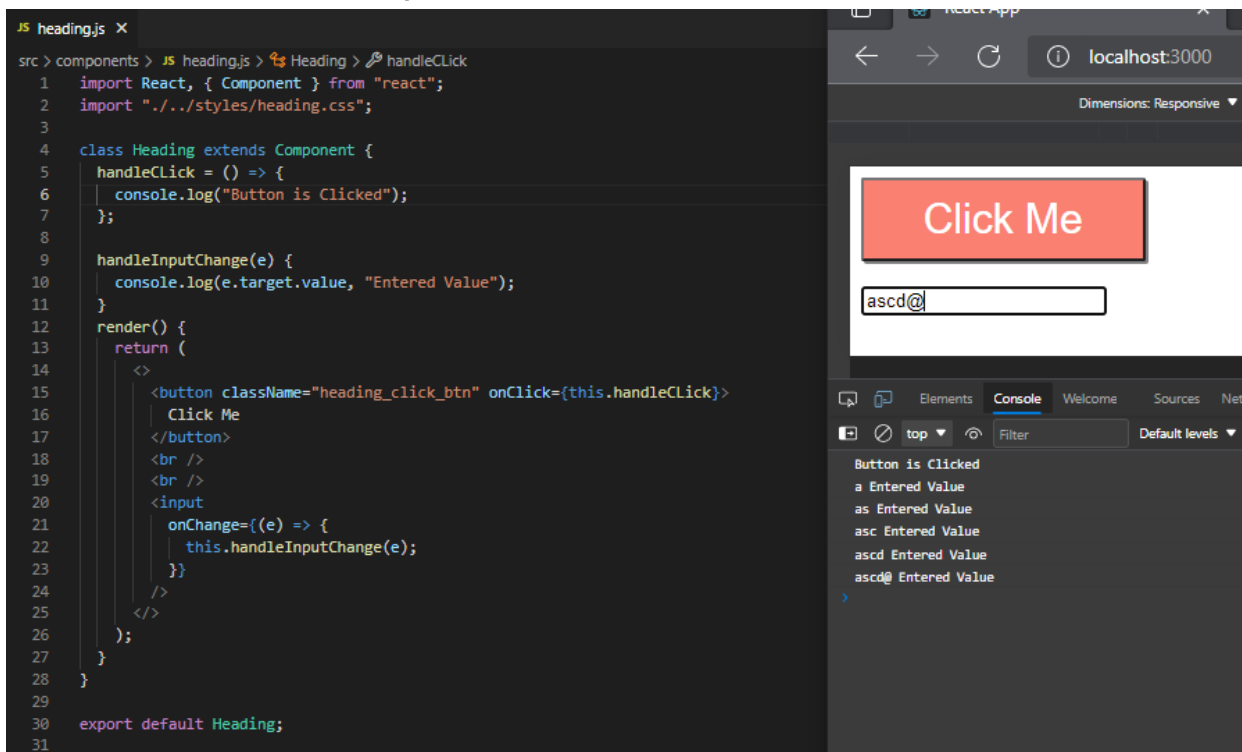
Notice that in React in event Names are declared in camelCase

Event group	Event group	Events supported by React
1	Mouse Events	onClick, onContextMenu, onDoubleClick, onDrag, onDragEnd, onDragEnter, onDragExit, onDragLeave, onDragOver, onDragStart, onDrop, onMouseDown, onMouseEnter, onMouseLeave, onMouseMove, onMouseOut, onMouseOver, onMouseUp
2	Keyboard Events	onKeyDown, onKeyPress, onKeyUp
3	Form events	onChange, onInput, onSubmit
4	Focus events	onFocus, onBlur
5	Clipboard events	onCopy, onCut, onPaste
6	Animation events	onAnimationStart, onAnimationEnd, onAnimationIteration
	Transition events	onTransitionEnd
8	Touch events	onTouchCancel, onTouchEnd, onTouchMove, onTouchStart
9	UI events	onScroll
10	Selection events	onSelect
11	Image events	onLoad, onError

Event declaration in Functional Component



Event Declaration in Class Components



VI. STATES IN CLASS COMPONENTS

What is a state in React?

The state of a component is an object that holds some information that may change over the lifetime of the component.

What is SetState(), Why should we not update the state directly?

- If we try to change the state without setState it will not work

```

let localCounter = this.state.counter;
this.state.counter = localCounter + 1;

```

- If we try to change the state with setState it will work

setState schedules an update to a component's state object. When the state changes, the component responds by re-rendering. because after setState, the render method is called.


```
let localCounter = this.state.counter;
this.setState({
  counter: localCounter + 1,
});
```

State change and setState in React class Component

```
components > JS heading.js > Heading > handleInputChange
import React, { Component } from "react";
import "../styles/heading.css";

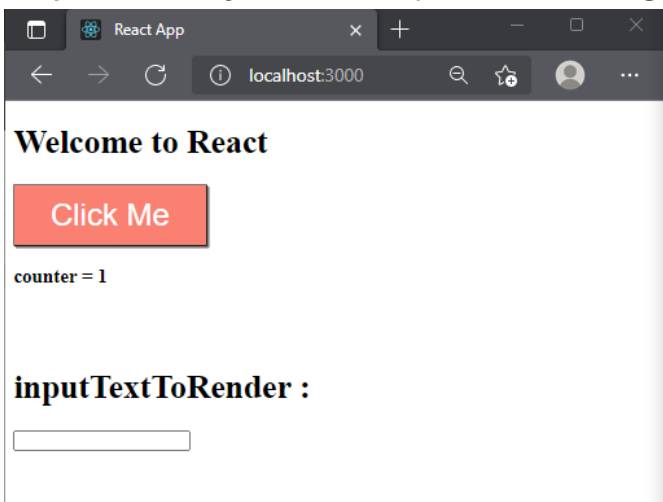
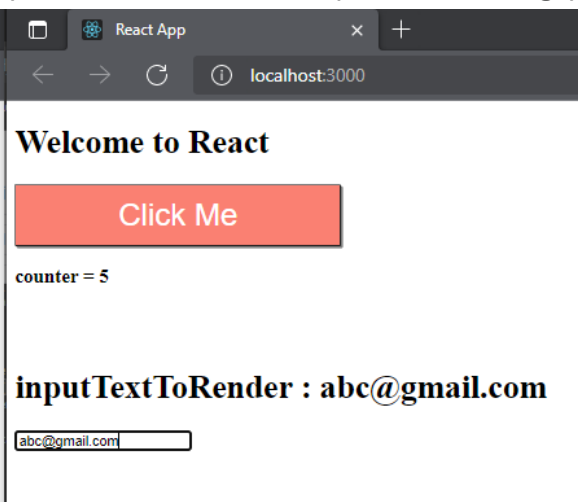
class Heading extends Component {
  constructor() {
    super();
    this.state = {
      headingText: "Welcome to React",
      counter: 1,
      inputTextToRender: "",
    };
  }
  handleClick = () => {
    let localCounter = this.state.counter + 1;
    this.setState({
      counter: localCounter + 1,
    });
  };

  handleInputChange(e) {
    this.setState({
      inputTextToRender: e.target.value,
    });
    console.log(e.target.value, "Entered Value");
  }

  render() {
    return (
      <>
        <h1>{this.state.headingText}</h1>
        <button className="heading_click_btn" onClick={this.handleClick}>
          Click Me
        </button>
        <h3> counter = {this.state.counter}</h3>
        <br />
        <br />
        <h1>inputTextToRender : {this.state.inputTextToRender}</h1>
        <input
          onChange={(e) => {
            this.handleInputChange(e);
          }}
        />
      </>
    );
  }
}

export default Heading;
```

Output Before any event is fired(before state change) and After Event is Fired (after state change)

Before Event (Before State Change)	After Event (After State Change)
 <p>Welcome to React</p> <p>Click Me</p> <p>counter = 1</p> <p>inputTextToRender :</p> <input type="text"/>	 <p>Welcome to React</p> <p>Click Me</p> <p>counter = 5</p> <p>inputTextToRender : abc@gmail.com</p> <input type="text" value="abc@gmail.com"/>

VII. PROPS IN REACT

What are Props in React?

"Props" stands for properties. It is a special keyword in React which is used for passing data from one component to another component

What is the difference between Props and State in React?

Sno	Props	State
1	Props allow you to pass data From One Component to another	state holds information about the components.
2	Props can be accessed by child Components	State cannot be accessed by Child Components, Scope limited to one but can be passed to children as Props
3	Functional component and Class Component can have props	Class Component Can only have state
4	Props are external and controlled by whatever renders the component	The state is internal and controlled by React Component itself
5	Props passed in Parent Component <pre><Heading courseList={this.state.courseList} /></pre>	state used in class component <pre>let localCounter = this.state.counter; this.setState({ counter: localCounter + 1, });</pre>

Passing Props in Class Component app.js,heading.js,course.json

```
src > components > app.js > App > state
1 import React, { Component } from "react";
2 import CourseJson from "../course.json";
3 import Heading from "../heading";
4
5 class App extends Component {
6   state = {
7     courseList: CourseJson,
8   };
9
10  render() {
11    return (
12      <>
13        <Heading courseList={this.state.courseList} />
14      </>
15    );
16  }
17 }
18
19 export default App;
20
```

```
src > components > heading.js
1 import React, { Component } from "react";
2
3 class Heading extends Component {
4   constructor(props) {
5     super(props);
6   }
7
8   render() {
9     return (
10       <>
11         <h1>Passing Props in Class Components</h1>
12         {this.props.courseList.map((item) => {
13           return (
14             <div>
15               <h1>{item.course}</h1>
16               <h2>{item.CourseDescription}</h2>
17             </div>
18           );
19         })}
20       </>
21     );
22   }
23 }
24
25 export default Heading;
```

```
src > components > course.json
1 [
2   {
3     "course": "Course1",
4     "CourseDescription": "This is really Good Course1"
5   },
6   {
7     "course": "Course2",
8     "CourseDescription": "This is really Good Course2"
9   },
10  {
11    "course": "Course3",
12    "CourseDescription": "This is really Good Course3"
13  }
14 ]
```

Output

```
localhost:3000
```

Course1

This is really Good Course1

Course2

This is really Good Course2

Course3

This is really Good Course3

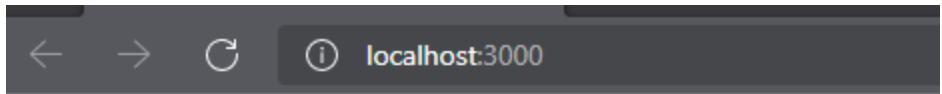
Passing Props in Functional Component app.js,heading.js,course.json

```
app.js
1 import React, { useState } from "react";
2 import Coursejson from "../course.json";
3 import Heading from "./heading";
4
5 function App() {
6   const [courselist, setCourseList] = useState(Coursejson);
7   return (
8     <>
9     <Heading courseList={courselist} />
10    </>
11  );
12 }
13
14 export default App;
```

```
heading.js
1 import React from "react";
2
3 function Heading(props) {
4   return (
5     <>
6     <h1>Passing Props in Functional Component</h1>
7     {props.courselist.map((item) => {
8       return (
9         <>
10         <h1>{item.course}</h1>
11         <h2>{item.CourseDescription}</h2>
12         </>
13       )
14     })}
15   );
16 }
17
18 export default Heading;
```

```
course.json
1 [
2   {
3     "course": "Course1",
4     "CourseDescription": "This is really Good Course1"
5   },
6   {
7     "course": "Course2",
8     "CourseDescription": "This is really Good Course2"
9   },
10  {
11    "course": "Course3",
12    "CourseDescription": "This is really Good Course3"
13  }
14 ]
```

Output



Passing Props in Functional Component

Course1

This is really Good Course1

Course2

This is really Good Course2

Course3

This is really Good Course3

Components and Routing

I. REACT LIFECYCLE OVERVIEW

What is the LifeCycle in React Components?

Lifecycle methods are special methods built-in to React, used to operate on components throughout their duration in the DOM.

What is mounting and unmounting in React Lifecycle?

mounting - Mounting means Putting element in DOM

Unmounting - Unmounting means component is removed from the DOM

What are the different phases of the component lifecycle?

1. Mounting: The component is ready to mount in the browser DOM. This phase covers initialization from

The phase covers initialization from

- constructor()
- getDerivedStateFromProps()
- render()
- componentDidMount()

2. Updating: In this phase, the component gets updated by, sending the new props and updating the state from setState()

This phase covers initialization From

The phase covers initialization from

- getDerivedStateFromProps()
- shouldComponentUpdate()
- render()
- getSnapshotBeforeUpdate()
- componentDidUpdate()

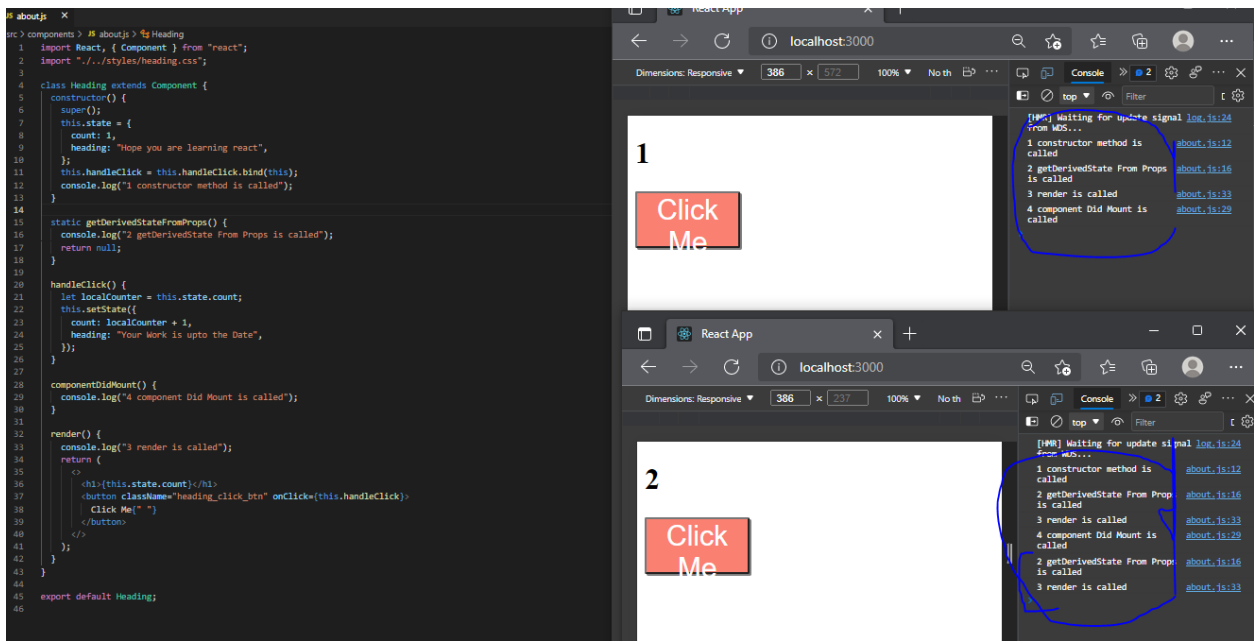
3. Unmounting: In this phase, the component is not needed and gets unmounted from the browser DOM.

The phase covers initialization from

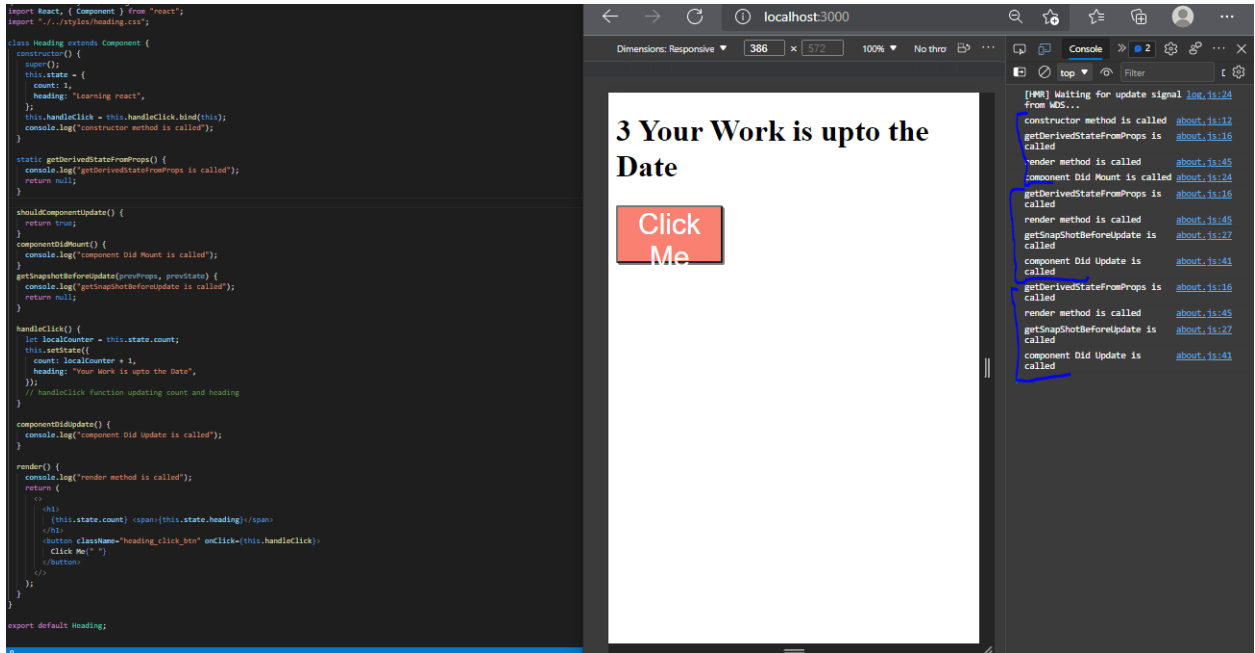
- componentWillUnmount()

Component is Mounting Code

constructor() and componentDidMount() called only the First time when the component is mounted in DOM the First time



Component is Updated code



Component is unmounted Code

```
components > # about.js > Heading > componentDidMount

class Heading extends Component {
  constructor() {
    super();
    this.state = {
      count: 1,
      heading: "Learning react",
      isBannerVisible : true
    };
    this.handleClick = this.handleClick.bind(this);
    console.log("constructor method is called");
  }

  static getDerivedStateFromProps() {
    console.log("getDerivedStateFromProps is called");
    return null;
  }

  shouldComponentUpdate() {
    return true;
  }

  componentDidMount() {
    console.log("component Did Mount is called");
  }

  getSnapshotBeforeUpdate(prevProps, prevState) {
    console.log("getSnapshotBeforeUpdate is called");
    return null;
  }

  handleClick() {
    let localCounter = this.state.count;
    this.setState({
      count: localCounter + 1,
      heading: "Your Work is upto the Date",
      isBannerVisible: false
    });
    // handleClick function updating count ,heading and isBannerVisible
  }

  componentDidUpdate() {
    console.log("component Did Update is called");
  }

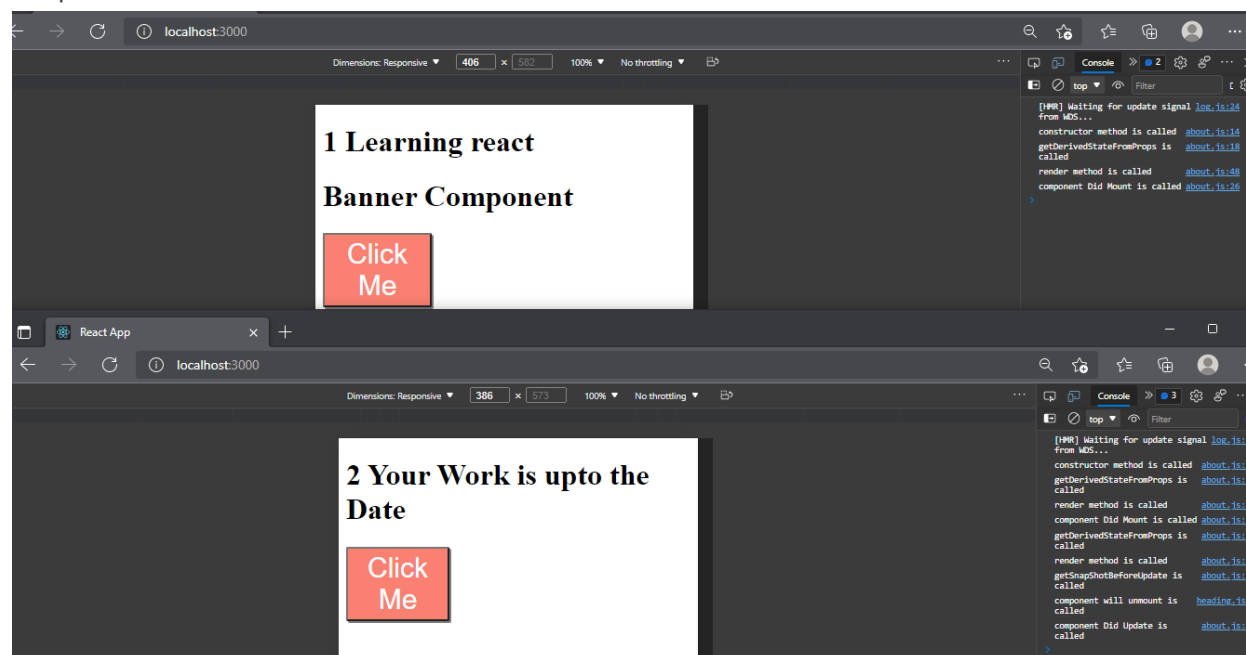
  render() {
    console.log("render method is called");
    return (
      <>
        <h1>
          {this.state.count} <span>{this.state.heading}</span>
        </h1>
        {this.state.isBannerVisible?<Banner />:null}

        <button className="heading_click_btn" onClick={this.handleClick}>
          Click Me(" ")
        </button>
      </>
    );
  }
}

export default Heading;
```

Output For Component is unMounted code

As the user click on the Click Me button isbannerVisible sets to be false, hence component will unmount and componentWillUnmount method will be called



II. REACT LIFECYCLE METHODS IN DETAIL

1. Constructor() :

constructor(props)

In React Constructor are used for two purpose

- (i) Initializing local state by assigning an object to this.state
- (ii) Binding event handler methods to an instance.

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

class Heading extends Component {
  constructor() {
    super();
    this.state = {
      count: 1,
    }; //Initializing this.state in constructor
    this.handleClick = this.handleClick.bind(this); //binding handleClick function ,
    // if we will not bind then
    //Cannot read properties of undefined (reading 'state') error will occur
  }

  handleClick() {
    let localCounter = this.state.count;
    this.setState({
      count: localCounter + 1,
    });
  }

  render() {
    return (
      <>
        <h1>{this.state.count}</h1>
        <button onClick={this.handleClick}>Click Me </button>
      </>
    );
  }
}

export default Heading;
```

1. getDerivedStateFromProps():

```
static getDerivedStateFromProps(props, state)
```

Its main function is to change the initial state based on the value of props .It is invoked right before calling the render method, both on the initial mount and on subsequent updates. It should return an object to update the state, or null to update nothing.

3. render():

```
render()
```

(A) The render() function does not modify the component state, it returns the same result each time it's invoked and is responsible for describing the view to be rendered to the browser window.

(B) render() is called by React at various app stages, generally when the React component is first instantiated, or when there is a new update to the component state.

Note : render() will not be invoked if shouldComponentUpdate() returns false.

3. componentDidMount():

```
componentDidMount()
```

componentDidMount is invoked immediately after a component is mounted

If we need to load data as soon as the app open, this is a good place to initiate the network request

4. shouldComponentUpdate():

```
shouldComponentUpdate(nextProps, nextState)
```

(A) If there is an update happening and when there is a certain value and props and we don't want that update to happen, then we use componentShouldUpdate(). This kind of scenario is not very often, if we have very specific requirements then we can use it

(B) Return value: by default it returns true and if it returns false then render() and getSnapshotBeforeUpdate() ,componentDidUpdate() method do not get invoked.

(C) The shouldComponentUpdate method is majorly used for optimizing the performance and to increase the responsiveness of the website but do not rely on it to prevent rendering as it may lead to bugs.

(D) If there is any update happening It doesn't call in the first slot, it gets called when the component is getting updated

CASE : When shouldComponentUpdate method will returning null or false

Then render() and getSnapshotBeforeUpdate() ,componentDidUpdate() will not work although state change will occur on click of click me button .

The screenshot shows a code editor on the left with the following code for a Banner Component:

```
class Banner extends Component {
  constructor() {
    super();
    this.state = {
      count: 1,
      heading: "Learning react",
      isVisible: true,
    };
    this.handleClick = this.handleClick.bind(this);
    console.log("constructor method is called");
  }

  static getDerivedStateFromProps(prevProps, prevState) {
    console.log("getDerivedStateFromProps is called");
    console.log(prevState.count, "COUNT");
    return null;
  }

  shouldComponentUpdate() {
    return false; //shouldComponentUpdate returning false
  }

  componentDidMount() {
    console.log("component Did Mount is called");
    getSnapshotBeforeUpdate(prevProps, prevState) {
      console.log("getSnapshotBeforeUpdate is called");
      return null;
    }
  }

  handleClick() {
    let localCounter = this.state.count;
    this.setState({
      count: localCounter + 1,
      heading: "Your Work is upto the Date",
      isVisible: false,
    });
    // handleClick function updating count ,heading and isVisible
  }

  componentDidUpdate() {
    console.log("component Did Update is called");
  }

  render() {
    console.log("render method is called");
    return (
      <div>
        <h1>1 Learning react</h1>
        <h2>Banner Component</h2>
        <button className="heading_click_btn" onClick={this.handleClick}>Click Me</button>
      </div>
    );
  }
}
```

The browser on the right shows the rendered component with the heading "1 Learning react" and a "Click Me" button. The console log shows the following sequence of events:

- constructor method is called
- getDerivedStateFromProps is called
- 1 'COUNT'
- render method is called
- component Did Mount is called
- getDerivedStateFromProps is called
- 2 'COUNT'
- getDerivedStateFromProps is called
- 3 'COUNT'

CASE : When shouldComponentUpdate method will returning true

Then render() and getSnapshotBeforeUpdate() , componentDidUpdate() will work

The screenshot shows a code editor on the left with the following code for a Banner Component:

```
import React, { Component } from "react";
import "./styles/header.css";

class Banner extends Component {
  constructor() {
    super();
    this.state = {
      count: 1,
      heading: "Learning react",
    };
    this.handleClick = this.handleClick.bind(this);
    console.log("constructor method is called");
  }

  static getDerivedStateFromProps(prevProps, prevState) {
    console.log("getDerivedStateFromProps is called");
    console.log(prevState.count, "COUNT");
    return null;
  }

  shouldComponentUpdate() {
    return true; //shouldComponentUpdate returning true
  }

  componentDidMount() {
    console.log("component Did Mount is called");
  }

  getSnapshotBeforeUpdate(prevProps, prevState) {
    console.log("getSnapshotBeforeUpdate is called");
    return null;
  }

  handleClick() {
    let localCounter = this.state.count;
    this.setState({
      count: localCounter + 1,
      heading: "Your Work is upto the Date",
    });
    // handleClick function updating count ,heading and isVisible
  }

  componentDidUpdate() {
    console.log("component Did Update is called");
  }

  render() {
    console.log("render method is called");
    return (
      <div>
        <h1>3 Your Work is upto the Date</h1>
        <button className="heading_click_btn" onClick={this.handleClick}>Click Me</button>
      </div>
    );
  }
}
```

The browser on the right shows the rendered component with the heading "3 Your Work is upto the Date" and a "Click Me" button. The console log shows the following sequence of events:

- constructor method is called
- getDerivedStateFromProps is called
- 1 'COUNT'
- render method is called
- component Did Mount is called
- getDerivedStateFromProps is called
- 2 'COUNT'
- render method is called
- getSnapshotBeforeUpdate is called
- component Did Update is called
- getDerivedStateFromProps is called
- 3 'COUNT'
- render method is called
- getSnapshotBeforeUpdate is called
- component Did Update is called

5. `getSnapshotBeforeUpdate()`

```
getSnapshotBeforeUpdate(prevProps, prevState)
```

(A) If there is anything that we want to save and want to use that particular value later on and majorly we will be using that value in `componentDidUpdate` then we will use it

(B) It is used to store the previous values of the state after the DOM is updated.

6. `componentDidUpdate()`

```
componentDidUpdate(prevProps, prevState, snapshot)
```

prevProps: Previous props passed to the component

prevState: Previous state of the component

snapshot: Value returned by `getSnapshotBeforeUpdate()` method

(A) This method allows us to execute the React Code when the component is updated.

(B) All the network requests that are to be made when the props passed to the component changes are coded here

Note Whenever we need to change state in `componentDidUpdate`

(A) change the state with some condition, if we will not change the state with the condition, then the infinite loop will start running (too many renderings) It is just because, as soon as any update happens `componentDidMount` is called, which leads to calls itself infinite time.

```
componentDidUpdate() {  
  let localCounter = this.state.count  
  this.setState({  
    count: localCounter + 1 // updating count value in componentDidUpdate with some condition  
  }) // if we will not update it with condition then infinite times componentDidUpdate will call itself  
  console.log("component Did Update is called");  
}
```

(B) Another one you can set some condition in `shouldComponentUpdate` and then return false, it prevents loop to run infinite times.

```
shouldComponentUpdate() {  
  if (this.state.count >= 10) {  
    return false  
  } // Another option we can set some condition in shouldComponentUpdate  
  // so that it will prevent from too many rendering or infinite time loop is running  
  return true;  
}
```

III. REACT ROUTING INTRODUCTION

What is a React Router ?

(A) React Router is a powerful routing library built on top of React that helps to flow your application incredibly quickly, while keeping the URL in sync with what's being displayed on the page.

(B) When a user types a specific URL into the browser, and if this URL path matches any 'route' inside the router file, the user will be redirected to that particular route

(C) React Router Dom is used to build applications that have many pages or components but the page is never refreshed instead the content is dynamically fetched based on the URL

What is the difference between `react-router` and `react-router-dom` ?

In React Router v4, the React Router was broken into two: `react-router` and `react-router-dom`. `react-router` is the core, and `react-router-dom` is the core plus the React Router elements such as

`<BrowserRouter>` and `<NavLink>`

Since `react-router-dom` is like a super-set of `react-router`, you only need to import `react-router-dom`.

What is the difference between `react-router-DOM` and `React-Router Native` ?

React Router DOM is for web applications and React Router Native is for mobile applications made with React Native. (or build for mobile apps for ios and Android)

IV. BROWSER ROUTER

What is BrowserRouter ?

BrowserRouter is responsible for understanding the url and then going ahead and ensuring that ui that we have or component we have is as per that particular url

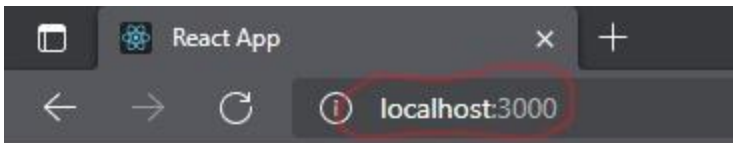
Installation

npm i react-router-dom

How to define routing in React ?

```
src > components > .js app.js > App > render
1 import React, { Component } from "react";
2 import { BrowserRouter, Route, Routes } from "react-router-dom";
3 import Header from "./header";
4 import About from "./about";
5
6 import Footer from "./Footer";
7 import Home from "./home";
8 class App extends Component {
9   render() {
10     return (
11       <BrowserRouter>
12         <Header />
13         <Routes>
14           <Route path="/" element={<Home />} />
15           <Route path="/about" element={<About />} />
16         </Routes>
17         <Footer />
18       </BrowserRouter>
19     );
20   }
21 }
22 export default App;
23
src > components > .js header.js > Header
1 import React from "react";
2 import './styles/heading.css';
3 function Header(){
4   return(
5     <h1
6       className="heading_react_dev"
7     >Header</h1>
8   )
9 }
10 export default Header;
11
src > components > .js home.js > default
1 import React from "react";
2 function Home(){
3   return(
4     <h1>Home</h1>
5   )
6 }
7 export default Home;
8
src > components > .js footer.js > default
1 import React from "react";
2 import './styles/heading.css';
3 function Footer(){
4   return(
5     <h1
6       className="heading_react_dev"
7     >Footer</h1>
8   )
9 }
10 export default Footer;
11
```

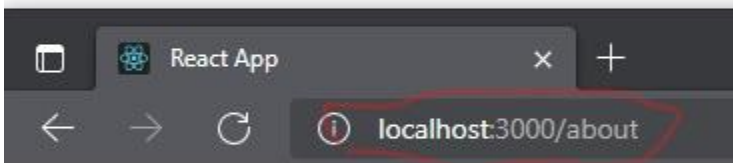
output



Header

Home

Footer



Header

About

Footer

V. SWITCH IN REACT

If you are using v5 of react-router then you need to use switch and exact

What is exact in React ?

React router does partial matching, so /course partially matches /course/zenith , so it would incorrectly return the Users route
The exact param disables the partial matching for a route and makes sure that it only returns the route if the path is an EXACT match to the current url

Syntax

```
<Route path="/course/zenith" element={<CourseZenith exact />}></Route>  
<Route path="/course" element={<Course exact />}></Route>
```

What is Switch in React ?

The `Switch` component will only render the first route that matches/includes the path. Once it finds the first route that matches the path, it will not look for any other matches. Not only that, it allows for nested routes to work properly, which is something that Router will not be able to handle.

What is the difference between switch and exact ?

To render particular component when there is exact match , if anything apart from url it should not match and component will not render

For these we have to write exact again and again for every route , its alternative is switch

Switch Once finds the first route that matches the path, it will not look for any other matches

Note - When we are using switch put more specific url on top

There are changes in react router from v5 to v6 , if you are using v6 then you need not use switch and exact

A FEW IMPORTANT THINGS TO NOTICE ABOUT V6

1. React Router v6 introduces a `Routes` component that is kind of like `Switch`, but a lot more powerful.

The main advantages of `Routes` over `Switch` is

You may put your routes in whatever order you wish

and the router will automatically detect

the best route for the current URL. This prevents bugs due to manually putting routes in the wrong order in a `<Switch>`

2. `<Route exact>` is gone.

3. `<Route children>` from the v5 changed to `<Route element>`

VI. LINKS

What is Link in React ?

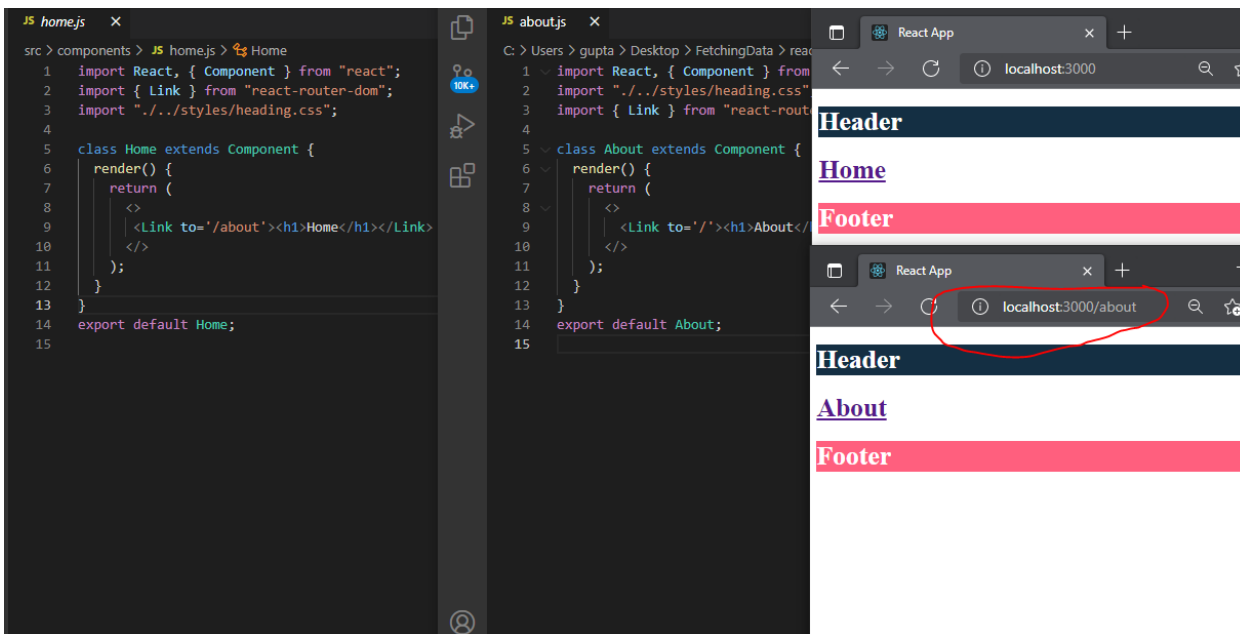
Link is a primary way to allow users to navigate around your application from page to another without loading

The difference between link and anchor tag is that

- **anchor tag** when we navigate to another page there is a loading happen
- **Link** when we navigate to another page through link the page will not load.

How to use links in React ?

Initially we are on Home page , on click of Home we move to about page and there is a change in url(localhost:3000/about)



How to pass parameters in react-router-dom Link and how to access them on another component?

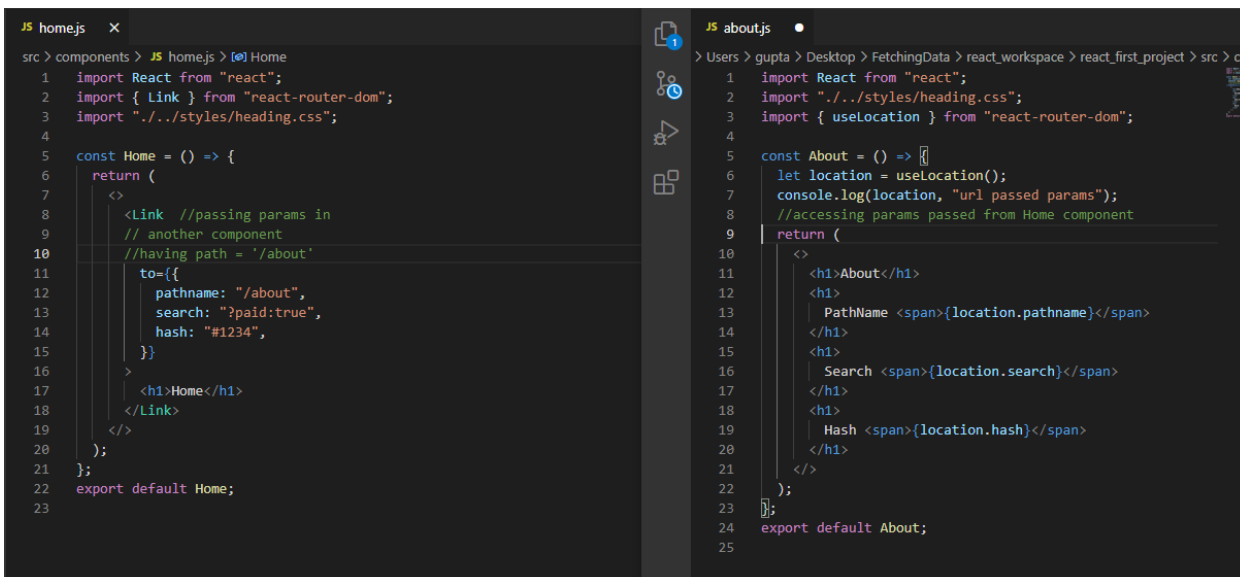
Step 01 How to declare parameters inside link

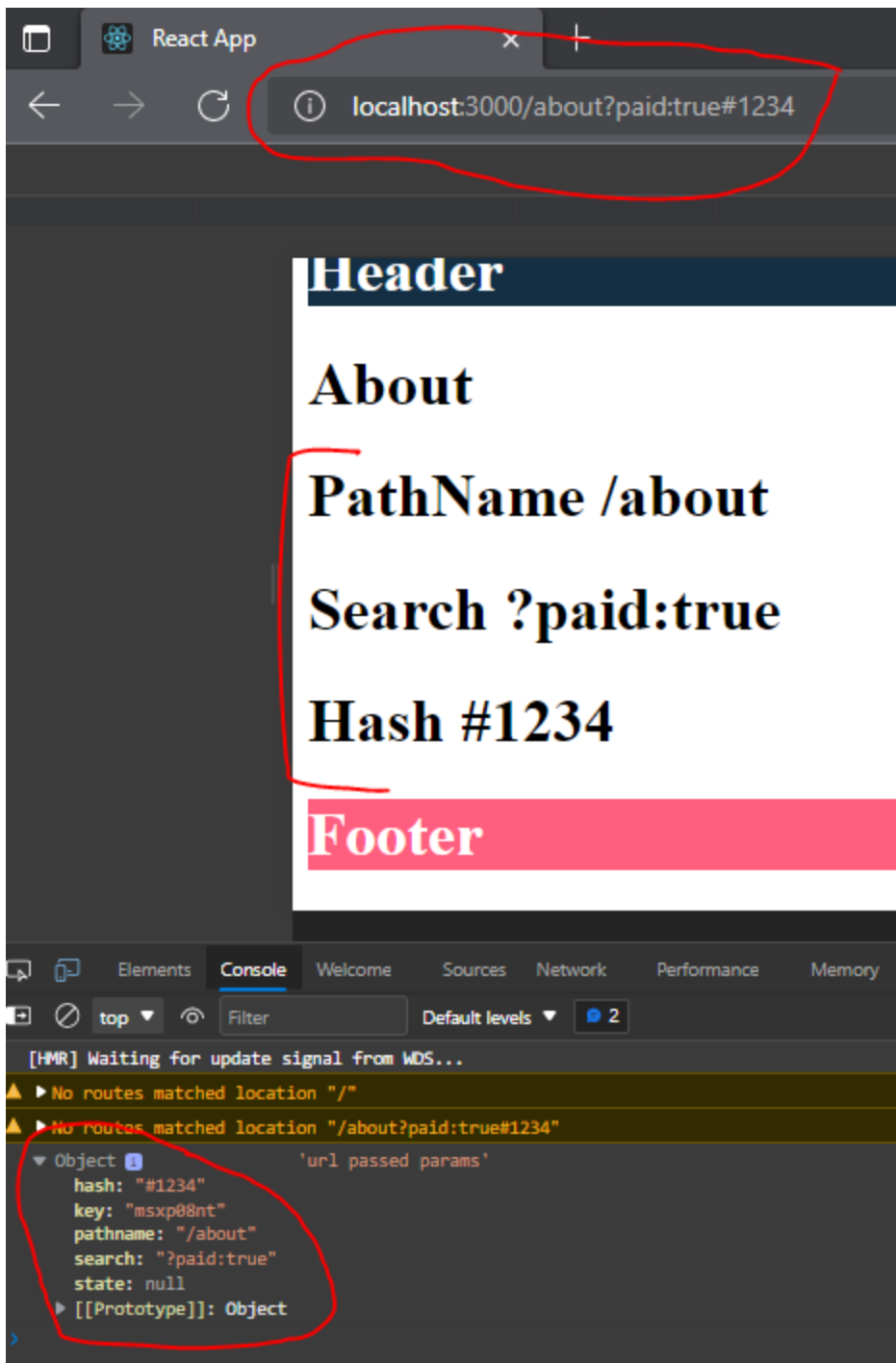
1. `pathname`: A string representing the path to link to.
2. `search`: A string representation of query parameters.
3. `hash`: A hash to put in the URL, eg `#123`
4. `state/any name`: Inside State to value.

Step 02 How to access Link Component State we Declare previous.

`useLocation()` : The `useLocation` hook is a function that returns the location object that contains information about the current URL. Whenever the URL changes, a new location object will be returned

Passing params through url and accessing them





VII. NAVLINKS

What is the difference between NavLink and Links in React ?

The Link component is used to navigate the different routes on the site.

But NavLink is used to add the style attributes to the active routes

What is activeClassName and activeStyle in NavLink ?

activeStyle = The Style is apply to element when element is active(when user click on that element)

activeClass = The class is apply to element when element is active(when user click on that element)

VIII. ROUTING PARAMS

What are params?

Params short for 'parameter'.

A params is a key-value pair that is encoded in an HTTP request.

There are three kinds of params: user-supplied parameters , routing parameters , and default parameters .

What are Routing params?

Route params are parameters whose values are set dynamically in a page's URL

Why do we need Routing parameters in react ?

react-router's parameterized routes can improve the specificity and behavior of routes in our React application . Based on the data that we fetch through URL we go ahead and take data from the database

How to pass Routing Params?

For Routing params to pass we have to define routing in these way

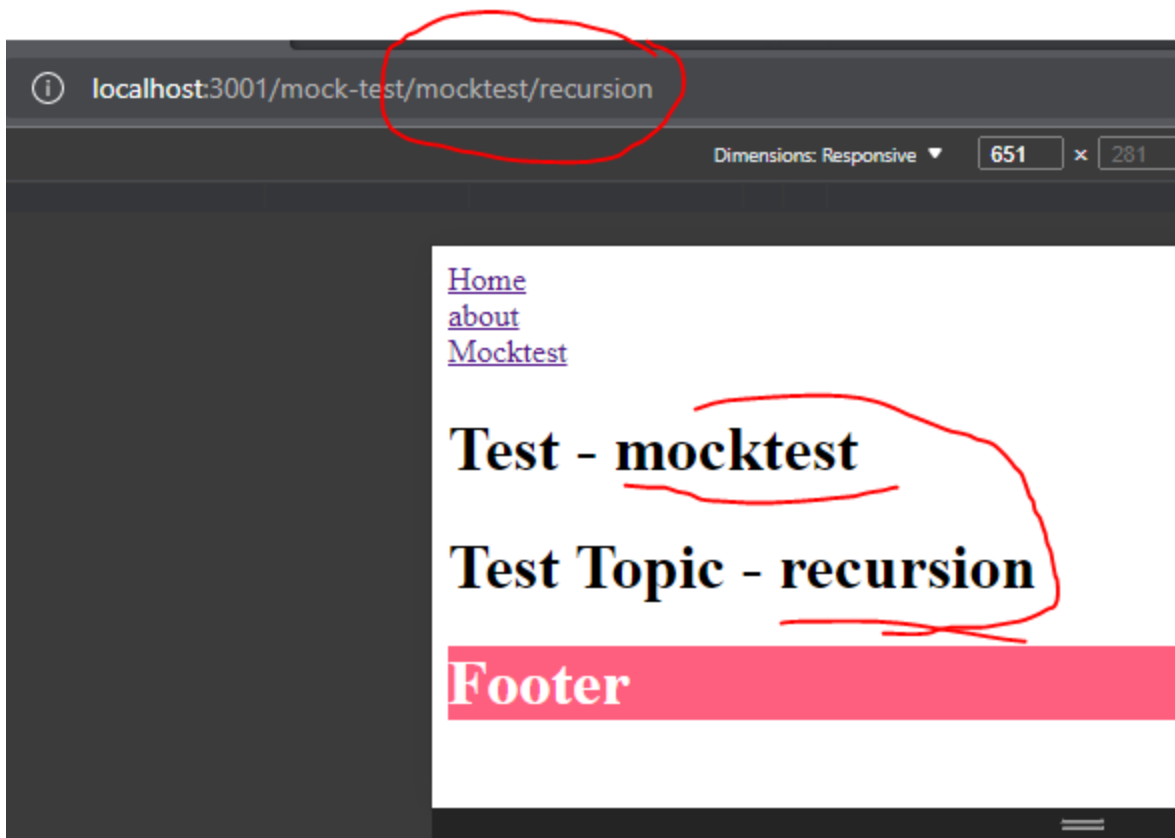
```
<Route
  path="/mock-test/:type/:topic"
  /* In these path we will pass routing params */
  element={<MocktestDetails />}
  exact
/>
```

Complete Code For Routing Params

```
JS app.js x
src > components > JS app.js > App > render
1 import React, { Component } from "react";
2 import { BrowserRouter, Route, Routes } from "react-router-dom";
3 import Header from "../header";
4 import MocktestDetails from "../mocktestData/mocktestDetails";
5 import Mocktest from "../mocktestData/mocktest";
6
7 import Footer from "../footer";
8 import Home from "../home";
9 class App extends Component {
10   render() {
11     return (
12       <BrowserRouter>
13       <Header />
14       /* when we want header to be shown on all pages */
15       <Routes>
16       <Route path="/" element={<Home />} />
17       <Route path="/mock-test" element={<Mocktest />} />
18       /* In these path we will pass routing params */
19       <Route path="/mock-test/:type/:topic" element={<MocktestDetails />} />
20       </Routes>
21       <Footer />
22     </BrowserRouter>
23   );
24 }
25
26
27 export default App;
28
```

```
JS mocktestDetails.js x
src > components > mocktestData > JS mocktestDetails.js > MocktestDetails
1 import React from "react";
2 import { useParams } from "react-router";
3 const MocktestDetails = () => {
4   let params = useParams();
5   console.log(params, "PARAMS");
6   return (
7     <>
8     <h1>
9     <span>Test -</span> <span>{params.type}</span>
10    </h1>
11    <h1>
12    <span>Test Topic - </span>
13    <span>{params.topic}</span>
14    </h1>
15    </>
16  );
17 };
18 export default MocktestDetails;
19
```

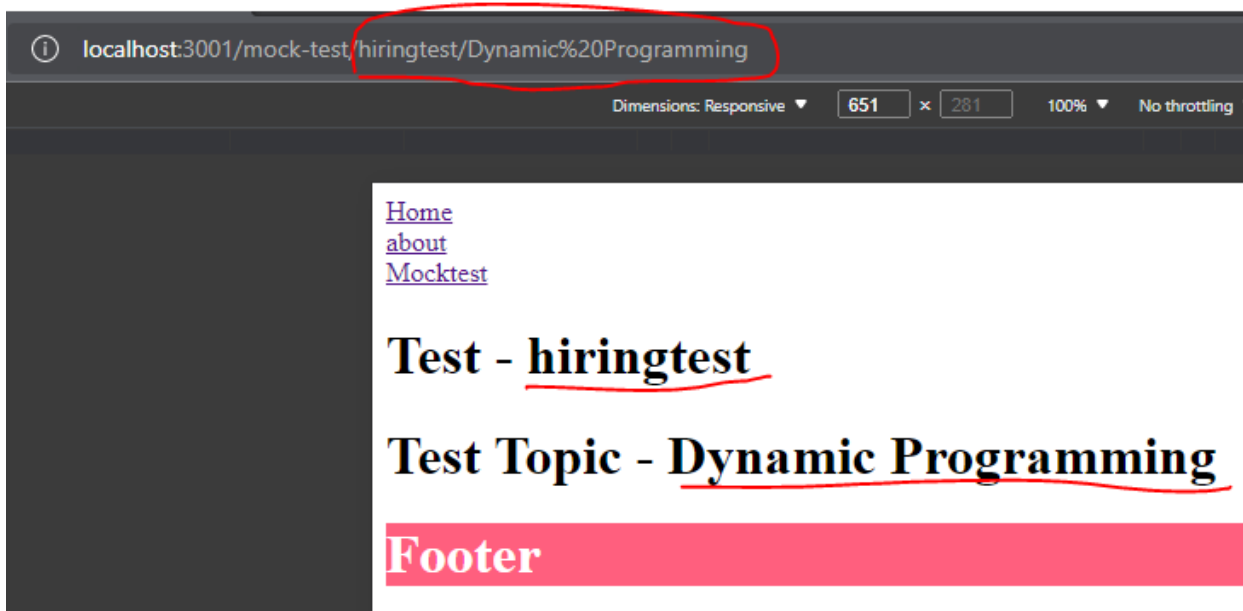
Output



Params On Console

```
{type: 'mocktest', topic: 'recursion'} 'PARAMS'  
  topic: "recursion"  
  type: "mocktest"  
  [[Prototype]]: Object
```

Changing Routing Params



Params on Console

```
{type: 'hiringtest', topic: 'Dynamic Programming'} 'PARAMS'  
  topic: "Dynamic Programming"  
  type: "hiringtest"  
  [[Prototype]]: Object
```

We can also define some text on a particular routing if we don't want to render a complete Component

The Syntax For that is

```
<Routes>
  <Route
    path="/about"
    element={
      <div>
        <h1>About Us </h1>
      </div>
    }
  />
</Routes>
```

If we don't have any provision and don't have any component to handle a particular url, so in that case either we will have blank space or 404 page Not Found

so for these we have a route that handle all the routes that don't exist on page , so if anyone puts wrong url we can handle that

How to define route for pageNotFound ?

Syntax

```
<Route path="*" element={<PageNotFound />}></Route>
{/* Route when page Not Found*/}
```

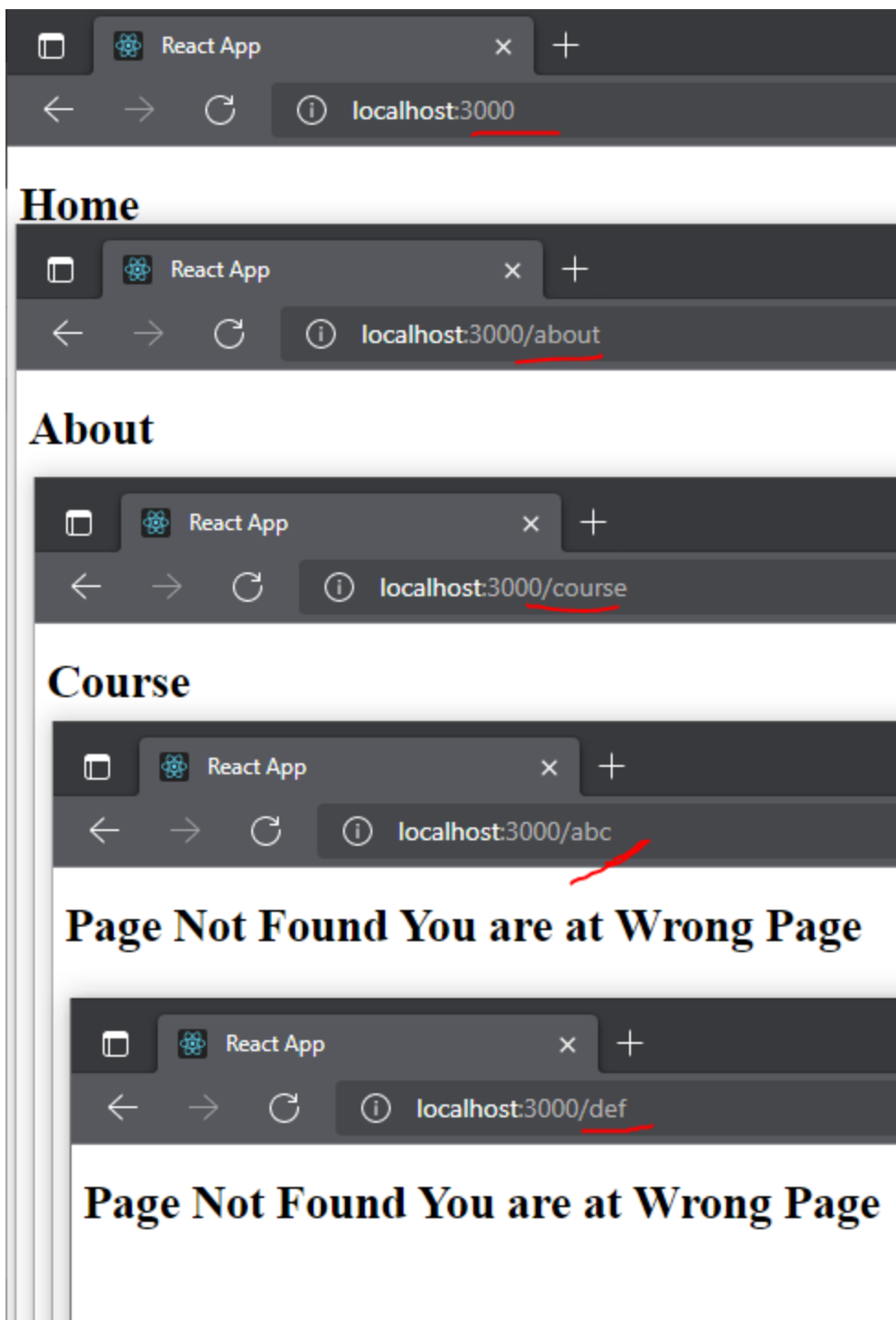
Code For Page Not Found

```
import Home from "../../reactrouter/heading";
import About from "../../reactrouter/about";
import Course from "../../reactrouter/course";
import CourseZenith from "../../reactrouter/courseZenith";
import PageNotFound from "../../reactrouter/pagenotfound";
import { BrowserRouter, Route, Routes } from "react-router-dom";

const App = () => {
  return (
    <>
      <BrowserRouter>
        <Routes>
          <Route path="/" element={<Home />}></Route>
          <Route path="/about" element={<About />}></Route>
          <Route path="/course/zenith" element={<CourseZenith exact />}></Route>
          <Route path="/course" element={<Course exact />}></Route>
          <Route path="*" element={<PageNotFound />}></Route>
          {/* Route when page Not Found*/}
        </Routes>
      </BrowserRouter>
    </>
  );
};

export default App;
```

Output



Since / ,/about,/course url found hence rendering their corresponding components .
/abc,/def not found hence rendering pageNotFound (Page Not Found your are at wrong Page)

X. PURE COMPONENTS

What are pure components ?

Pure Components in React are the components which do not re-renders when the value of state and props has been updated with the same values.

If the value of the previous state or props and the new state or props is the same, the component is not re-rendered.

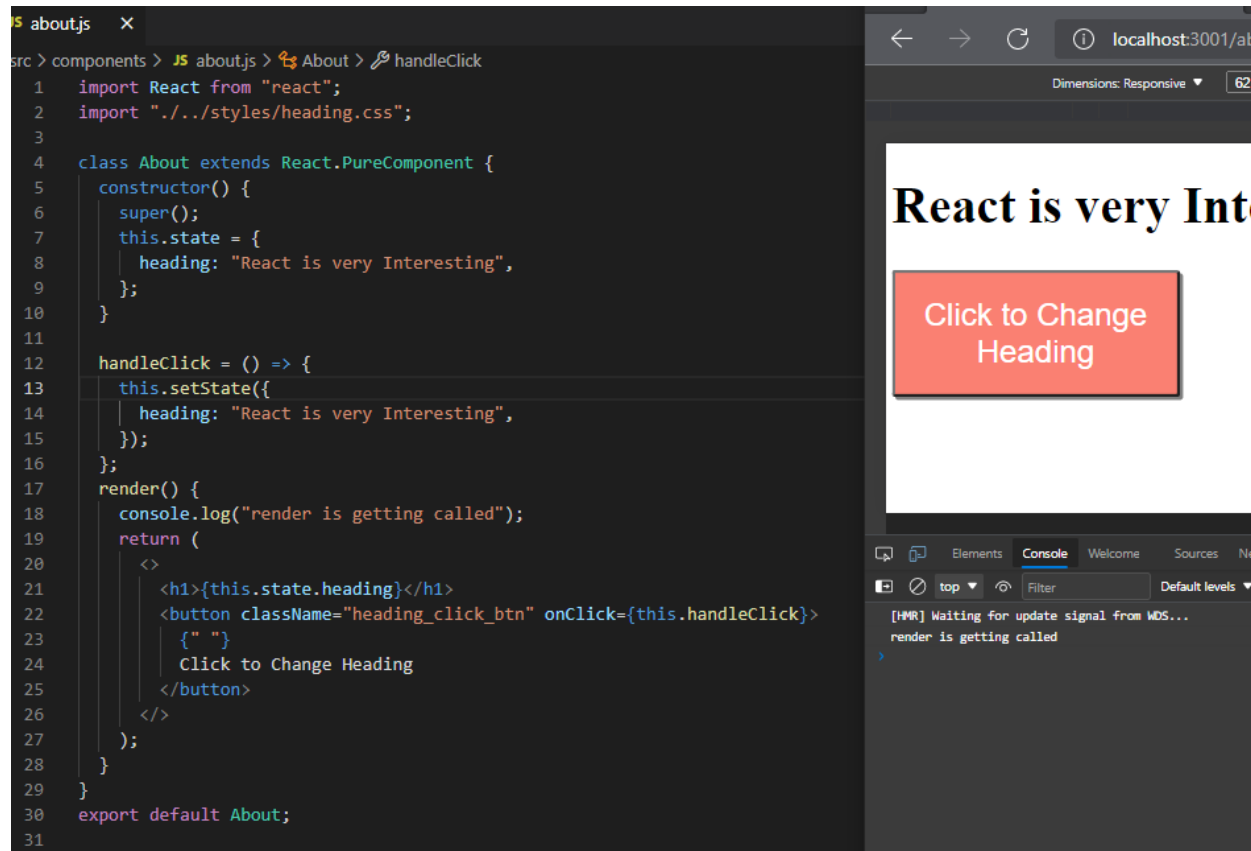
What are the benefits of pure components in React?

1. These are useful when to avoid re-rendering cycles of your component when its props and state are not changed .
2. Takes care of "shouldComponentUpdate" implicitly
3. Pure Components are more performant in certain cases.

How to use pure components in code ?

CASE: pure Component in case of state change

If the state of any variable is not changing then `render()` Will not be called, if the state is changing then `render()` will be called

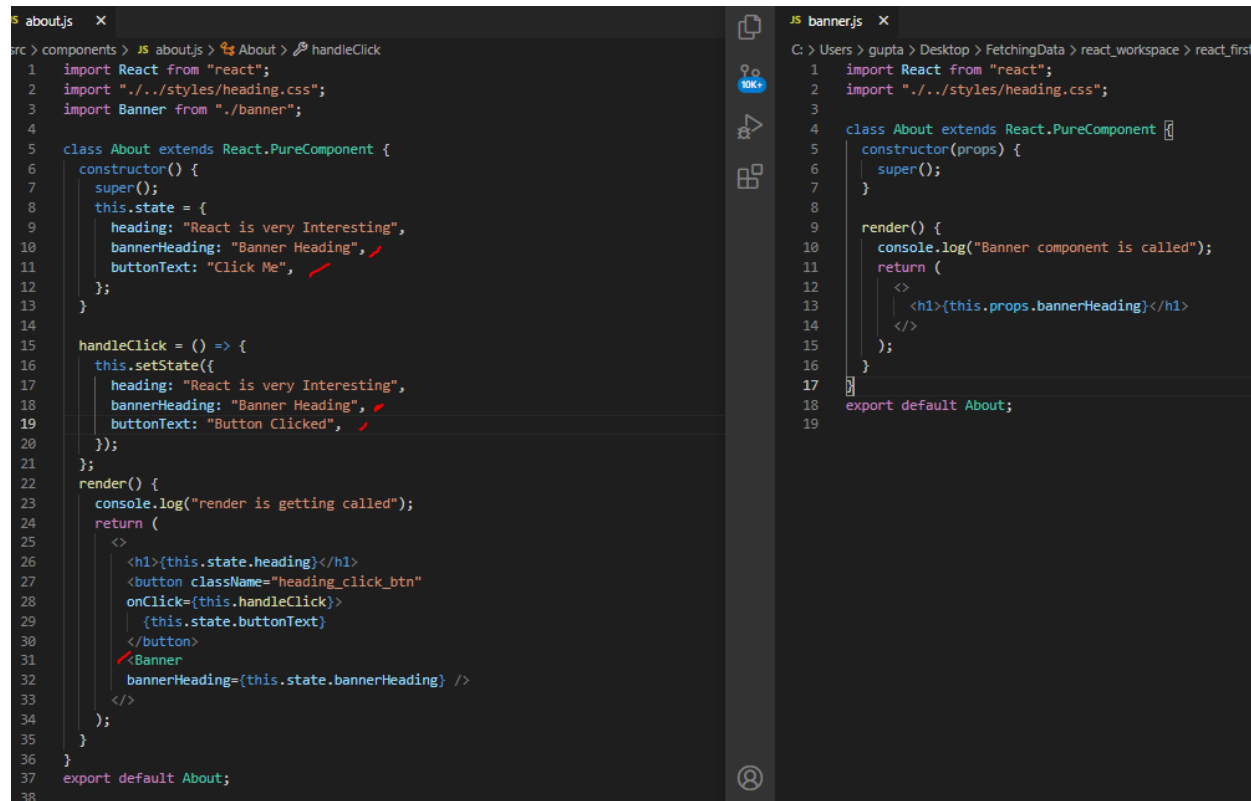


The screenshot shows a VS Code editor with a file named `about.js` and a web browser at `localhost:3001/about`. The browser displays a heading "React is very Interesting" and a button "Click to Change Heading". The console shows the message "render is getting called".

```
src > components > JS about.js > About > handleClick
1  import React from "react";
2  import "../styles/heading.css";
3
4  class About extends React.PureComponent {
5    constructor() {
6      super();
7      this.state = {
8        heading: "React is very Interesting",
9      };
10   }
11
12   handleClick = () => {
13     this.setState({
14       heading: "React is very Interesting",
15     });
16   };
17   render() {
18     console.log("render is getting called");
19     return (
20       <>
21         <h1>{this.state.heading}</h1>
22         <button className="heading_click_btn" onClick={this.handleClick}>
23           { " " }
24           Click to Change Heading
25         </button>
26       </>
27     );
28   }
29 }
30 export default About;
31
```

CASE: when the state of any variable is changing, but the passing value in props for child component is not changing

In these case child component will also render itself for every state change, to prevent it from rendering use pure components in the child component as well

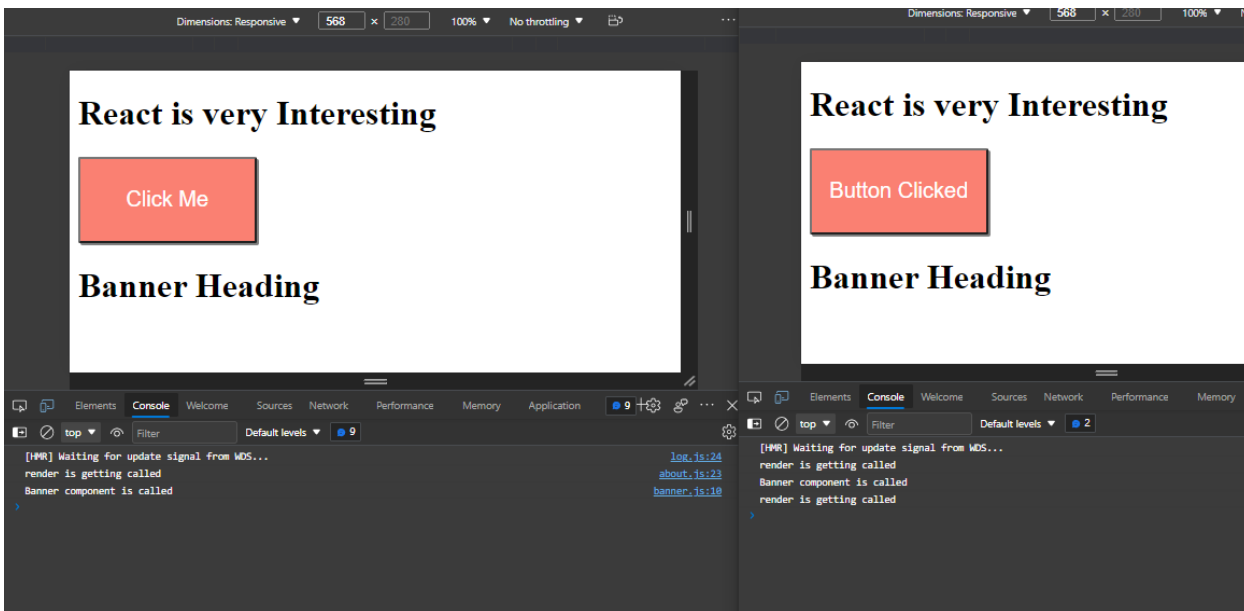


The screenshot shows two files in VS Code: `about.js` and `banner.js`. The `about.js` file shows a pure component that updates its state and renders a button and a `Banner` component. The `banner.js` file shows a pure component that receives props and renders a heading.

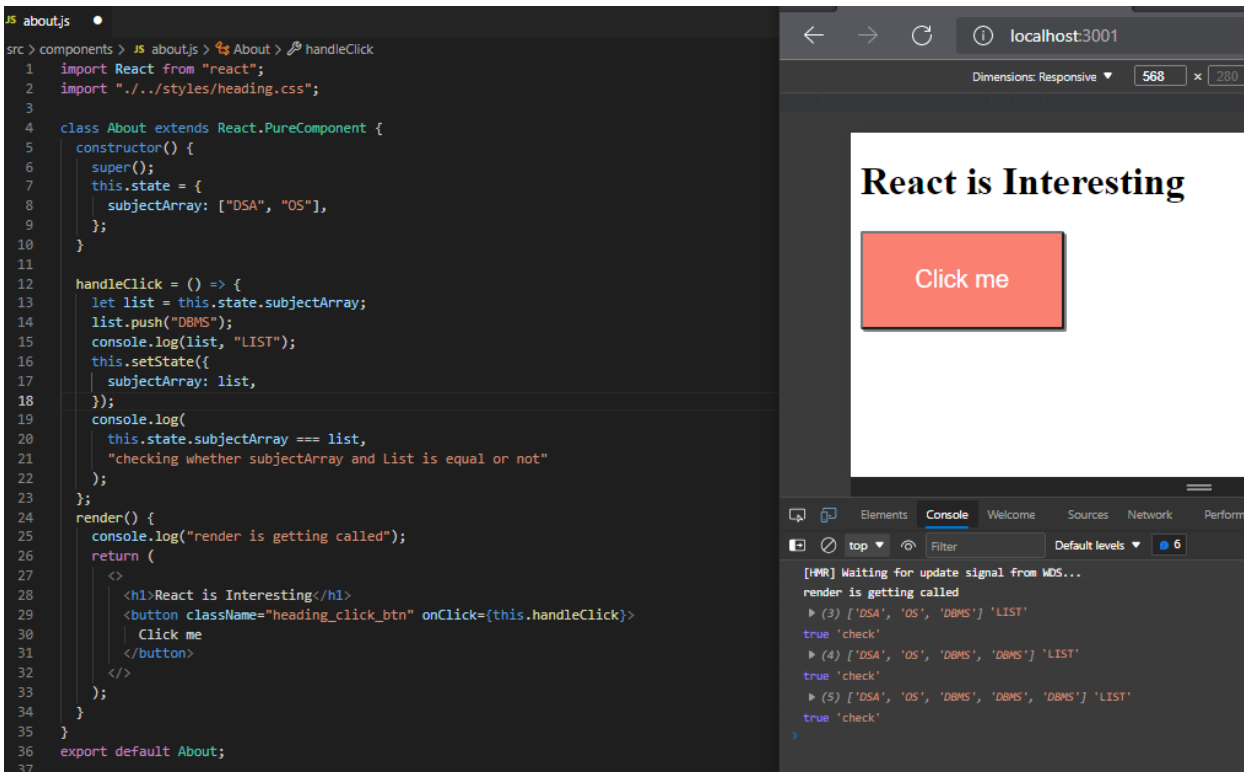
```
src > components > JS about.js > About > handleClick
1  import React from "react";
2  import "../styles/heading.css";
3  import Banner from "../banner";
4
5  class About extends React.PureComponent {
6    constructor() {
7      super();
8      this.state = {
9        heading: "React is very Interesting",
10        bannerHeading: "Banner Heading",
11        buttonText: "Click Me",
12      };
13   }
14
15   handleClick = () => {
16     this.setState({
17       heading: "React is very Interesting",
18       bannerHeading: "Banner Heading",
19       buttonText: "Button Clicked",
20     });
21   };
22   render() {
23     console.log("render is getting called");
24     return (
25       <>
26         <h1>{this.state.heading}</h1>
27         <button className="heading_click_btn"
28           onClick={this.handleClick}>
29           {this.state.buttonText}
30         </button>
31         <Banner
32           bannerHeading={this.state.bannerHeading} />
33       </>
34     );
35   }
36 }
37 export default About;
38
```

```
JS banner.js
1  import React from "react";
2  import "../styles/heading.css";
3
4  class About extends React.PureComponent {
5    constructor(props) {
6      super();
7    }
8
9    render() {
10     console.log("Banner component is called");
11     return (
12       <>
13         <h1>{this.props.bannerHeading}</h1>
14       </>
15     );
16   }
17 }
18 export default About;
19
```

Output



XI. PURE COMPONENTS ISSUES



since here we are adding new elements to the array, but still `render()` is not called

Reason: In case of the object the reference is being compared, we are getting inside the array and changing a particular value **but the reference to that particular array is not getting changed**, it still remains the same as a result of which this particular will give us true and for pure component, these are true as well and hence pure component feel that there is no change happen and it will not go ahead and will not render the components hence pure components are only to be used with simple variables that we have

While using Pure Components, thing to be noted,

In these components, the Value of State and Props are Shallow Compared So there is a possibility that if these State and Props Objects contain nested data structures then Pure Component's don't work properly.

XII. REACT HOOKS

What are Hooks in react?

Hooks are functions that let you “hook into ” React state and lifecycle features from class components .

Or

Hooks is a new feature(React 16.8) that lets you use state and other React features without writing a class.

After React 16.8 Functional Component are no more stateless components

Important things to remember while using hooks:

- Hooks are available for React version 16.8 or higher.
- Hooks doesn't violate any existing React concepts. Instead, Hooks provides a direct API to react concepts such as props, state, context, refs, and life-cycle.

Benefits of using Hooks and Why Hooks was introduced ?

- In react class component, we split our work into different life-cycle methods like componentDidMount, componentDidUpdate and componentWillUnmount, but in hooks, we can do everything in a single hook called useEffect.
- In the class component, we have to use this keyword and also we have to bind event listeners, which increases complexity. This is prevented in react functional components.

What is useState hook , how does it works ? What are arguments accepted by this hook and what is returned by hook ?

- useState hook is a function that is used to store state value in a functional component
- It accepts an argument as the initial value of the state. It returns an array with 2 elements
- First element is the current value of state. Second element is a function to update the state.

We import useState from React as

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

We will use it like

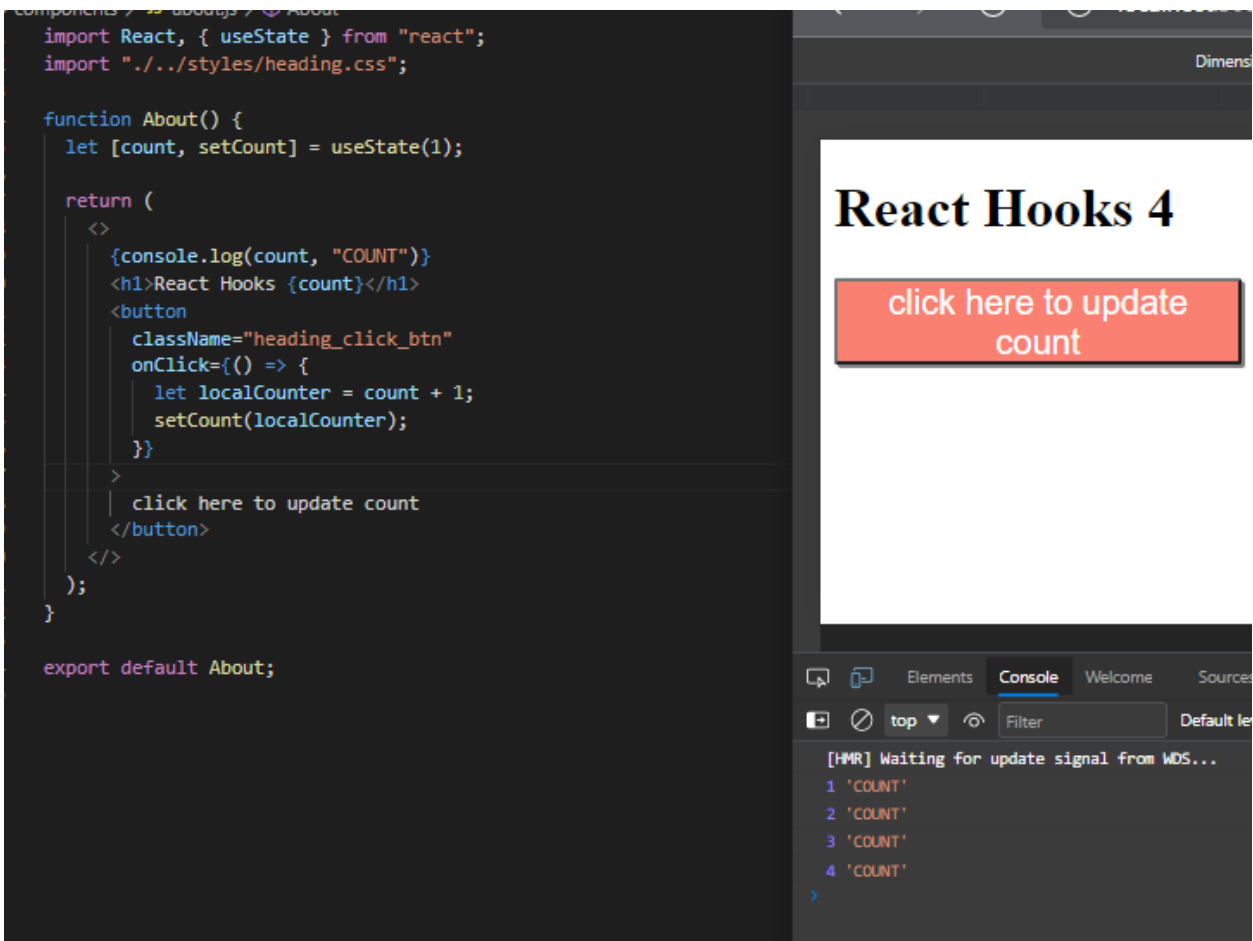
```
const [count, setCount] = useState(1);
```

What are the differences in using hooks and class components with respect to state management?

1. When using setState() in class components, the state variable is always an object (this.setState({})). Whereas, the state variable in hooks can be of any type like number, string, boolean, object, or array.(setCount([]),setDate(1))
2. When a state variable is an object, setState() in class components automatically merges the new value to the state object. But in the case of the functional component , we need to explicitly merge the updated object property using spread operator(...)

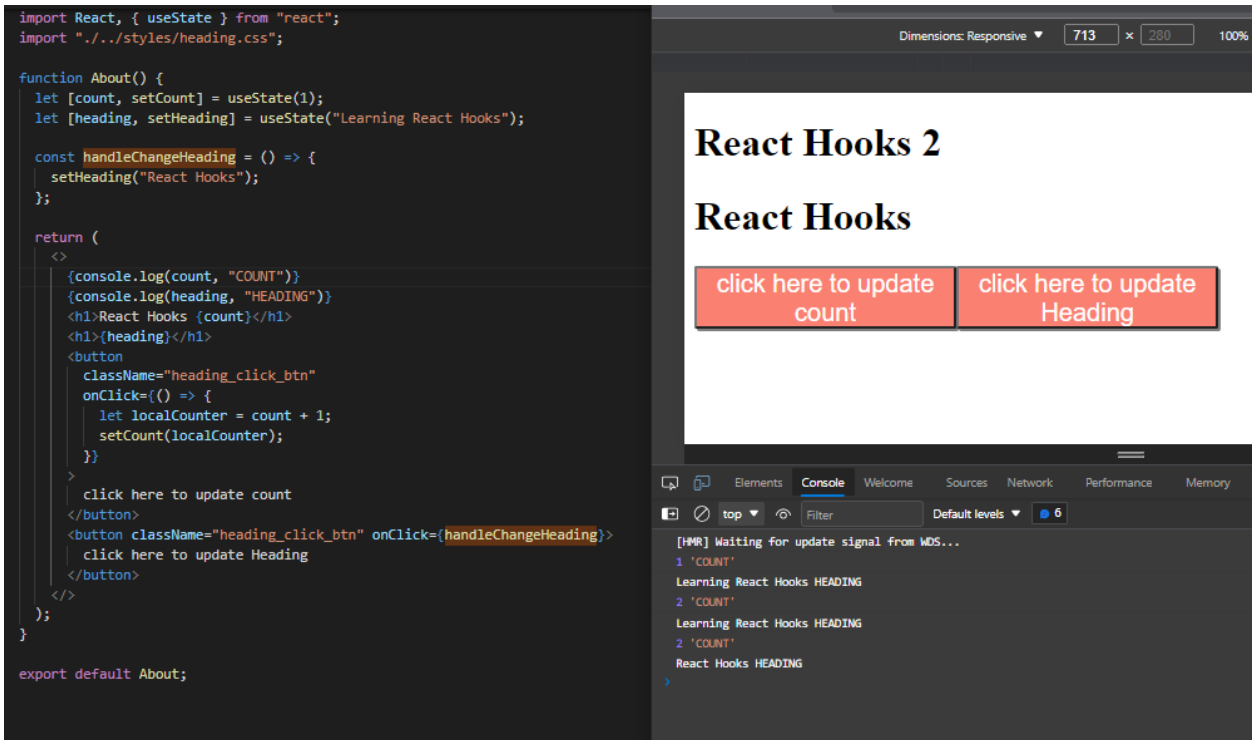
What is the purpose of useEffect hook?

The useEffect hook lets us perform different effects in functional components. It helps us to avoid redundant code in different lifecycle methods of a class component. It helps to group related code.



XIII. MORE ON USESTATE

Case When we have to update variable defined in useState in Functional Component



Case When we have to update object values

The image shows a web application built with React and Hooks. The code on the left defines a `useAddress` hook that manages address state, a `useCount` hook that manages a counter, and a `handleChangeAddress` function that updates the address state. The UI on the right displays the current state: "Street No 3", "State MP", and "City Bhopal". There are two buttons: "click here to update count" and "click here to update Address". The browser's developer console shows the state updates triggered by the buttons.

Behaviour of useState for different Life-cycle Methods

[illegible]

The `componentDidMount()` method allows us to execute the React code when the component is already placed in the DOM .

If we don't pass an empty array to `useEffect`, it will run on every change. Therefore, we must give as a second argument an empty array to mimic the `componentDidMount` behavior. It tells React that your effect doesn't depend on any values from props or state, so it never needs to be re-run, `useEffect` will run only once after the component is created.

The `componentDidUpdate()` method is invoked immediately after updating occurs.

```
//with componentDidUpdate
componentDidUpdate(prevProps){
  console.log(`component Did Update is called ${prevProps}`)
}
//with useEffect
useEffect(() => {
  console.log('useEffect called to behave like componentDidUpdate')
},[prevProps])

return /
```

When we pass a value(prevProps) to the array, it tells to useEffect to run only if the value change.

3. componentWillUnmount

The componentWillUnmount() method allows us to execute the React code when the component gets destroyed or unmounted

```
//withcomponentWillUnmount
componentWillUnmount(){
  console.log("componentWillUnmount is called")
}

//withUseEffect
useEffect(() => {
  return () => {
    console.log("unmounting");
  };
}, []);
```

XV. COMPONENTSWILLUNMOUNT USING USEEFFECTS

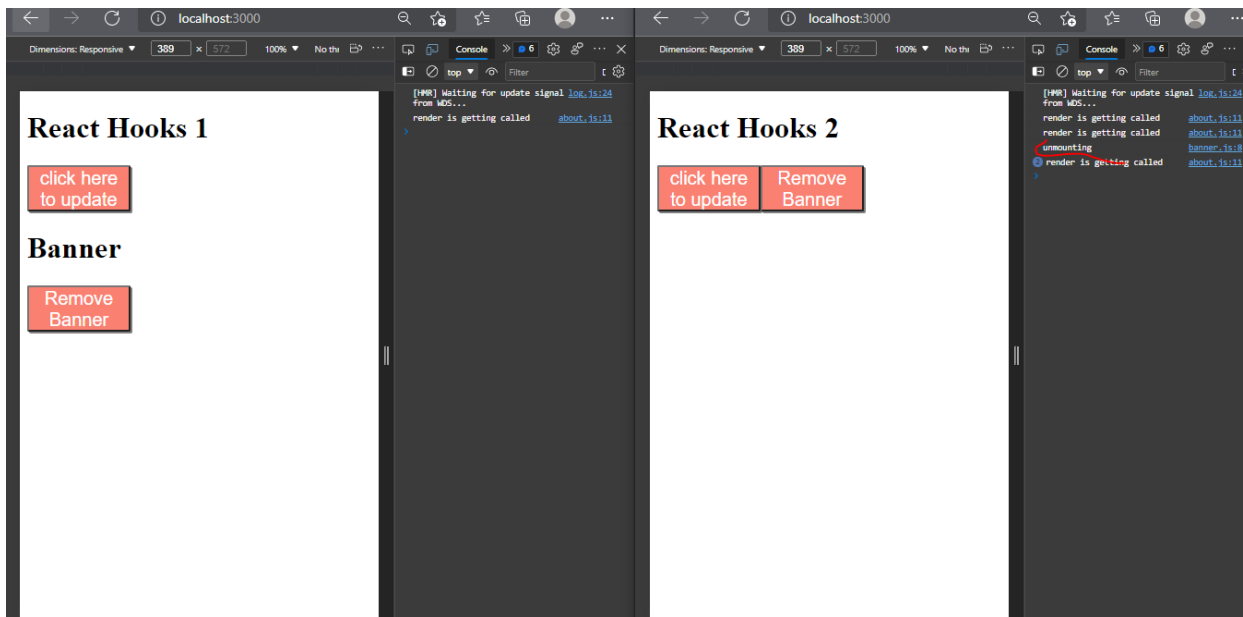
How can we call componentWillUnmount using useEffect ?

Code for calling componentWillUnmount using useEffect

```
about.js
src > components > JS about.js
1 import React, { useState } from "react";
2 import "../styles/heading.css";
3 import Banner from "../banner.js";
4 function About() {
5   let [count, setCount] = useState(1);
6   let [bannerVisible, setBannerVisible] = useState(true);
7
8   return (
9     <>
10      {console.log("render is getting called")}
11      <h1>React Hooks {count}</h1>
12
13      <button
14        className="heading_click_btn"
15        onClick={() => {
16          let localCounter = count + 1;
17          setCount(localCounter);
18        }}
19      >
20        click here to update count
21      </button>
22
23      {bannerVisible ? <Banner /> : null}
24      <button
25        className="heading_click_btn"
26        onClick={() => {
27          setBannerVisible(false);
28        }}
29      >
30        Remove Banner
31      </button>
32    </>
33  );
34 }
35
36 export default About;
37
```

```
banner.js
C:\Users\gupta\Desktop\FetchingData> react_workspace > react_first_project > src > c
1 import React, { useEffect } from "react";
2 import "../styles/heading.css";
3
4 function Banner() {
5   // useEffect will behave like componentWillUnmount
6   useEffect(() => {
7     return () => {
8       console.log("unmounting");
9     };
10   }, []);
11
12   return <h1>Banner</h1>;
13 }
14 export default Banner;
15
```

Output



Redux

I. INTRODUCTION TO REDUX

what is Redux ?

In React with the help of props, we pass non-static variables from child component to parent component. But as the count of components increases in the application in accordance with its large objective, we need to pass these states to other components located far away from each other in the component tree. At that moment, the way of transferring state as props turns out to be complex. Hence it brings a need for Redux in our react-application. Redux will give you a common store where data will store and if there is a particular change in the state that is stored in the state in that case all the components who are listening that particular value they will get updated as well.

What is the redux working procedure?

The redux working contains majorly 4 elements

Actions -> Dispatcher -> Reducers -> state

1. **Actions:** The actions part of the react-redux basically contains the different actions that are to be performed on the state present in the store. **The actions must contain action type**
2. **Dispatcher:** The task of the dispatcher is to dispatch the action from action to reducers
3. **Reducers :** The reducers in react-redux contain the operations that need to be performed on the state. These functions accept the initial state of the state being used and the action type. It updates the state and responds with the new state. **A reducer must contain the action type**
4. **Store:** It contains the state of the components which need to be passed to other components. **store makes it much easier by containing state as we no longer need to maintain a state inside a parent component in order to pass the same to its children components**

II. REDUX WORKING

Redux Installation

npm i redux

What is CreateStore in Redux ?

CreateStore :It Creates a Redux store that holds the complete state tree of your application. There should only be a single store in your app

```
CreateStore(reducer,[preloadedStore],enhancer)
```

```
const store = createStore(rootReducer);
```

CreateStore contains 3 parameters

Reducer(Function) : A reducing function that returns the next state, given the current state tree and an action to handle

preloadedStore(any) : The initial State(optional)

Enhancer(Function) : The Store enhancer,used to enhance store with third party capabilities like middleware(optional)

What is Subscribe() in redux ?

Subscribe(listener)

It will be called any time an action is dispatched, and some part of the state tree may potentially have changed. You may then call getState to read the current state tree inside the callback.

What is getState() in redux ?

getState()

Returns the current state tree of your application. It is equal to the last value returned by the store's reducer.

```
store.subscribe(() => {  
  console.log(store.getState());  
}); //Subscription to get updated value
```

How does redux work?

Redux follow following steps :

Step 01 : Create Store

```
const store = createStore(rootReducer);
```

Create Store ,but createStore needs reducer (reducer is just a function which does a task of taking an action and based on whatever the type of action, it goes ahead and update the state and return that updated state and whatsoever initialization of state)

Step 02 : Create reducer

```
const rootReducer = (state = initialState, action) => {  
  //creating reducer  
  if (action.type === "Add_MOVIE") {  
    return {  
      ...state, //updating state  
      movieList: action.list,  
    };  
  } else if (action.type === "CHANGE_USERNAME") {  
    return {  
      ...state,  
      name: action.name,  
    };  
  }  
  return state;  
};
```

Step 03 : since some part of the state tree may potentially have changed.we need to subscribe it , so we will call subscribe()
You may then call getState to read the current state tree inside the callback.

```
store.subscribe(() => {
  console.log(store.getState());
}); //Subscription to get updated value
```

complete code how redux work

```
const redux = require("redux");
const createStore = redux.createStore; //used for creating store

const initialState = {
  movielist: [],
  name: "James Potter",
};

const rootReducer = (state = initialState, action) => {
  //creating reducer
  if (action.type === "Add_MOVIE") {
    return { //updating state
      ...state,
      movielist: action.list,
    };
  } else if (action.type === "CHANGE_USERNAME") {
    return { //updating state
      ...state,
      name: action.name,
    };
  }
  return state;
};

const store = createStore(rootReducer);

console.log(store.getState()); //initially return current state

store.subscribe(() => {
  console.log(store.getState());
}); //Subscription to get updated value

//Dispatch Action
store.dispatch({
  type: "Add_MOVIE",
  list: {
    name: "DOLJ",
    year: "1990",
  },
}); //this is action dispatch by dispatch method(as soon it is dispatch it will go to the
// reducer)

store.dispatch({
  type: "CHANGE_USERNAME",
  name: "Harry Potter",
}); //this is another action dispatch by dispatch method(as soon it is dispatch it will go
//to the reducer)
```

III. REACT REDUX OVERVIEW

Installation

npm i react-redux

What is React Redux ?

React Redux the official React UI bindings layer for Redux. It lets your React components read data from a Redux store, and dispatch actions to the store to update the state.

What is the difference between Redux and React-Redux ?

Redux is an open-source JavaScript library for managing and centralizing application state. It is independent in itself Redux allows React components to read data from a Redux Store, and dispatch Actions to the Store to update data, that is react redux

What is Provider ?

component, which makes the Redux store available to the rest of your app or the rest of your component .

```
<Provider store={store}>
  {/* provider makes the Redux store available to the rest of your component */}
  <App />
</Provider>
```

What is connect() ?

The connect() function connects a React component to a Redux store. It provides its connected component with the pieces of the data it needs from the store, and the functions it can use to dispatch actions to the store.

```
const mapStateToProps = (state) => { //state is that we have in our redux store
  return {
    ctr: state.counter,
    val: state.value,
  };
};

const mapDispatchToProps = (dispatch) => {
  return {
    incrementCounter: () => dispatch({ type: "INCREMENT_COUNTER" }), // providing dispatch functions that can dispatch actions to the store.
  };
};

export default connect(mapStateToProps, mapDispatchToProps)(App);
```

connect accepts four different parameters, all optional

```
connect(mapStateToProps, mapDispatchToProps, null, {context: MyContext})(MyComponent)
```

1. mapStateToProps: Function
2. mapDispatchToProps: Function | Object
3. mergeProps: Function
4. options: Object

1.mapStateToProps : Any Time Store is updated mapStateToProps is called
A mapStateToProps function takes a maximum of two parameters.

(i) state: Object

(ii) ownProps: Object

If your mapStateToProps function is declared as taking one parameter, it will be

```
const mapStateToProps = (state) => { //state is that we have in our redux store
  return { //state is updating
    ctr: state.counter,
    val: state.value,
  };
};
```

2.mapDispatchToProps: Function that dispatch actions to the store. it can be called with a maximum of two parameters.

(i) dispatch: Function

(ii) ownProps: Object If your mapDispatchToProps function is declared as taking one parameter, it will be

```
const mapDispatchToProps = (dispatch) => {
  return {
    incrementCounter: () => dispatch({ type: "INCREMENT_COUNTER" }),
  }; // providing dispatch functions that can dispatch actions to the store.
};
```

3.mergeProps : defines how the final props for your own wrapped component are determined.

They are specified with 3 parameters :

(i) stateProps

(ii) dispatchProps

(iii) ownProps

The parameters are the result of mapStateToProps(), mapDispatchToProps(), and the wrapper component's props respectively .

```
const mergeProps = (ownProps, stateProps, dispatchProps) => {
  ...ownProps,
  ...stateProps,
  ...dispatchProps,
}
```

4.options: Object : This parameter is supported in react-redux >= v6.0 only

```

{
  context?: Object,
  pure?: boolean,
  areStatesEqual?: Function,
  areOwnPropsEqual?: Function,
  areStatePropsEqual?: Function,
  areMergedPropsEqual?: Function,
  forwardRef?: boolean,
}

```

React Redux -working code

Step: 01 Create a separate file let's say reducer.js and create reducer here

```

JS reducer.js x
src > components > store > reducer > JS reducer.js > ...

1  const initialState = {
2    |   counter:0,
3    |   value:3
4  }
5  //reducer which take two parameter state and action
6  const rootReducer = (state=initialState, action) => {
7    |   if (action.type == "INCREMENT_COUNTER") {
8    |     |   return {
9    |     |     ...state,
10    |     |     counter: state.counter + 1,
11    |     |   };
12    |   }
13
14   return state
15 };
16
17 export default rootReducer;
18

```

Step: 02 Create a store in index.html and use component, which makes the Redux store available for rest of your component:

```

JS index.js
src > JS index.js > ...

1  import React from "react";
2  import ReactDOM from "react-dom";
3  import App from "../components/app";
4  import { createStore } from "redux";
5  import rootReducer from "../components/store/reducer/reducer.js";
6  import { Provider } from "react-redux";
7
8  const store = createStore(rootReducer); //created a store
9
10 ReactDOM.render(
11   |   <Provider store={store}>
12   |   |   /* provider makes the Redux store available to the rest of your component */
13   |   |   <App />
14   |   </Provider>,
15   |   document.getElementById("root")
16 );
17

```

Step :03 in your app.js use connect that connects a React component to a Redux store.and pass mapStateToProps (Any Time Store is updated mapStateToProps is called) and mapDispatchToProps(Function that dispatch actions to the store.):

BABEL & WEBPACK

I. WEBPACK INTRODUCTION

What is Webpack?

Webpack is a static module bundler for modern javascript applications. When Webpack processes your application ,it internally builds a dependency graph that maps every Module your project needs and generate one or more bundles .

Some Core Concepts of Webpack are

- 1. Entry:** The point where webpack starts building.
- 2. Output:** It decides after the bundling happened where the file resides.
- 3. Loaders:** Loaders are transformations that are applied on the Source code of the module, that describes webpack how to process non-javascript modules.
- 4. Plugins:** used to add certain functionality.
- 5. Mode:** By setting the mode parameter to either development, production, or none, we can enable webpack's built-in optimizations , correspond to each environment. The default value of Mode is a production .
- 6 Browser Capability:** Webpack supports all browsers that are ES5-compliant (IE8 and below are not supported) If we want to support older browsers, we will need to load a polyfill(A polyfill is a piece of code (usually JavaScript on the Web) used to provide modern functionality on older browsers that do not natively support it.) before using.

What are the pros and cons of using Webpack?

pros

1. It bundles your multiple modules and packs them into a single .js file, It gives Javascript a modular system (JavaScript doesn't have a module system built-in)
2. The development can be speed up when using webpack
3. You can set up single page application better while using Webpack
4. Total Control overbuilding System: we need to transpile our ES6+ code in previous versions to make our Javascript code compatible with older browsers we can choose various build systems we will need for this

cons

1. It requires polyfills for loading modules, Webpack's bundle size can be much larger than other bundlers that use ESM and modern JavaScript syntax
2. We need plugins for doing simple things, like loading CSS, and the config files are extremely complicated

What is a bundle in webpack ?

A bundle is some related code packaged into a single file.If we don't want all of our code to be put into a single bundle we can split it into multiple bundles which are called chunks in webpack terminology.

What is the format of the webpack config file ?

webpack's config file is a javascript file in commonjs module pattern. (CommonJS is a module formatting system. It is a standard for structuring and organizing JavaScript code)

```
const path = require('path');

module.exports = {
  entry: './src/index.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: 'bundle.js',
  },
};
```

II. WEBPACK ENTRY AND OUTPUT

What are the entry points and output in the webpack ?

The entry object is where webpack looks to start building the bundle, at this point the application starts executing.

Output: The output property tells webpack where to emit the bundles it creates and how to name these files .

What is a dependency graph and how does webpack build it ?

At any time one file depends on another, webpack treats this as a dependency. Starting from entry points, webpack recursively builds a dependency graph that includes every module your application needs, then bundles all of those modules into a small number of bundles using the import statement

Is it possible to have multiple entry points in a single webpack configuration file?

yes, It is helpful when we would like to inject multiple dependent files together and graph their dependencies into one "chunk".

```
module.exports = {
  entry: ['./src/file_1.js', './src/file_2.js'],
  output: {
    filename: 'bundle.js',
  },
};
```

How can we generate a webpack config file automatically ?

using
npm i webpack-cli init

III.BABEL LOADER

What is a loader in webpack ?

Loaders are transformations that are applied on the source code of a module. webpack supports modules written in a variety of languages

Loaders describe to webpack how to process non-javascript modules and include these dependencies into your bundles.

Loaders have two properties in webpack Configuration

- 1 **The test property:** used to identify which file or files should be transformed by the respective loader
- 2 **The use property:** indicate which loader should be used to do transforming .

Example of some Loaders

url-loader, html-loader, file-loader, style-loader, css-loader, script-loader, babel-loader, coffescript,

What is Babel Loader ?

Babel loader is used to convert code written in modern Javascript into plain old JavaScript code supported by older browsers

Installation

npm install -D babel-loader @babel/core @babel/preset-env webpack

What are presets ?

Presets are collection of plugins that helps babel to do the conversion From ES6 to ES5 conversion .

```

module: {
  rules: [
    {
      test: /\.m?js$/,
      exclude: /(node_modules|bower_components)/,
      use: {
        loader: 'babel-loader',
        options: {
          presets: ['@babel/preset-env']
        }
      }
    }
  ]
}

```

****rules["test"]:**** which particular file do you want to be executed

****rules["Use"] ["loader"]:**** which particular loader do you want to be used

****rules["Use"]["options"]["presets"]:**** which presets do you want to use

****rules["exclude"]:**** whatever file or modules you want to remove to get transpile

List Out Some of presets ?

1. @babel/preset-env - for compiling ES2015+ syntax
2. @babel/preset-typescript - for TypeScript
3. @babel/preset-react - for React
4. @babel/preset-flow - for Flow

What are plugins in webpack?

A webpack plugin is a JavaScript object that has an apply property. This apply property is called by the webpack compiler, giving access to the entire compilation lifecycle. Webpack comes with multiple built-in plugins available under webpack.

What is ProvidePlugin ?

This Plugin automatically load modules instead importing them or require them everywhere.

```

plugins: [
  new webpack.ProvidePlugin({
    React: "react",
  }),
]

```

Once we have React declared inside ProvidePlugin, now we need not to import it in all .js files .

How to move some data (e.g css code) from a bundle to a separate file in webpack ?
using ExtractTextWebpackPlugin.

V. CSS LOADERS

What is CSS Loader ?

css-loader is the npm module that would help webpack to collect CSS from all the css files referenced in your application and put it into a single file. Css loader does the task of loading css in webpack

What is a Style Loader ?

Style loader automatically injects styles into the DOM using style tag .

or

Style loader does the task of injecting CSS in our html file through style element

Installation css loader and style-loader

npm i css-loader style-loader

```
module.exports = {  
  module: {  
    rules: [  
      {  
        test: /\.css$/i,  
        use: ["style-loader", "css-loader"],  
      },  
    ],  
  },  
};
```

What is a mini-css extract plugin ?

This plugin extracts CSS into separate files. It creates a CSS file per JS file which contains CSS .

Installation :

npm i mini-css-extract-plugin


```

const MiniCssExtractPlugin = require("mini-css-extract-plugin");
const path = require("path");
const webpack = require("webpack");

module.exports = {
  entry: "./index.js",
  output: {
    path: path.resolve(__dirname, "build"),
    filename: "bundle.js",
    publicPath: "build/",
  },
  module: {
    rules: [
      {
        test: /\.js$/,
        use: "babel-loader",
        exclude: /node_modules/,
      },
      {
        test: /\.css$/,
        use: [MiniCssExtractPlugin.loader, "css-loader"],
      },
    ],
  },
  plugins: [
    new webpack.ProvidePlugin({
      React: "react",
    }),
    new MiniCssExtractPlugin({
      filename: "index.css",
    }),
  ],
};

```

Complete webpack.config.js file having entry, output, loaders, plugins

VI. WEBPACK DEV SERVER

What is a webpack dev server ?

Webpack With a dev server provides live reloading This should be used only for development server webpack-dev-server simplifies the development process due to integrated fast in-memory access to the webpack assets.

Installation

npm i webpack-dev-server

What is the hot modules replacement feature?

Hot-Modules-Replacement(HMR) is a webpack feature that allows updating modules in applications without page reload.

What is the advantage of using webpack dev-server?

webpack-dev-server simplifies the development process due to integrated fast in-memory access to the webpack assets and hot-modules-replacement features.

```

1 import React, { Component } from "react";
2 import { connect } from "react-redux";
3
4 class App extends Component {
5   render() {
6     console.log(this.props, "PROPS");
7     return (
8       <>
9         <h1>React Redux</h1>
10        <h1>{this.props.ctr}</h1>
11        <button onClick={this.props.incrementCounter}>Increment</button>
12      </>
13    );
14  }
15 }
16
17 const mapStateToProps = (state) => {
18   //state is that we have in our redux store
19   return {
20     //state is updating
21     ctr: state.counter,
22     val: state.value,
23   };
24 };
25
26 const mapDispatchToProps = (dispatch) => {
27   return {
28     incrementCounter: () => dispatch({ type: "INCREMENT_COUNTER" }),
29   }; // providing dispatch functions that can dispatch actions to the store
30 };
31
32 export default connect(mapStateToProps, mapDispatchToProps)(App);
33 //connect that connect react component to store
34

```

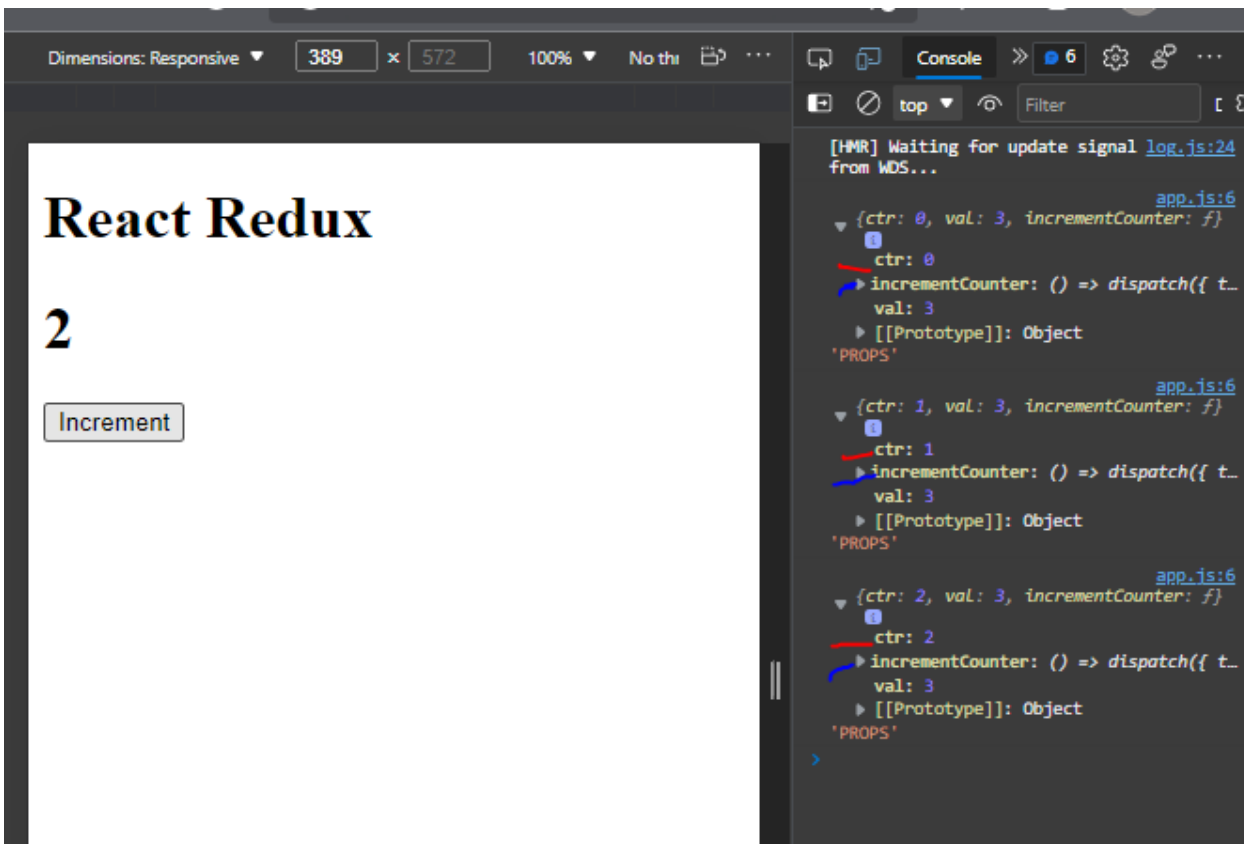
React-redux Complete working code

```

1 const initialState = {
2   counter:0,
3   value:3
4 }
5 //reducer which take two parameter state and action
6 const rootReducer = (state=initialState, action) => {
7   if (action.type == "INCREMENT_COUNTER") {
8     return {
9       ...state,
10      counter: state.counter + 1,
11    };
12  }
13  return state
14 };
15 export default rootReducer;
16
17
18
19 import React from "react";
20 import ReactDOM from "react-dom";
21 import App from "../components/app";
22 import { createStore } from "redux";
23 import { Provider } from "react-redux";
24
25 const store = createStore(rootReducer); //create
26
27 ReactDOM.render(
28   <Provider store={store}>
29     <App />
30   </Provider>,
31   document.getElementById("root")
32 );
33
34
35 import React, { Component } from "react";
36 import { connect } from "react-redux";
37
38 class App extends Component {
39   render() {
40     console.log(this.props, "PROPS");
41     return (
42       <>
43         <h1>React Redux</h1>
44         <h1>{this.props.ctr}</h1>
45         <button onClick={this.props.incrementCounter}>
46           Increment
47         </button>
48       </>
49     );
50   }
51 }
52
53 const mapStateToProps = (state) => {
54   //state is that we have in our redux store
55   return {
56     //state is updating
57     ctr: state.counter,
58     val: state.value,
59   };
60 };
61
62 const mapDispatchToProps = (dispatch) => {
63   return {
64     incrementCounter: () =>
65       dispatch({ type: "INCREMENT_COUNTER" }),
66   }; // providing dispatch functions that can dispatch act
67 };
68
69 export default connect(mapStateToProps, mapDispatchToProps)(App);
70 //connect that connect react component to store
71

```

Output



IV. ACTION CREATORS

we can make separate files for all our actions since in props we have dispatch function with the help of dispatch function we can dispatch those actions we just have to use that dispatch function ,hence action will be dispatched .

Step : 01 create a new file for action to create let's say action.js

```
JS action.js
src > components > store > actions > JS action.js > ...

1  export const counterAction = () => {
2    return {
3      type: "INCREMENT_COUNTER",
4    };
5  };
6
7  export const counterValue = () => {
8    return {
9      type: "INCREMENT_VALUE",
10   };
11 };
12 |
```

Step : 02 create a file for reducer lets say reducer.js

JS reducer.js X

src > components > store > reducer > JS reducer.js > [0] rootReducer

```
1  const initialState = {
2    counter: 0,
3    value: 3,
4  };
5  //reducer which take two parameter state and action
6  const rootReducer = (state = initialState, action) => {
7    if (action.type === "INCREMENT_COUNTER") {
8      return {
9        ...state,
10       counter: state.counter + 1,
11     };
12   }
13
14   if (action.type === "INCREMENT_VALUE") {
15     return {
16       ...state,
17       counter: state.counter + 5,
18     };
19   }
20
21   return state;
22 };
23
24 export default rootReducer;
25
```

Step : 03 since we already have dispatch function in our props we just have to use that dispatch function ,and pass action in that function hence action will be dispatched

```
app.js:7
▼ {ctr: 0, val: 3, dispatch: f}
  ctr: 0
  dispatch: f dispatch(action)
    length: 1
    name: "dispatch"
    ► prototype: {constructor: f}
      arguments: (...)
      caller: (...)
      [[FunctionLocation]]: redux.js:273
      ► [[Prototype]]: f ()
      ► [[Scopes]]: Scopes[3]
      val: 3
      ► [[Prototype]]: Object
  'PROPS'
```

```
const dispatch = useDispatch(); //useDispatch
//return store dispatch method

return (
  <>
    <h1>{counter}</h1>
    <button
      onClick={() => {
        dispatch(counterAction());
      }}
    >
      Increment by One
    </button>

    <button
      onClick={() => {
        dispatch(counterValue());
      }}
    >
      Increment by Five
    </button>
  </>
);
```

Complete code of react-redux

```

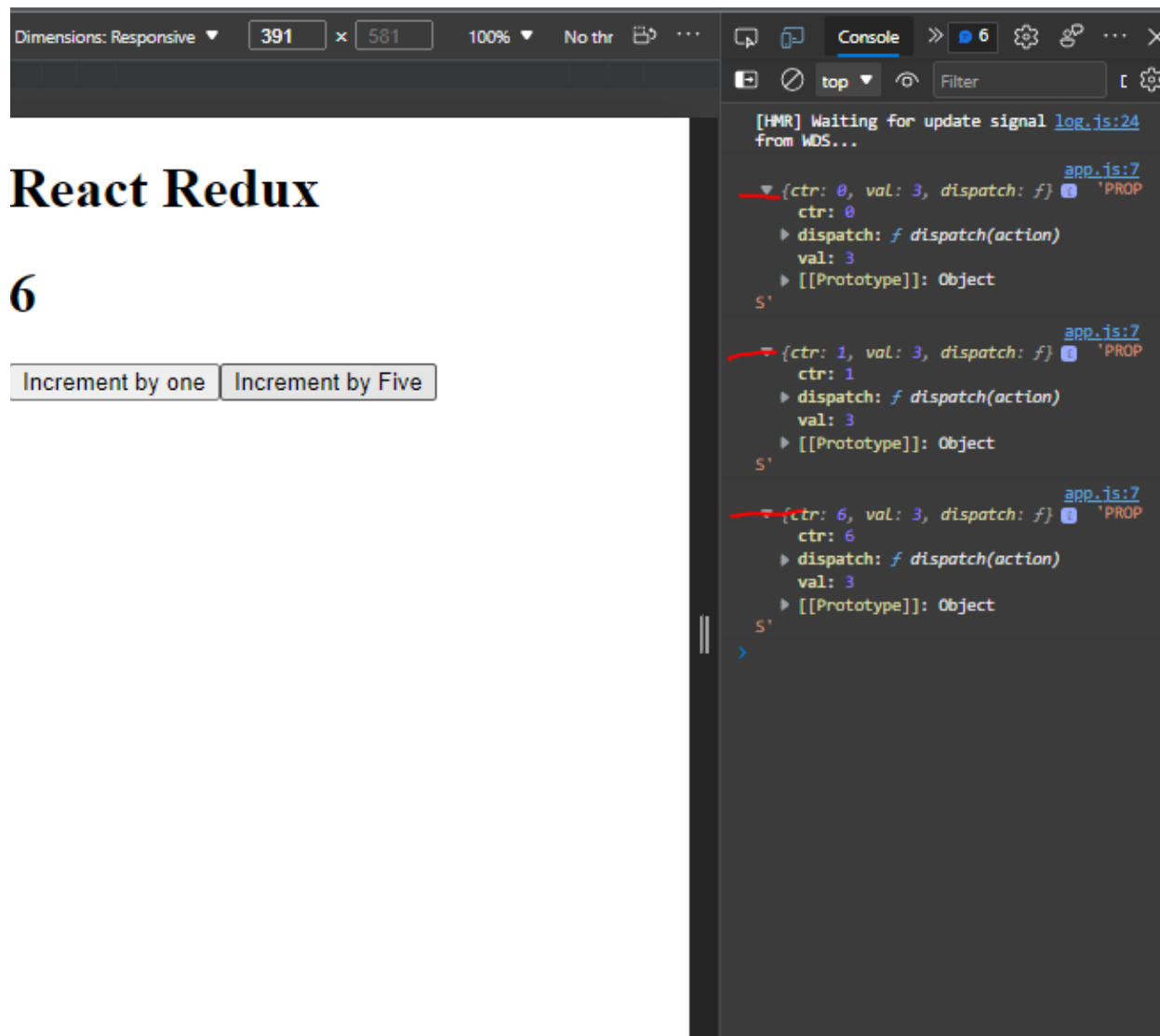
# actions.js
src > components > store > actions > # actions.js > ...
1 export const counterAction = () => {
2   return {
3     type: "INCREMENT_COUNTER",
4   };
5 };
6
7 export const counterValue = () => {
8   return {
9     type: "INCREMENT_VALUE",
10  };
11 };
12
13
14
15 //ACTIONS CREATED

# app.js
C:\Users\gupta\Desktop> FetchingData > react_workspace > react_first_project > src > components > # app.js > ...
1 import { useSelector, useDispatch } from "react-redux";
2 import { counterAction, counterValue } from "../store/actions/action";
3
4 const App = () => {
5   const counter = useSelector((state) => {
6     return state.counter;
7   }); //useSelector to read
8   //Value from state
9
10  const dispatch = useDispatch(); //useDispatch
11  //return store dispatch method
12
13  return (
14    <>
15      <h1>{counter}</h1>
16      <button
17        onClick={() => {
18          dispatch(counterAction());
19        }}
20      >
21        Increment by One
22      </button>
23
24      <button
25        onClick={() => {
26          dispatch(counterValue());
27        }}
28      >
29        Increment by Five
30      </button>
31    </>
32  );
33 };
34
35 export default App;
36
37 //DISPATCHING ACTIONS
38

# reducer.js
Desktop > FetchingData > react_workspace > react_first_project > src > components > store > # reducer.js > ...
1 const initialState = {
2   counter: 0,
3   value: 3,
4 };
5
6 const rootReducer = (state = initialState, action) => {
7   if (action.type === "INCREMENT_COUNTER") {
8     return {
9       ...state,
10      counter: state.counter + 1,
11    };
12  }
13
14  if (action.type === "INCREMENT_VALUE") {
15    return {
16      ...state,
17      counter: state.counter + 5,
18    };
19  }
20
21  return state;
22 };
23
24 export default rootReducer;
25
26
27 // REDUCER PERFORMING OPERATION
28 //ON STATE ACCORDING TO ACTION
29

```

Output



V. FUNCTION COMPONENT AND REDUX

React Redux provides a pair of React hooks that allow your React components to interact with the Redux store.

What are useSelector and useDispatch ?

useSelector: useSelector reads a value from the store state.

```
const counter = useSelector((state) => {  
  return state.counter;  
});
```

useDispatch: useDispatch returns the store's dispatch method to let you dispatch actions.

```
const dispatch = useDispatch();
```

React Redux - working code through Hooks

Step : 01 create a new file for action to create lets say action.js

```

JS action.js
src > components > store > actions > JS action.js > ...

1  export const counterAction = () => {
2    return {
3      type: "INCREMENT_COUNTER",
4    };
5  };
6
7  export const counterValue = () => {
8    return {
9      type: "INCREMENT_VALUE",
10   };
11 };
12

```

Step : 02 create a file for reducer lets say reducer.js

```

JS reducer.js X
src > components > store > reducer > JS reducer.js > [0] rootReducer

1  const initialState = {
2    counter: 0,
3    value: 3,
4  };
5  //reducer which take two parameter state and action
6  const rootReducer = (state = initialState, action) => {
7    if (action.type === "INCREMENT_COUNTER") {
8      return {
9        ...state,
10       counter: state.counter + 1,
11     };
12   }
13
14   if (action.type === "INCREMENT_VALUE") {
15     return {
16       ...state,
17       counter: state.counter + 5,
18     };
19   }
20
21   return state;
22 };
23
24 export default rootReducer;
25

```

Step : 03 since in useDispatch() we already have dispatch function we just have to use that dispatch function ,and pass action in that function hence action will be dispatched.

```

from WDS...
f dispatch(action) { app.js:11
  if (!isPlainObject(action)) {
    throw new Error( false ? undefined
: "Actions must be plain objects.
Instead, the actual type was: '" +
kindOf(action) + "'. You may need to_ 'us
eDispatch'
>

```

Interacting with Server

I. FRONTEND BACKEND INTERACTION OVERVIEW

Frontend:

Frontend or client Side rendering is a part of a website where users can interact with the website Some Of the Frontend Languages are

HTML: Developed by Tim Berners-Lee at CERN around year 1990

CSS: Developed by World Wide Web Consortium (W3C) and its initial release around 1996

JavaScript: Developed by Brendan Eich at Company NetScape in 1995

JQuery: John Resig working on his Own Project in 2006

SASS: Designed by Hampton Catlin and developed by Natalie Weizenbaum,Chris Eppstein in the year 2006

Bootstrap: Developed by Mark Otto and Jacob Thornton in Twitter and First released in 2012

React js: developed by Jordan Walke with thousands of Open-Source Contributors in 2011 at Facebook and Come public in 2013

Backend

The Code that runs on Server Side that receives requests from Clients and Contains the logic to send appropriate data back to the client Some of the Backend Languages are

C++: Developed by Bjarne Stroustrup in Bell Laboratories in the year 1979

JavaScript: It can be used as Both Frontend and Backend language

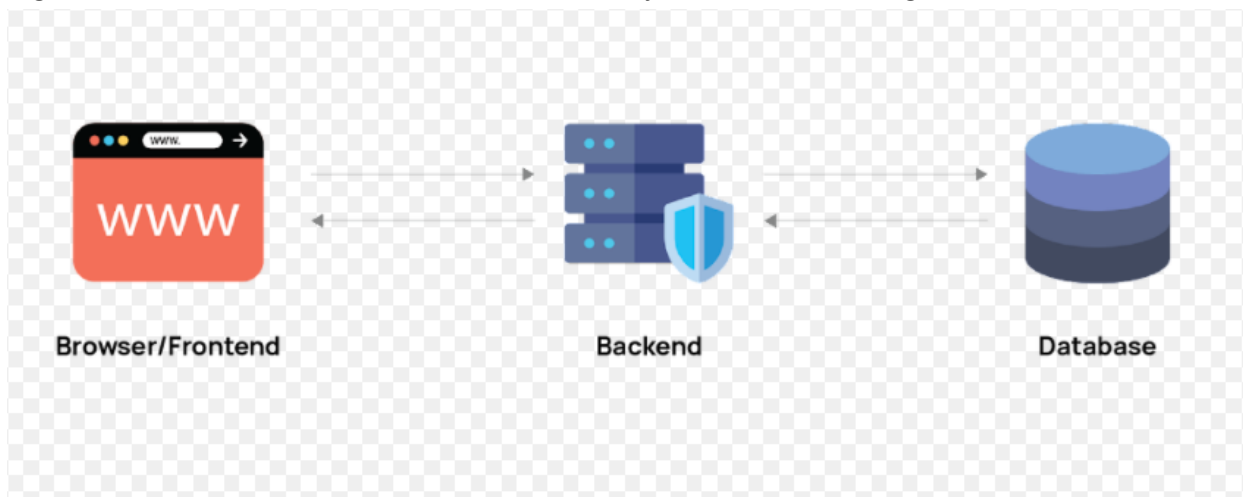
Java: Developed by James Goslings at Sun Microsystem(acquired by Oracle) in 1995

Python: Developed by Guido van Rossum and first released in February 1991

Nodejs: Neither Programming Language nor Framework, It is an Open-source Cross-Platform run-time Environment For Executing Javascript Code, We used it as a backend service to build Api's Developed by Ryan Dahi in the year 2009 .

Database:

Organized Collection of Data So that data can be easily accessed and managed



How Frontend Backend Interacts ?

Suppose we want a result Corresponding to any user-id

1. Firstly we will take input of that user-id through the frontend
2. The frontend will send a request to the backend by passing that userId in the request
3. Now backend have some API, through that Apis backend will interact with Database and take data from database corresponding to requested user ID

- once the backend get data from the database it will send a response to the frontend (Database will provide the data corresponding to the requested User Id)

To make Call from Fontend to backend we Will use Axios in React

What is API ?

Application Programming Interface

It helps to communicate between two applications and provides necessary data API is the messenger that delivers your request to the provider whom you are requesting it from and then deliver the response back to you .

II. AXIOS OVERVIEW

What is HTTP ?

HTTP (Hypertext Transfer Protocol) is a protocol that is used to transfer hypertext from client end to server end .

What are HTTP request Methods ?

Get,Post,Put,Delete

Get: This method retrieves information from a given server using a given URL

Post: Post method sends the data to server Example: need to update the phone number of user we will use post method

Put: It is used to replace all current representations of target resources with uploaded content

Delete: It is used to remove all the current representations of the target resource which is given by URL

What is a Status code ?

The Server issues some codes in response to a request of the client Basically, these are 3 digit codes, According to the status code we will identify the response from the backend

Status Code	Message
200	Ok, Request succeed
204	There is no content to send these request
400 (Bad Request)	The server could not understand the request due to invalid Syntax
403(Forbidden)	The Client does not have access rights Content unauthorized
500 (Internal Server Error)	The server has encountered a situation it does not know how to handle

What is Axios ?

Axios is a promise-based HTTP-Client in javascript or it is a library that serves to create HTTP requests from frontend to the backend. It Performs CRUD(Create Read Update Delete) Operations

What is a Promise ?

It is an object that returns a value that will either be in 3 states

Pending: initial state, neither fulfilled nor rejected

Fulfilled: meaning that operation was completed successfully resolve()

Rejected: meaning that the operation is failed reject()

Case when a promise is resolved

```
function dataToRender(data) {
  return new Promise((resolve, reject) => {
    if (data === "userdata") {
      resolve("Promise Resolved");
    } else {
      reject("Promise Rejected");
    }
  });
}

dataToRender("userdata")
  .then((result) => {
    console.log(result);
  })
  .catch((error) => {
    console.log(error);
  });
```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

PS C:\Users\gupta\Desktop\js> node index.js
Successful
PS C:\Users\gupta\Desktop\js>

Case when a promise is rejected

```
function dataToRender(data) {
  return new Promise((resolve, reject) => {
    if (data === "userdata") {
      resolve("Promise Resolved");
    } else {
      reject("Promise Rejected");
    }
  });
}

dataToRender("user")
  .then((result) => {
    console.log(result);
  })
  .catch((error) => {
    console.log(error);
  });
```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

PS C:\Users\gupta\Desktop\js> node index.js
Promise Rejected
PS C:\Users\gupta\Desktop\js>

What are Synchronous and Asynchronous Functions?

Synchronous Functions: In Synchronous Functions tasks are performed one at a time and you need to wait for a task to finish to move to the next one.

Asynchronous Functions: In Asynchronous Functions, the second task can begin executing in parallel, without waiting for an earlier task to finish.

What is await?

The await operator is used to wait for a Promise. It can only be used inside an async function within regular JavaScript code .

How to use await

Case when await is not using

```
function handleSubmit(data) {
  return new Promise((resolve, reject) => {
    if (data === "userData") {
      resolve("Promise Resolved");
    } else {
      reject("Promise Rejected");
    }
  });
}

// Case when await is not using
const handleClick = () => {
  let dataToSend = "userData";
  handleSubmit({ dataToSend })
    .then((result) => {
      console.log(result);
    })
    .catch((err) => {
      console.log(err);
    });
  console.log("Fetching Data");
};

handleClick(); // Function Call
```

> PROBLEMS

✓ TERMINAL

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\gupta\Desktop\FetchingData\data> node text.js
Fetching Data
Promise Rejected
PS C:\Users\gupta\Desktop\FetchingData\data> █
```

In these case, Function does not wait for the promise to resolve or reject, and start executing the next line of code within Function .

Case when await is using

```
function handleSubmit(data) {
  return new Promise((resolve, reject) => {
    if (data === "userData") {
      resolve("Promise Resolved");
    } else {
      reject("Promise Rejected");
    }
  });
}

💡 Case when await using ,
//Its parent Function must be async
const handleClick = async() => {
  let dataToSend = "userData";
  await handleSubmit({ dataToSend })
    .then((result) => {
      console.log(result);
    })
    .catch((err) => {
      console.log(err);
    });
  console.log("Fetching Data");
};

handleClick(); // Function Call
```

> PROBLEMS

✓ TERMINAL

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\gupta\Desktop\FetchingData\data> node text.js
Promise Rejected
Fetching Data
PS C:\Users\gupta\Desktop\FetchingData\data> █
```

The await keyword will cause the function to "wait" until its promise is resolve or reject before executing the next line Hence promise is returning first and then function is executing next line of code

Note : When using await its parent function must be async

What are Callback Functions ?

Callbacks are the functions that make sure that function is not going to run before a task is completed but will run right after the task has been completed.

How to declare Callback functions ?

Method 01

```
function totalPayment(x) {
  console.log(x);
}

function sumNumbers(a, b) {
  let c = a + b;
  totalPayment(c);
}

sumNumbers(1, 2);
// here totalPayment is callback function
// which will be called after sumNumbers
// functions will be called
```

> PROBLEMS
✓ TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

PS C:\Users\gupta\Desktop\js> node index.js
3
PS C:\Users\gupta\Desktop\js>

Method 02

```
componentDidMount() {
  axios
    .get("http://localhost:8000/")
    .then((response) => {
      //callback function which will
      // be called when promise is
      // returned (axios is promise based http client library)
      this.setState({
        postList: response.data,
      });
    })
    .catch((err) => {
      console.log(err);
    });
}
```

What are the advantages of Axios ?

- supports older browsers
- has a way to set a response timeout
- has a way to abort a request (by canceling a request using a cancel token API)
- performs automatic JSON data transformation.(by calling json() method on the response)

How to install and import Axios in .js file ?

Axios Installallation

npm i axios

API From which we will fetch data:

Frontend and Output

```

import React, { Component } from "react";
import axios from "axios";

class App extends Component {
  constructor() {
    super();
    this.state = {
      postList: [],
    };
  }
  componentDidMount() {
    axios
      .get("https://jsonplaceholder.typicode.com/comments", {})
      .then((response) => {
        this.setState({
          postList: response.data,
        });
      })
      .catch((err) => {
        console.log(err);
      });
  }
  render() {
    return (
      <>
        <div>
          {this.state.postList.map((item) => {
            return (
              <div key={item.id}>
                <div>
                  <span>Name : </span>
                  <span>{item.name}</span>
                </div>
                <div>
                  <span>Email : </span>
                  <span>{item.email}</span>
                </div>
                <div>
                  <span>Body : </span>
                  <span>{item.body}</span>
                </div>
                <br/>
              </div>
            );
          })}
        </div>
      </>
    );
  }
}
export default App;

```

Name : id labore ex et quam laborum
 Email : Eliseo@gardner.biz
 Body : laudantium enim quasi est quidem magnam voluptate

Name : quo vero reiciendis velit similique earum
 Email : Jayne_Kuhic@sydney.com
 Body : est natus enim nihil est dolore omnis voluptatem

Name : odio adipisci rerum aut animi
 Email : Nikita@garfield.biz
 Body : quia molestiae reprehenderit quasi aspernatur aut ducimus et vero voluptates excepturi deleniti ratione

Name : alias odio sit
 Email : Lew@afysha.tv
 Body : non et atque occaecati deserunt quas accusantium

Name : vero eaque aliquid doloribus et culpa
 Email : Hayden@althea.biz
 Body : harum non quasi et ratione tempore iure ex voluptate

Name : et fugit eligendi deleniti quidem qui sint nihil aut
 Email : PresleyMueller@myrl.com
 Body : doloribus at sed quis culpa deserunt consectetur qui

Name : repellat consequatur praesentium vel minus molestiae
 Email : Dallas@ole.me
 Body : maiores sed dolores similique labore et inventore

Name : et omnis dolorem
 Email : Mallory_Kunze@marie.org
 Body : ut voluptatem corrupti velit ad voluptatem maiores

Name : provident id voluptas
 Email : Meghan_Littel@rene.us
 Body : sapiente assumenda molestiae atque adipisci laborum

Name : eaque et deleniti atque tenetur ut quo ut
 Email : Carmen_Keeling@caroline.name
 Body : voluptate iusto quis nobis reprehenderit ipsum amet

Why do we make a call back function after Axios' request ?

Since Axios is a promise-based HTTP client-side library hence to wait for a promise to resolve or reject we need to make a callback there and get a response according to promise resolve and reject

III. AXIOS GET REQUEST

CRUD: CRUD stands for

Create, Read, Update, and Delete

which are four primitive database operations.

What are GET and POST methods and what is the difference between them?

Both GET and POST method is used to transfer data from client to server in HTTP protocol but the Main difference between the POST and GET method is that GET carries request parameter appended in URL string while POST carries request parameter in the message body

```

// Requested parameter name : 'Rina' city : 'Bangalore'
axios.get("http://localhost:8000/?name=Rina&city=Bangalore") // in get method requested parameters are append in url
axios.post("http://localhost:8000/updateData",{
  name: 'Rina',
  city: 'Bangalore'
})
// In post method requested parameters are within message body

```

What is cors ?

“CORS” stands for Cross-Origin Resource Sharing. CORS enables you to access a resource from a different origin. It is used to override your browser's default behavior due to SOP. So now when your client requests a resource, the response will additionally contain a stamp that tells your browser to allow resource sharing across different origins.

CORS INSTALLATION

npm i cors

How to get data from the backend through the get method?

Run your frontend and backend on different ports

Ports : a virtual point where network connections start and end .

Frontend and output

```
end > src > components > App.js > App > componentDidMount
import React, { Component } from "react";
import axios from "axios";

class App extends Component {
  constructor() {
    super();
    this.state = {
      postlist: [],
    };
  }
  componentDidMount() {
    axios
      .get("http://localhost:8000/")
      .then((response) => {
        this.setState({
          postlist: response.data,
        });
      })
      .catch((err) => {
        console.log(err);
      });
  }
  render() {
    return (
      <>
        {this.state.postlist.map((item) => {
          return (
            <div key={item.id}>
              <div>
                <span>Name : </span>
                <span>{item.name}</span>
              </div>
              <div>
                <span>Email : </span>
                <span>{item.email}</span>
              </div>
              <div>
                <span>Body : </span>
                <span>{item.body}</span>
              </div>
            </div>
          );
        })}
      </>
    );
  }
}
export default App;
```

localhost:3001
Name : id labore ex et quam laborum
Email : Eliseo@gardner.biz
Body : laudantium enim quasi est quidem magnam voluptate ipsam eos tempora quo necessitatibus c

Name : quo vero reiciendis velit similique earum
Email : Jayne_Kuhic@sydney.com
Body : est natus enim nihil est dolore omnis voluptatem numquam et omnis occaecati quod ullam at

Name : odio adipisci rerum aut animi
Email : Nikita@garfield.biz
Body : quia molestiae reprehenderit quasi aspernatur aut expedita occaecati aliquam eveniet laudanti
voluptates excepturi deleniti ratione

Name : alias odio sit
Email : Lew@alysha.tv
Body : non et atque occaecati deserunt quas accusantium unde odit nobis qui voluptatem quia volupt

backend

```

const { response } = require('express')
const express = require('express')
const PORT = 8000
let cors = require('cors');
let app = express()
app.use(cors());

const postlist = [
  {
    "postId": 1,
    "id": 1,
    "name": "Id labore ex et quam laborum",
    "email": "Eliseo@gardner.biz",
    "body": "laudantium enim quasi est quidem magnam voluptate ipsam eos\ntempora quo necessitatibus\ndolor quam autem quasi\nreiciendis e
  },
  {
    "postId": 1,
    "id": 2,
    "name": "quo vero reiciendis velit similique earum",
    "email": "Jayne_Kuhic@sydney.com",
    "body": "est natus enim nihil est dolore omnis voluptatem numquam\net omnis occaecati quod ullam at\mvoluptatem error expedita pariatur
  },
  {
    "postId": 1,
    "id": 3,
    "name": "odio adipisci rerum aut animi",
    "email": "Wikita@garfield.biz",
    "body": "quia molestiae reprehenderit quasi aspernatur\naut expedita occaecati aliquam eveniet laudantium\nomnis quibusdam delectus sa
  },
  {
    "postId": 1,
    "id": 4,
    "name": "alias odio sit",
    "email": "Lea@alysa.tv",
    "body": "non et atque\occaecati deserunt quas accusantium unde odit nobis qui voluptatem\nquia voluptas consequuntur itaque dolor\net
  }
]

app.get('/', (request, response) => {
  console.log(request.query, 'QUERY RESW')
  response.send(postlist)
})

app.listen(PORT, () => {
  console.log("server is running")
})

```

IV. AXIOS POST REQUEST

JsonParser: the base class to define public API for reading JSON content

Frontend and Output

Successful

```
1 import React, { Component } from "react";
2 import axios from "axios";
3
4 class App extends Component {
5   constructor() {
6     super();
7     this.state = {
8       messageToDisplay: " ",
9     };
10  }
11  componentDidMount() {
12    axios
13      .post("http://localhost:8000/updateData", {
14        name: "Rina",
15        City: "Benglore",
16      })
17      .then((response) => {
18        if (response.status === 200) {
19          this.setState({
20            messageToDisplay: "Successful",
21          });
22        } else {
23          this.setState({
24            messageToDisplay: "unsuccessful",
25          });
26        }
27      })
28      .catch((err) => {
29        console.log(err);
30      });
31  }
32  render() {
33    return (
34      <>
35        <h1>{this.state.messageToDisplay}</h1>
36      </>
37    );
38  }
39 }
40 export default App;
```

backend

```
const { response } = require("express");
const express = require("express");
const PORT = 8000;
let cors = require("cors");
let app = express();
app.use(cors());

app.post("/updateData", bodyParser.json((request, response) => {
  response.sendStatus(200);
}));

app.listen(PORT, () => {console.log("server is running")});
```

SASS OVERVIEW

I. SASS INTRODUCTION

CSS Cascading Style Sheet

SASS Syntactically Awesome Style Sheet

SCSS Syntactically Cascading Style Sheet

What is SASS ?

SASS is a better way of writing CSS by providing variables, Functions and other features which makes the CSS files code more maintainable .

It uses indentations

It is also known as Intended CSS .

What is SCSS ?

It is similar to SASS(See above SASS definition) It uses brackets rather than indentations.

It is also known as Sassy CSS, It is the latest version of SASS i.e.SASS3.

What is the Similarity and difference between SASS and SCSS?

SIMILARITY

Both SASS and SCSS are extension languages of CSS that uses variables and Nesting modules .

DIFFERENCE

The major difference between SASS and SCSS is of syntax .

SASS Follows indentation architecture

SCSS Follows bracket architecture

What are the advantages and disadvantages of using SASS?

Advantages

1. We can Write codes easily and efficiently and they are easy to maintain.
2. It facilitates reusability methods, logic statements, and some of the built-in functions like color manipulation, mathematics, and parameter lists item.
3. SASS allow writing clean CSS code and avoid code redundancy.
4. SASS is a superset of CSS which will help the developers in writing the code more efficiently and quickly.
5. SASS is also compatible with different kinds of CSS versions and any available CSS libraries can be used.
6. It has a powerful syntax that will help in writing the short and efficient code.

Disadvantages

1. Using Sass may cause losing benefits of browser's built-in element inspector.
2. The developer must have enough time to learn new features present in this preprocessor before using it.
3. It has a more complex syntax than CSS.

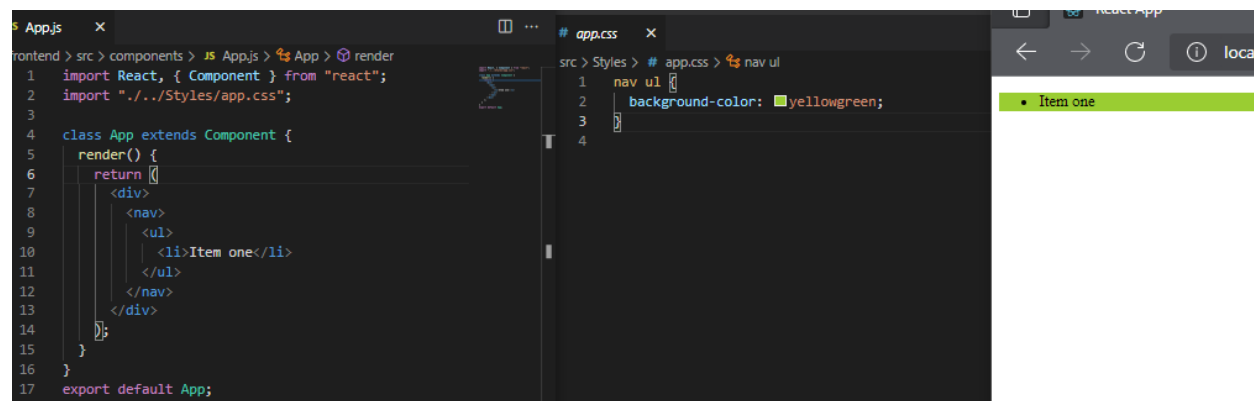
How to declare and write code in .css .sass .scss file

CSS File Extension : filename.css

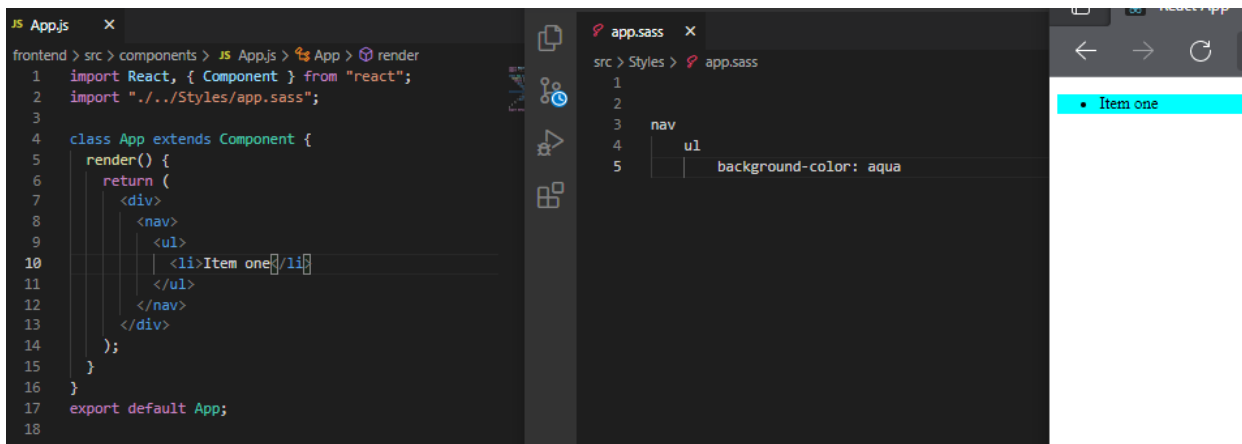
SASS File Extensions : filename.sass

SCSS File Extensions : filename.scss

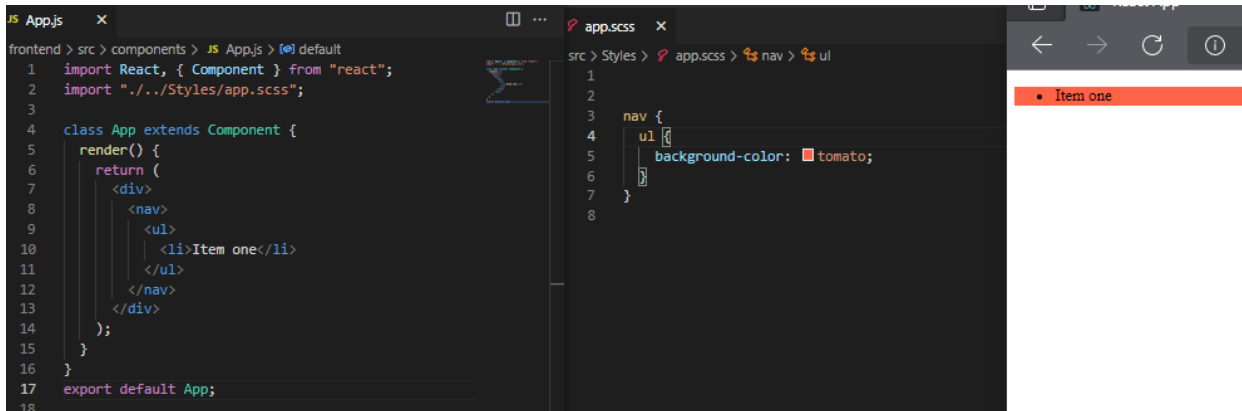
CSS File



SASS File



SCSS File



II. SASS INSTALLATION

Installation

npm install sass

How is SASS different from CSS ?

SASS is a superset of CSS, SASS is a CSS pre-processor that will allow using the different operations and using variables, mixins, functions, imports, and different kinds of loop.

What are the different types of operations in SASS ?

The different types of operations used in SASS are

- Number Operations
 - List Operations
 - String Operations
 - Color Operations
 - Boolean Operations
1. **Number operations:** It involves operations such as addition, subtraction, multiplication, and division.
 2. **List Operations:** It involves operations such as a set of values that are separated by commas in the form of an array.
 3. **String Operations:** It involves operations such as concatenation or splitting the strings.
 4. **Color Operations:** it involves operations such as using the Color components with arithmetic operations
 5. **Boolean Operations:** It involves operations such as using AND, OR, NOT operators to perform Boolean operations.

III. SASS VARIABLES

What is a variable and how is it defined in SASS ?

A Variable in SASS is used to store the information which can be reused throughout the style sheet.

These variables should be declared at the top of the file . A variable will be defined by starting it with a dollar (\$) sign.

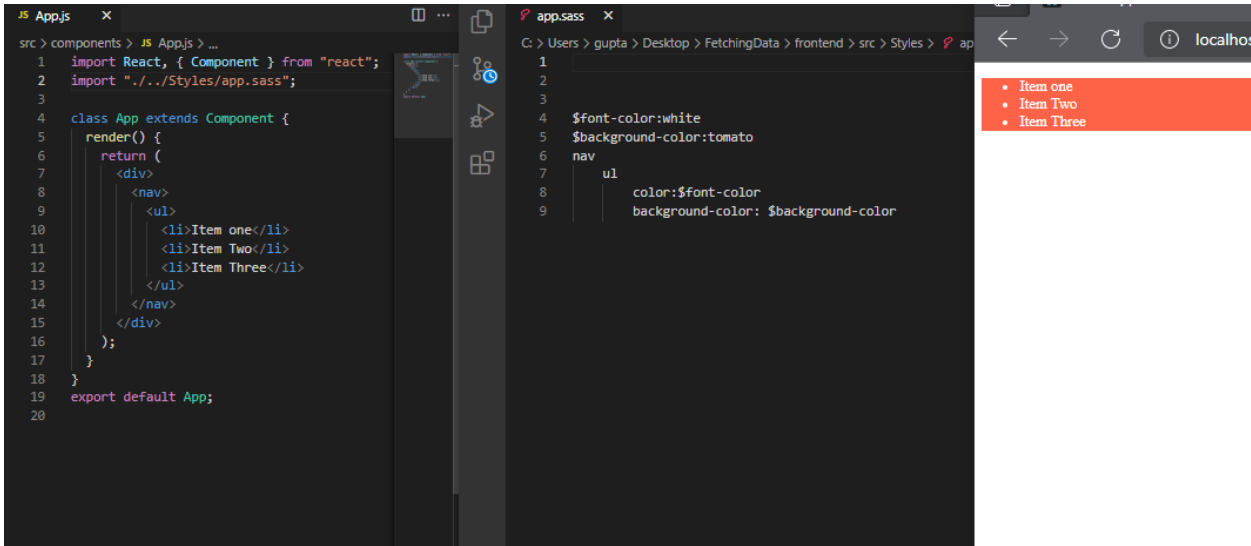
Example :

f
o
n

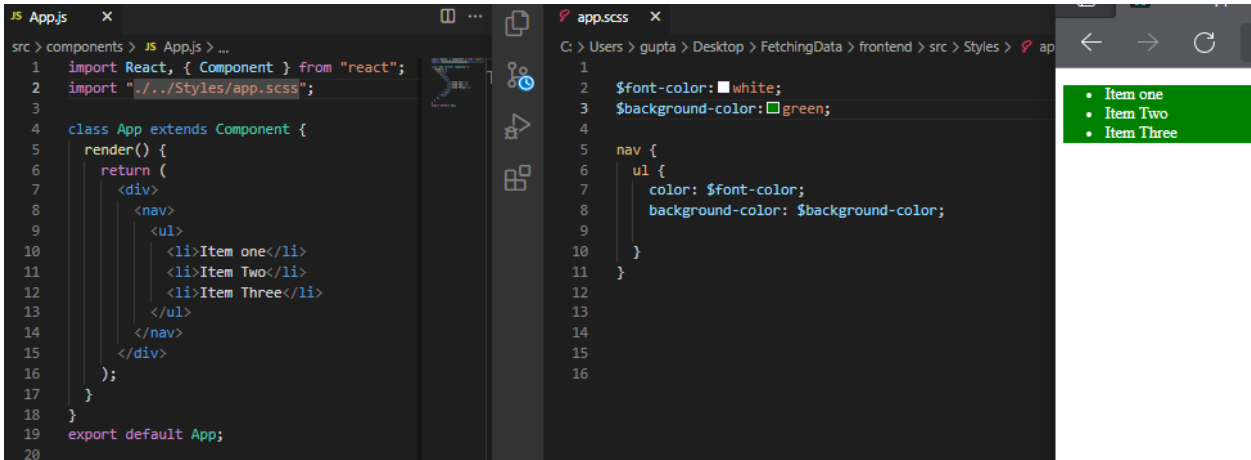
t
:
T
i
m
e
s
N
e
w
R
o
m
a
n
;

font:TimesNewRoman;
color: #222;

Variable declaration in sass



Variable declaration in scss

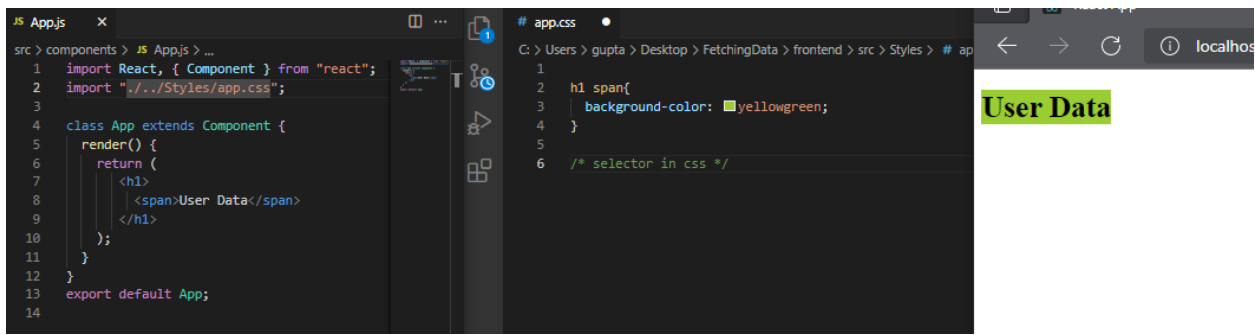


IV. SASS NESTING

What is a CSS selector ?

In CSS, pattern matching rules determine which style rules apply to elements in the document tree. These patterns, called selectors.

CSS selectors select HTML elements according to their id, class, type, attribute, etc .



What is Nesting in SASS?

Nesting is a shortcut to creating CSS rules. Nesting in SASS works as selector of multiple CSS and combine them within one another instead of writing different CSS lines just to be precise about the style that we want to add to an element, we just nest it .

V. SASS MIXINS

What is mixin in sass ?

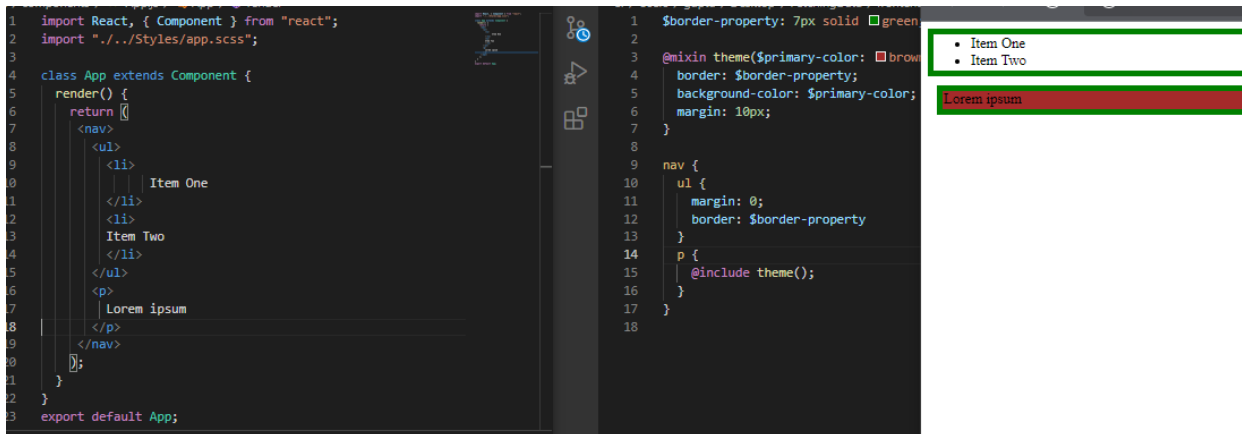
Mixin are like functions , but they are not functions We declare or define these functions at the top of the file using @mixin keyword We call or use these mixin using @include keyword

When should mixin used ?

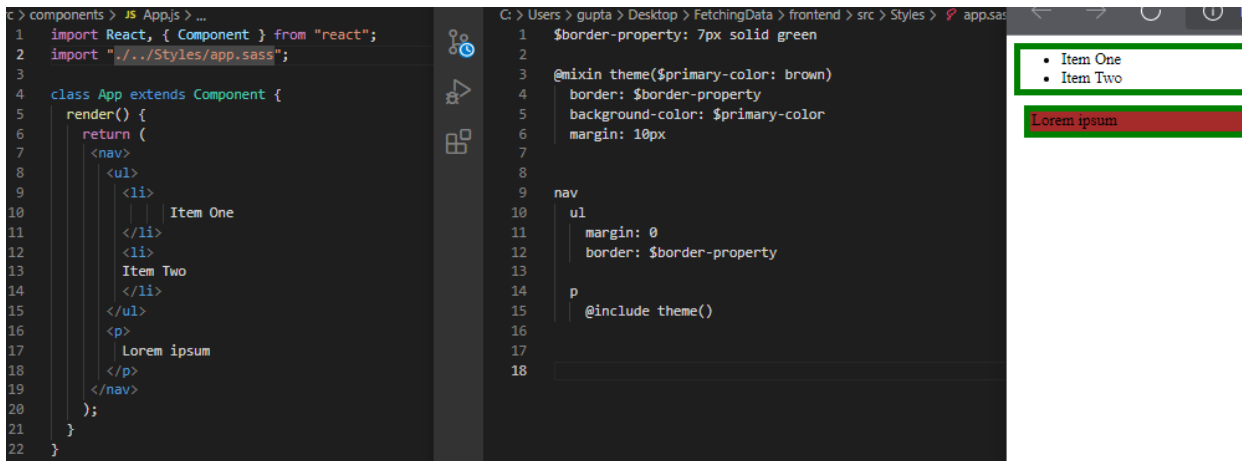
1. When we want to provide a lot of optional feature for a class
2. When we want to use a particular feature in lot of different class.

How to use mixin in sass and scss?

Mixin in scss



Mixin in SASS



What is the difference between mixin and extend in sass ?

@mixin is used to group CSS code that has to be reused a number of times

@extend is used to inherit properties from another css selector.

VI. SASS MODULES

What are Modules in sass ?

module is a unit of code contains in partial

In modules, we create one scss file separately and we need not to write all the scss content separately we just need to import that module using the keyword **@use** module filename.

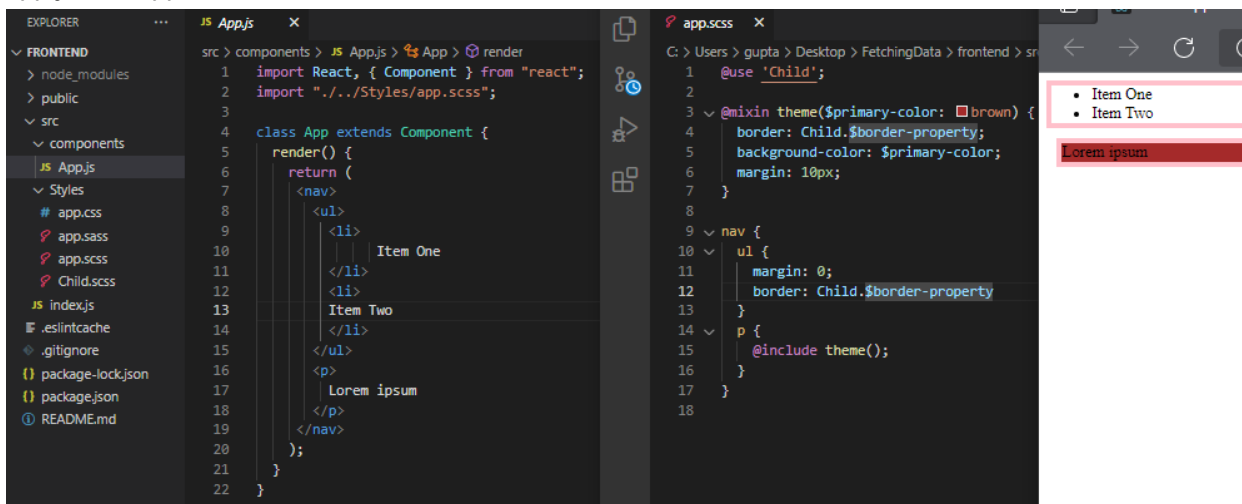
How to use modules in sass ?

Create a scss file separately and then write all the scss content separately To import that module in scss use keyword **@use** filename

Child.scss

```
Child.scss x
src > Styles > Child.scss > ...
1 $border-property: 5px solid pink;
2
```

App.js and App.scss



```
EXPLORER
...
JS App.js x
src > components > JS App.js > App > render
1 import React, { Component } from "react";
2 import "../Styles/app.scss";
3
4 class App extends Component {
5   render() {
6     return (
7       <nav>
8         <ul>
9           <li>Item One
10          </li>
11          <li>Item Two
12          </li>
13          </ul>
14          <p>Lorem ipsum
15          </p>
16        </nav>
17      );
18    }
19  }
20 }
21
22 }

app.scss x
C: > Users > gupta > Desktop > FetchingData > frontend > sr
1 @use "Child";
2
3 @mixin theme($primary-color: brown) {
4   border: Child.$border-property;
5   background-color: $primary-color;
6   margin: 10px;
7 }
8
9 nav {
10   ul {
11     margin: 0;
12     border: Child.$border-property;
13   }
14   p {
15     @include theme();
16   }
17 }
18
```