Q. What is React?

React is an open source JavaScript library used for building user interface (UI),especially React is used for designing single-page applications. It was developed by Facebook(now Meta) and maintained by Facebook also with community developers.

Q. Why use React JS?

1. **Component Based Architecture:** React breaks the UI into reusable components , each managing its own state this makes the developer modular and maintainable.

   Scenario

   Imagine your building an e-commerce site. The Product Card,Shopping Cart and navigation bar can each be individual components once created the product card can be reused on the home page ,search page or recommendation section with different data.
2. **Virtual DOM for Better performance**

   React uses a virtual DOM to minimize the direct manipulation of the Actual DOM which is slow. It calculates the most efficient way to update the UI.

   Scenario:
   Suppose on a social media app like facebook ,if someone likes a post only that post like count should update without reloading the whole feed . React virtual DOM ensure this update is fast and smooth and after that update on actual DOM

3. **Unidirectional data flow :** Data in React flows from parent to child components , making it easier to trace how data moves where changes happened.

   Example: In a dashboard , the main component may fetch data and pass it down to child components like charts or tables. This structure helps isolate bugs when a chart does not update correctly

4. **Rich Ecosystem and tooling**

   React has powerful developer tools ,state management libraries like a s Redux, Zustand) and routing (react router) making app development easier

5. **Easy to learn and adopt**

   React has a simple API and uses JavaScript syntax so developer easily understand the code flow of React Application
   Etc

How to develop the application using React

Steps.

1. **Download the node js and install it**
   https://nodejs.org/en/download

```
C:\Users\Hp>node -v
v22.16.0

C:\Users\Hp>npm --v
11.4.1

C:\Users\Hp>
```

2. **Create React Application using vite**

Q. What is vite?

Vite is a modern build tool and development server for web projects ,especially in the JavaScript and TypeScript ecosystem.
It is designed to provide fast startup ,instant hot module replacement and optimized production builds

Why use vite?

1. **Lighting fast development server:**

   Uses native ES modules and browser support to serve files instantly without binding.
2. Optimized Production Build
3. Support to Vue,React etc
4. Rich Plugin Ecosystem
5. Zero config setup

How to create React Project using vite build tool

If we want to create react project using vite build tool we have following syntax

Syntax:
npm create vite@latest

Example:

```
D:\july 2024\reactprojects\first>npm create vite@latest
```

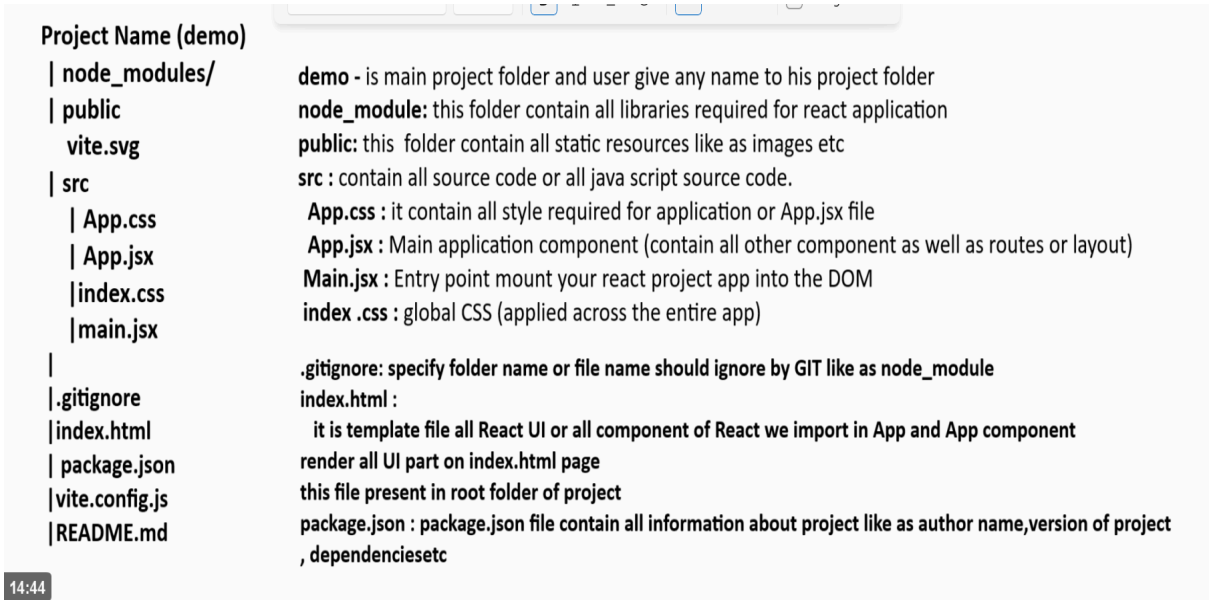
Once we create project then we required to create node_module folder in project
And for that we required to use npm install command in project
So we need to enter in project folder i.e demo in our example and run command npm install

```
D:\july 2024\reactprojects\first\demo>npm install

added 198 packages, and audited 199 packages in 1m

33 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

D:\july 2024\reactprojects\first\demo>
```

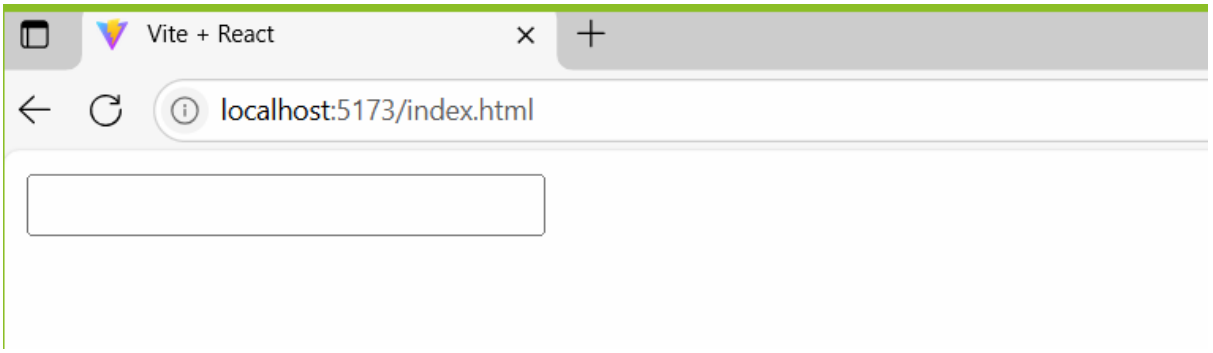
Once we install node_module in react project after that you can run project and test output

```
D:\july 2024\reactprojects\first\demo>npm run dev

> demo@0.0.0 dev
> vite


  VITE v6.3.5  ready in 5481 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

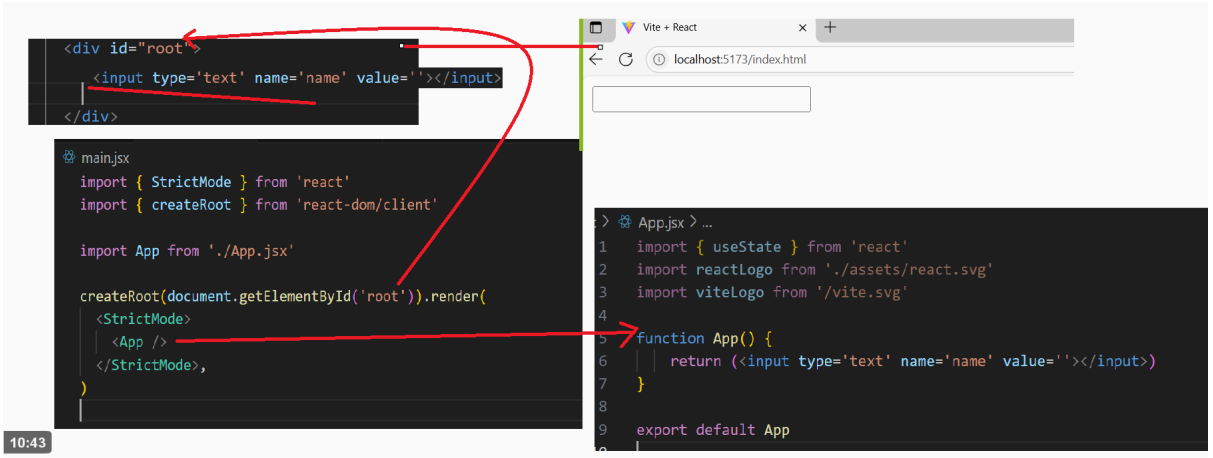
Now we want to understand the folder structure of React



Example:



Flow of code



Example with source code

App.jsx

```jsx
import { useState } from 'react'
import reactLogo from './assets/react.svg'
import viteLogo from '/vite.svg'

function App() {
    return (<input type='text' name='name' value=''></input>)
}

export default App
```

Main.jsx

```jsx
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'

import App from './App.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App />
  </StrictMode>,
)
```

Index.html

```html
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
    <title>Vite + React</title>
  </head>
  <body>
    <div id="root">


    </div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

Note: if we think about App.jsx we have one tag
<input type='text' name='name' value='' ></input>
We think this is HTML tag but it is not HTML tag it is JSX element

Q. What is JSX?

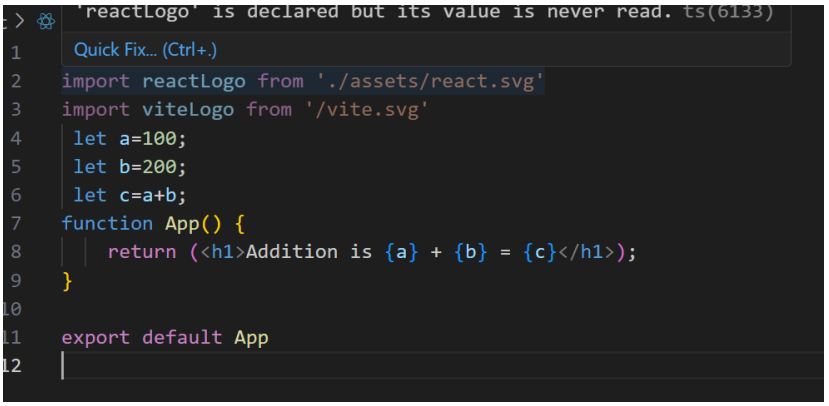
JSX (JavaScript XML) is a syntax extension used in React that allow you to write HTML like code within JavaScript
It is not required in React but it makes writing components more readable and easier

```
let element=<h1>Good Evening</h1>

this is look like as HTML but it is not HTML . It is JSX element and React convert it internally

let element =React.createElement("h1",null,"Good Evening");
```

Q. What are the benefits of JSX?

---

1. **Declarative UI :** it allow describe UI in more readable and declarative way like as HTML
2. **Combine Logic and Layout:** you can add run time data as UI output using JSX expression



3. **Component composition :** using JSX we can pass data from one component to another component using props or nesting component

Rules of JSX

---

1. **JSX must return a single parent element**
   If we try to return more than one JSX element from component then we required return as single element by adding as child of another element or they must have parent element like div or ul etc

2. Use className instead class if we apply CSS on JSX element

3. **Tag closing must**

Q. What is babel ?

Babel is a JavaScript compiler that is commonly used to transform JSX to modern JavaScript code(ES6) into browser understandable code.

Babel convert new JavaScript syntax like as arrow function, let/const ,classes
Into older syntax that older browser can run
Means ES6 to ES5

React Fragment

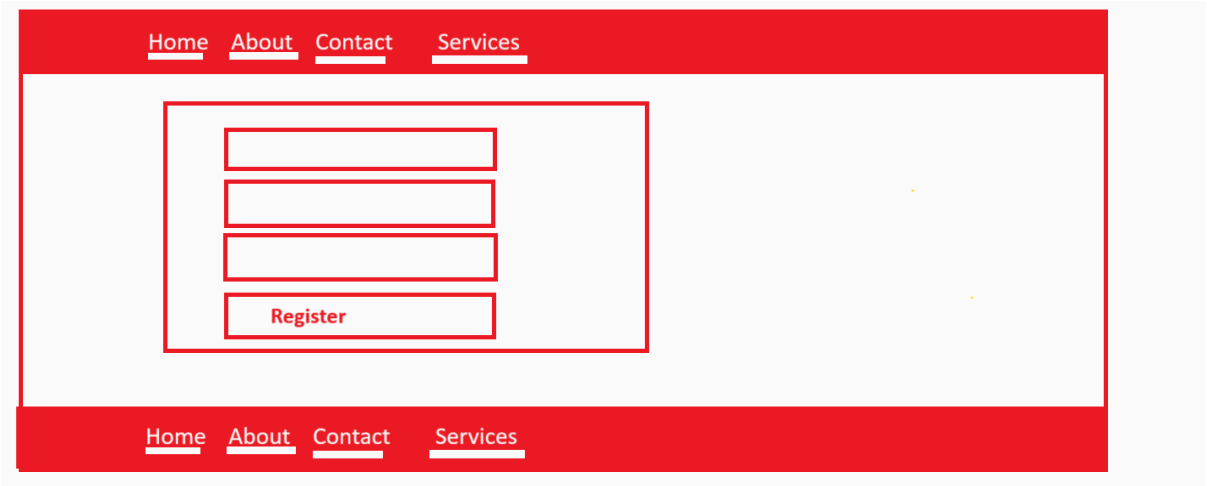
React Fragment is used for return multiple JSX element as single bunch of element from component and render on UI part

```
import React, { useState } from 'react'
import reactLogo from './assets/react.svg'
import viteLogo from '/vite.svg'
let a=100;
let b=200;
let c=a+b;

function App() {
    return (
      <React.Fragment>
            <h1 className="headColor">First Value {a}</h1><br/>
           <h1 className="headColor">Second Value {b}</h1>
           <h1 className="headColor">Addition is {c}</h1>
      </React.Fragment>
    );
}

export default App
```

Note: You can use sugar form of React Fragment like as <> </>

```jsx
import React, { useState } from 'react'
import reactLogo from './assets/react.svg'
import viteLogo from '/vite.svg'
 let a=100;
 let b=200;
 let c=a+b;

function App() {
    return (
      <>    <h1 className="headColor">First Value {a}</h1><br/>
            <h1 className="headColor">Second Value {b}</h1>
            <h1 className="headColor">Addition is {c}</h1>
      </>
    );
}
export default App
```

Assignment



index.css

```css
ul{
  width:100%;
  background-color:red;
  padding:20px;
}
ul li{
  display:inline-block;
  padding:10px;
}
ul li a{
  text-decoration: none;
```

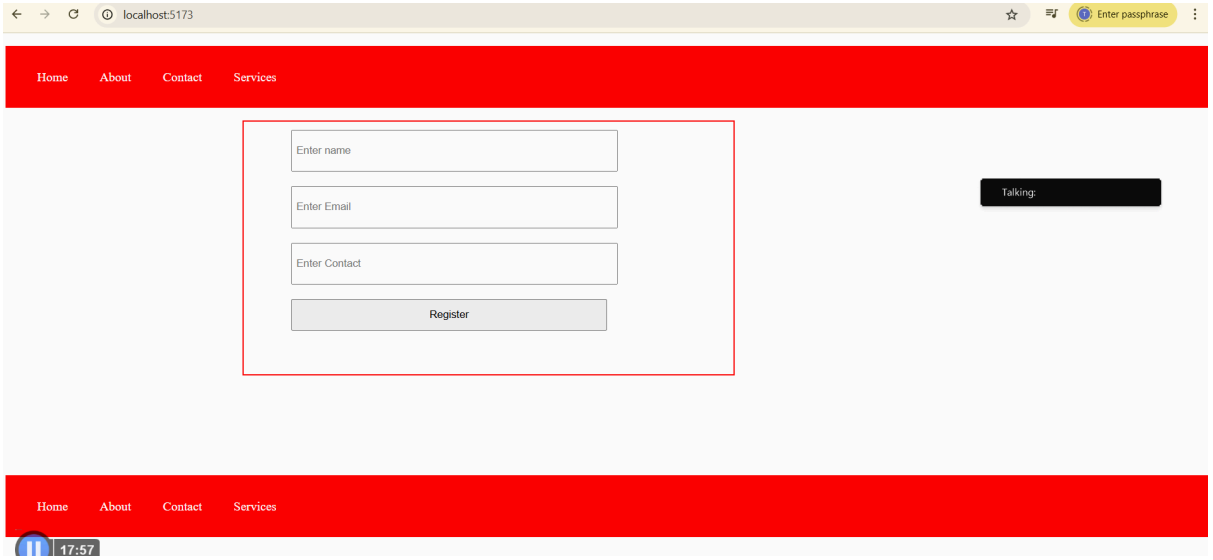
```
  padding:10px;
  color:white;
}
.container{
  width:600px;
  height:300px;
  border:2px solid red;
  padding:10px;
  margin-left:300px;
}
input{
   width:400px;
   height:40px;
   margin-left:50px;
   padding:5px;
}
```

App.jsx

```
import { useState } from 'react'
import reactLogo from './assets/react.svg'
import viteLogo from '/vite.svg'
 function App() {
  return (
    <>
     <ul>
       <li><a href=''>Home</a></li>
       <li><a href=''>About</a></li>
       <li><a href=''>Contact</a></li>
       <li><a href=''>Services</a></li>
     </ul>
     <div className='container'>
        <input type='text' name='name' id='name' value=''
placeholder='Enter name '/><br/><br/>
        <input type='text' name='email' id='name' value=''
placeholder='Enter Email '/><br/><br/>
        <input type='text' name='contact' id='name' value=''
placeholder='Enter Contact '/><br/><br/>
        <input type='submit' name='s' id='name'   value='Register'
/><br/><br/>
     </div>
     <br/><br/><br/><br/><br/><br/>
     <ul>
```

```
        <li><a href=''>Home</a></li>
        <li><a href=''>About</a></li>
        <li><a href=''>Contact</a></li>
        <li><a href=''>Services</a></li>
      </ul>
    </>
  )
}
export default App
```

Output



Note: if we think App.jsx file and it code there is some limitation

1. We use same code two times for generate header and footer UI
   The limitation unnecessary boilerplate code generated and if we want to perform change in code we need to perform changes at both places according to our example if we add new link header and if we want to show same link in footer then we must be add same code in footer it will take more time for modification if we have large and complex UI
2. We write all code in App.jsx file for geneate UI if we want to generate home page ui, about us page ui etc then we need to write all code in single file so it is very complicated to manage UI in application and for future changes

If we want to solve this problem, React suggests we create separate components for every UI part.

Q. What is the component?

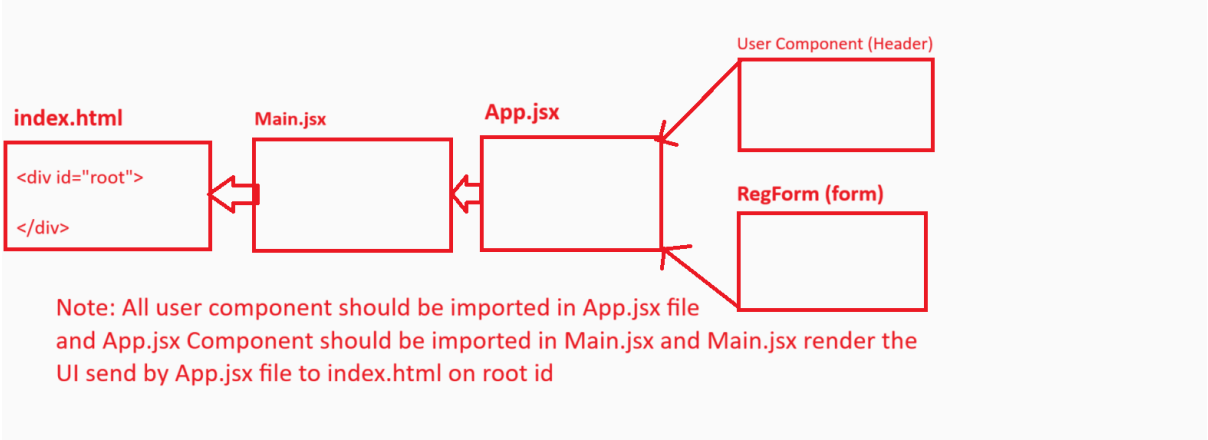
Component is block of UI code or individual block of code and we can reuse it more than one time according to our need as well as segregate or separate UI parts and integrate them where we required

Q. How to create components in React?

If we want to create component in React we have there are two ways

1. **Class component**
2. **Function component**

If we want to work with component we have following standard structure



How to create class component in React

Steps to create class component

---

1. **Create user define class and inherit the React.Component in it**

```
Syntax:
 class classname extends React.Component
 {
 }
 Example:
 class Header extends React.Component
 {

 }
```

2. **Override render() method of Component**

```
class  Header extends React.Component
{
     render()
     {
       return  <> JSX elements </>
     }
}
```

3. **Export component :** component export indicate we are creating object of the component here

   There are two ways to export the component

   ---

   a. **After component creation**

```
Syntax :   export  default componentname;
```

```
class  Header extends React.Component
{
    render()
    {
      return  <> JSX elements </>
    }
}
export  default Header;
```

   b. **At the time of component creation**

```
export  default class classname extends React.Component
{
     render(){
        return JSX
     }
}
```

```
Example:
export default Header extends React.Component
{
     render(){
        return <></>
     }
}
```

4. Import component we need or where we want to use
Note: if we want to reuse any component in React we required to import it

Syntax for import component

```
import  componentname from "./filename.jsx";
```

Example: import Header from "./Header.jsx";

5. Use Component as JSX element
<componentname />
Example: <Header />

Example: we want to create HellowWord component which return <h1>Welcome in class component </h1> JSX element and we want to import component in App.jsx file i.e in App component and use it

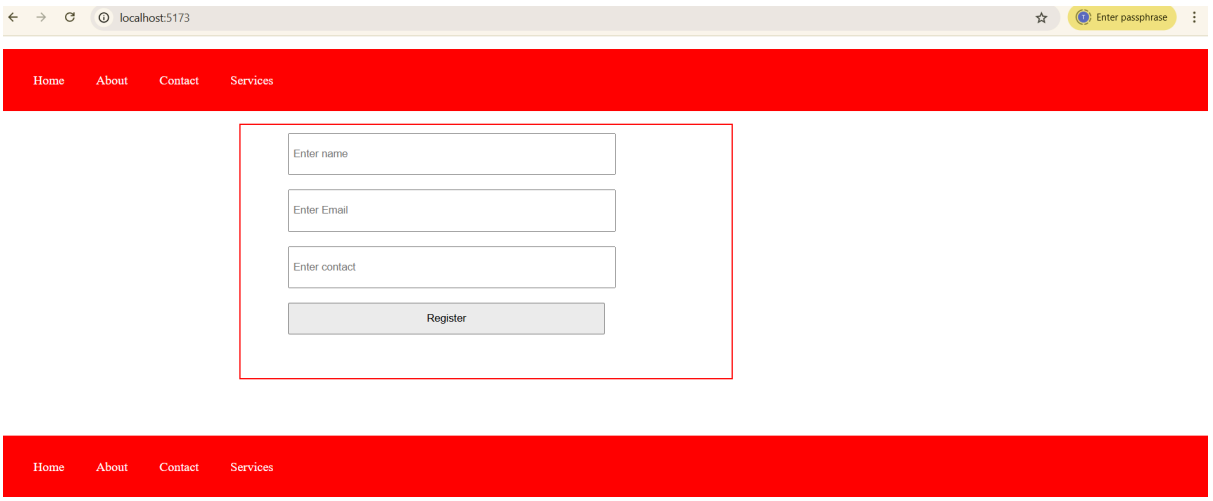
HelloWord.jsx

```
import React from "react";

class HelloWord extends React.Component{

     render(){
        return <h1>Welcome in class component </h1>
     }
}
export default HelloWord;
```

App.jsx

```
import HelloWord from "./HelloWord"
 function App() {
  return (<><HelloWord/></> )
}
export default App
```

Example: Design following UI using class component



Header.jsx

```
import React from "react";

class Header extends React.Component{

     render(){
        return <ul>
             <li>
               <a href=''>Home</a>
             </li>
             <li>
               <a href=''>About</a>
             </li>
             <li>
               <a href=''>Contact</a>
             </li>
             <li>
               <a href=''>Services</a>
             </li>
        </ul>
     }
}
export default Header;
```

Register.jsx

```
import React from "react";

export default class Register extends React.Component{

     render(){
        return <>
           <div className='container'>
             <input type='text' name='name' value='' placeholder="Enter
name"/><br/><br/>
             <input type='text' name='name' value='' placeholder="Enter
Email"/><br/><br/>
             <input type='text' name='name' value=''
placeholder="Enter contact"/><br/><br/>
             <input type='submit' name='s' value='Register'/><br/>
```

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

Index.css

```
ul{
  width:100%;
  background-color:red;
  padding:20px;
}
ul li{
  display:inline-block;
  padding:10px;
}
ul li a{
  text-decoration: none;
  padding:10px;
  color:white;
}
.container{
  width:600px;
  height:300px;
  border:2px solid red;
  padding:10px;
  margin-left:300px;
}
input{
   width:400px;
   height:40px;
   margin-left:50px;
   padding:5px;
}
```

App.jsx

```
import Header from "./Header.jsx";
import Register from "./Register.jsx";
 function App() {
  return (<><Header/>
         <Register/>
```

```
    <br/><br/><br/>
          <Header/>
  </> )
}
export default App
```

How to use CSS in React

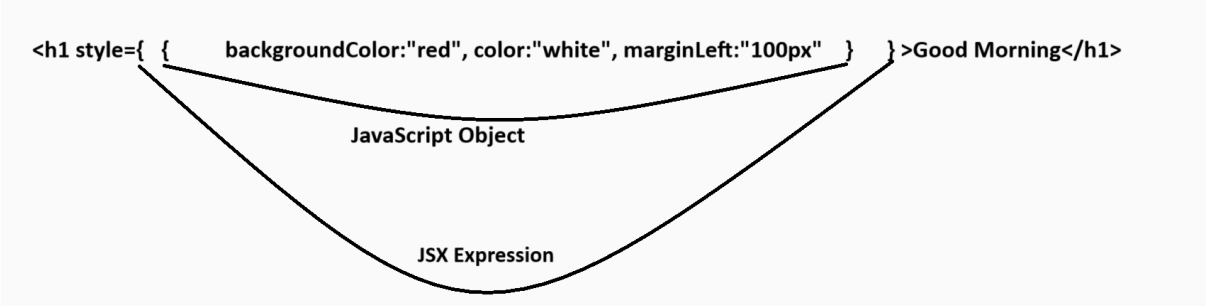
If we want to use CSS in React we have three ways

Rules of Use CSS in React

---

a. Use JavaScript object in JSX expression for apply CSS
b. Convert CSS property as JavaScript object key and value of property as value
c. Convert kebab case standard of CSS property to camel case

Example: we want to create <h1> as JSX element and apply style on it with background color red and text color white with left margin 100 px



1. **Inline CSS :** when we use CSS with a JSX element called inline CSS.
2. **Internal CSS : when we declare CSS with JSX file and use with JSX element called as internal CSS**
3. **External CSS :** external CSS means when create separate .css file in application use with JSX element called as external CSS

Example of inline CSS

App.jsx

```
import React from "react";

class App extends React.Component{
  render(){
    return<>
```

```
        <h1 style={
                    {
                       backgroundColor:"red",
                       color:"white",
                       marginLeft:"100px",
                       padding:"20px",
                       marginRight:"100px",
                       borderRadius:"50px"
                    }
                  }>
                    Good Evening
         </h1>
      </>
   }
}
export default App;
```

Output

Example of internal CSS

Internal CSS in React means we store all CSS property in one java script variable use with style property as JSX expression called as internal CSS

App.jsx

```
import React from "react";

let headingStyle= {
                       backgroundColor:"red",
                       color:"white",
                       marginLeft:"100px",
                       padding:"20px",
                       marginRight:"100px",
                       borderRadius:"50px"
                    };
class App extends React.Component{
  render(){
    return<>
       <h1 style={headingStyle}>Good Evening</h1>
```

```
        <h1 style={headingStyle}>Good Night</h1>
        <h1 style={headingStyle}>Good Morning</h1>
      </>
    }
}
export default App;
```

Output

External CSS

There are two ways to use external CSS

a. **Global css :** global css means we write CSS in index.css and we import in main.jsx file here we write style logics which for all components

b. **Component level CSS :** we design seperate CSS file for every component and write css logics in that file according to component

Example with source code

Index.css

```
h1{
  background-color:red;
  color:white;

}
```

Second.css

```
#secondHead{
    border-radius: 50%;
    width:400px;
    height:300px;
    padding:20px;
    margin-left:100px;
    margin-top:100px;
    text-align: center;
}
```

First.jsx

```
import React from "react";

class First extends React.Component{
    render(){
        return <><h1>I am first component</h1></>
    }
}
export default First;
```

Second.jsx

```
import React from "react";
import "./second.css";
class Second extends React.Component{
    render(){
        return <><h1 id="secondHead">Second component</h1></>
    }
}
export default Second;
```

App.jsx

```
import React from "react";
import First from "./First.jsx";
import Second from "./Second.jsx";
class App extends React.Component{
  render(){
    return<>
        <First/>
        <Second/>
     </>
  }
}
export default App;
```

How to use bootstrap in React

If we want to use bootstrap in React we have following ways

1. **Using online CDN**
2. Installing bootstrap dependency
3. Using React bootstrap package

Using online CDN

If we want to use online cdn for bootstrap add all CDN files in index HTML page and use bootstrap class with JSX element

Example we want to design registration form using bootstrap

Use following CDN

```
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/css/bootstrap.m
in.css"
integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJw
J8ERdknLPMO" crossorigin="anonymous">




<!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8ab
tTE1Pi6jizo" crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.14.3/dist/umd/popper.min.
js"
integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l
8WvCWPIPm49" crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/js/bootstrap.min
.js"
integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/Jm
ZQ5stwEULTy" crossorigin="anonymous"></script>
```

Index.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
```

```
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
    <title>Vite + React</title>
     <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/css/bootstrap.m
in.css"
integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJw
J8ERdknLPMO" crossorigin="anonymous">

  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>


     <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8ab
tTE1Pi6jizo" crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.14.3/dist/umd/popper.min.
js"
integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l
8WvCWPIPm49" crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/js/bootstrap.min
.js"
integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/Jm
ZQ5stwEULTy" crossorigin="anonymous"></script>
  </body>
</html>
```

App.jsx

```
import React from "react";

class App extends React.Component{
  render(){
    return<>
    <div className="container mt-5 bg-dark p-5">
       <div className="form-group">
```

```
        <input type='text' name='name' className="form-control"
value=''  placeholder="Enter Name" />
       </div>
       <div  className="form-group">
        <input type='email' name='email' className="form-control"
value='' placeholder="Enter Email" />
       </div>
       <div className="form-group">
        <input type='text' name='contact' className="form-control"
value='' placeholder="Enter Contact" />
       </div>
       <div className="form-group">
        <input type='submit' name='s' className="form-control btn
btn-primary" value='Register'  />
       </div>
       </div>
     </>
  }
}
export default App;
```

2. Install bootstrap as library by using npm

```
D:\july 2024\reactprojects\cssdemo>npm install bootstrap@latest

added 46 packages, and audited 202 packages in 8s

35 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

D:\july 2024\reactprojects\cssdemo>
```

Main.jsx

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import "./index.css";
import App from './App.jsx'
import "bootstrap/dist/css/bootstrap.min.css";
import "bootstrap/dist/js/bootstrap.min.js";

createRoot(document.getElementById('root')).render(
  <StrictMode>
```

```
    <App />
  </StrictMode>,
)
```

App.jsx

```
import React from "react";


class App extends React.Component{
  render(){
    return<>
    <div className="container mt-5 bg-dark p-5">
       <div className="form-group m-2">
        <input type='text' name='name' className="form-control"
value=''  placeholder="Enter Name" />
       </div>
       <div  className="form-group m-2">
        <input type='email' name='email' className="form-control"
value='' placeholder="Enter Email" />
       </div>
       <div className="form-group m-2">
        <input type='text' name='contact' className="form-control"
value='' placeholder="Enter Contact" />
       </div>
       <div className="form-group m-2">
        <input type='submit' name='s' className="form-control btn
btn-primary" value='Register'  />
       </div>
       </div>
     </>
  }
}
export default App;
```

How to manage the events in React using class component

If we want to handle event in react we have two ways

1. **Using non arrow function**
2. **Using arrow function**

Steps to handle the event using non arrow function or normal function

---

Important points related with non arrow function

1. **When we use non arrow function for manage the events we need to bind handler function with current object**
2. **When we use non arrow function class not need to write function keyword at the time of function definition**

Steps

---

1. **Create class component**

```
import React from "react";

export default class App extends React.Component{

   render(){
    return <h1>I am class component</h1>
   }
}
```

2. **Define event handler function within class component outside of render function**

```
import React from "react";

export default class App extends React.Component{

  //event handler function
  showTest()
  {

  }
   render(){
    return <h1>I am class component</h1>
   }
}
```

3. **Create constructor in class component and register the event handler with current object and call super constructor**

   If we want to bind the handle function with current object we have to use following syntax
   Syntax: this.handlername=this.handlername.bind(this);

Example with source code

```
import React from "react";

export default class App extends React.Component{

  //in constructor we bind handler function
  constructor(){
    super();
    this.showTest=this.showTest.bind(this);
  }
  //event handler function
  showTest()
  {

  }
   render(){
    return <h1>I am class component</h1>
   }
}
```

4. **Create JSX element and call event function using HTML events**
   Note: when we use html event with JSX element we required to convert event name in camel name like as onClick, onChane, onKeyUp etc

Note: we want to create a button using JSX and call the handler function on button click.

```
import React from "react";
import "bootstrap/dist/css/bootstrap.min.css";
import "bootstrap/dist/js/bootstrap.min.js";

export default class App extends React.Component{

  //in constructor we bind handler function
  constructor(){
    super();
    this.showTest=this.showTest.bind(this);
  }
  //event handler function
  showTest()
  {
   alert("Hey event executed");
  }
   render(){
    return <>
```

```
    <div className="container mt-5">
    <input type="button" className="btn btn-danger"
    value="Call Function" onClick={this.showTest} />
    </div>
    </>
   }
}
```

How to handle the event using arrow function

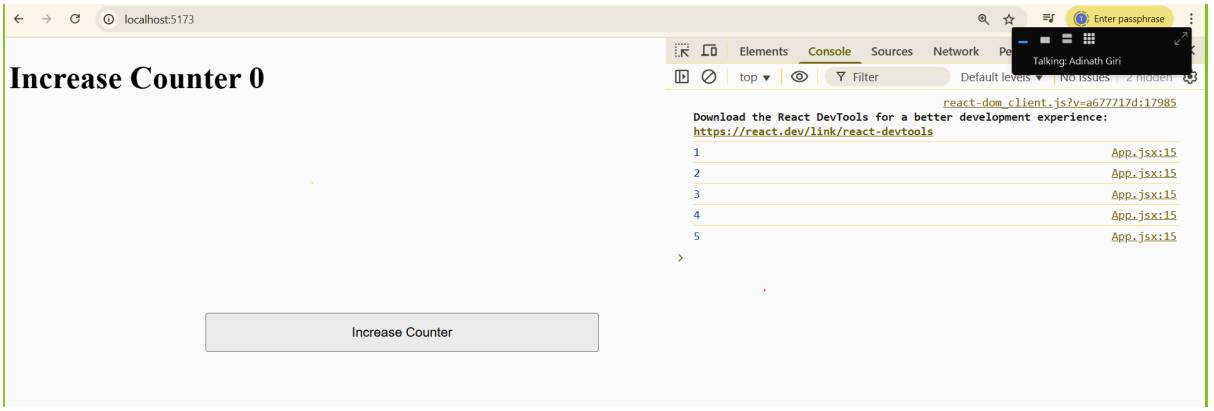
If we want to handle the event using arrow function then we do not need to bind the handler function with the current object as well as not need to use let ,var or const keyword with function definition in class.

```
import React from "react";
import "bootstrap/dist/css/bootstrap.min.css";
import "bootstrap/dist/js/bootstrap.min.js";

export default class App extends React.Component{


  //event handler function
  showTest=()=>
  {
   alert("Hey event executed");
  }
   render(){
    return <>
    <div className="container mt-5">
    <input type="button" className="btn btn-danger"
    value="Call Function" onClick={this.showTest} />
    </div>
    </>
   }
}
```

Example: we want to design a web page with one button & one one counter variable and when the user clicks on the button then we want to increase counter value by 1 and display it on the web page.

App.jsx

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

let btnCss={
   width:"400px",
   height:"40px",
   marginLeft:"200px",
   marginTop:"200px"
}
let count=0;
export default class App extends React.Component{

   incCounter=()=>{
     ++count;
      console.log(count);
   }
   render(){
      return <>
         <h1>Increase Counter {count}</h1>
        <input type="button" name="s" value="Increase Counter"
style={btnCss} onClick={this.incCounter} />
      </>
   }
}
```



Note: if we think about above output when we click on button then counter value increase by 1 but not render on component so not display on browser and when we show the counter value on console output then it is incremented by 1 and display in console

But we expect counter should be display on browser means we need to render the updated counter value in component so component can update in DOM
So if we want to update any variable value within a component and render on a web page means update in DOM we need to use the state variable concept in the class component.

Q. What is the state variable?

State variable is a concept in class component which work like as instance variable in java means state variable normally used for hold data in class and when we change data or update state variable then component render method automatically executed and show the updated data on browser or web page via DOM

How to use state variable in class component

Steps

1. **Declare constructor within class component and call super constructor**

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

let btnCss={
   width:"400px",
   height:"40px",
   marginLeft:"200px",
   marginTop:"200px"
}
let count=0;
export default class App extends React.Component{

    App(){
      super(); //call parent constructor

    }
   incCounter=()=>{
     ++count;
      console.log(count);
   }
   render(){
      return <>
         <h1>Increase Counter {count}</h1>
        <input type="button" name="s" value="Increase
Counter" style={btnCss} onClick={this.incCounter} />
      </>
   }
}
```

2. **Declare state variable in class component using a following syntax**

```
this.state={
  key:value
}
```

Note: if we think about state variables it is a javascript object which contains data in the form of key and value pair.

3. Update state variable

If we want to update state variable value we have a function name as setState() this function can accept javascript objects and update state variable value using its key.

```
this.setState({
    key: updatedvalue
  })
```

Example using state variable concept

```
import React from "react";
import ReactDOM from "react-dom";
let btnCss={
   width:"400px",
   height:"40px",
   marginLeft:"200px"
}
export default class App extends React.Component{
  constructor(){
      super(); //call parent constructor
      this.state={
```

```
        count:0
      }
    }
   incCounter=()=>{

       this.setState({count:this.state.count+1});
   }
   render(){
    console.log("I am executed");
      return <>
               <h1>Increase Counter {this.state.count}</h1>
        <input type="button" name="s" value="Increase Counter"
style={btnCss} onClick={this.incCounter} />
      </>
   }
}
```

Note: State variable help us to handle the form using React

How to work with form in React using State variable

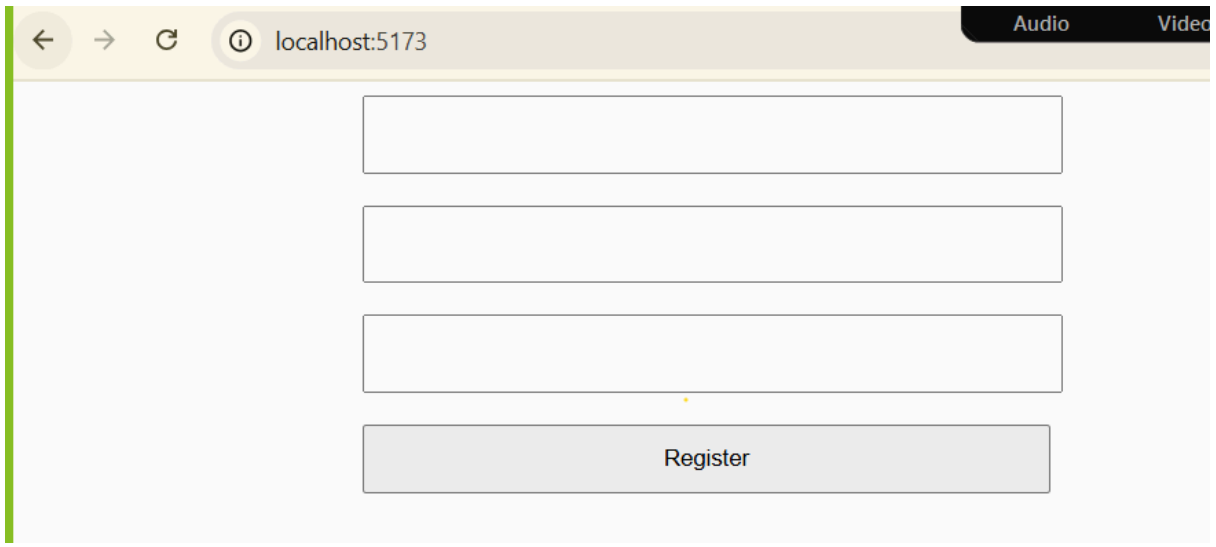
Example: we want to create registration with name email and contact and manage form using state variable

Example with source code

```
import React from "react";
import ReactDOM from "react-dom";
let btnCss={
   width:"400px",
   height:"40px",
   marginLeft:"200px"
}
export default class App extends React.Component{
  constructor(){
      super(); //call parent constructor
      this.state={
        count:0
      }
    }
   incCounter=()=>{

       this.setState({count:this.state.count+1});
```

```
    }
    render(){
     console.log("I am executed");
       return <>
                <input type='text' name='name' value=''
style={btnCss}/> <br></br><br></br>
                <input type='text' name='email' value=''
style={btnCss}/> <br></br><br></br>
                <input type='text' name='contact' value=''
style={btnCss}/> <br></br><br></br>
                <input type='button' name='s' value='Register'
style={btnCss}/> <br></br>

       </>
    }
}
```

Output



Note: if we think about above code we have registration form but when we input the data in textbox then textbox not accept data so if we want to make your form editable by user then we can use state variable

How to make form editable using React

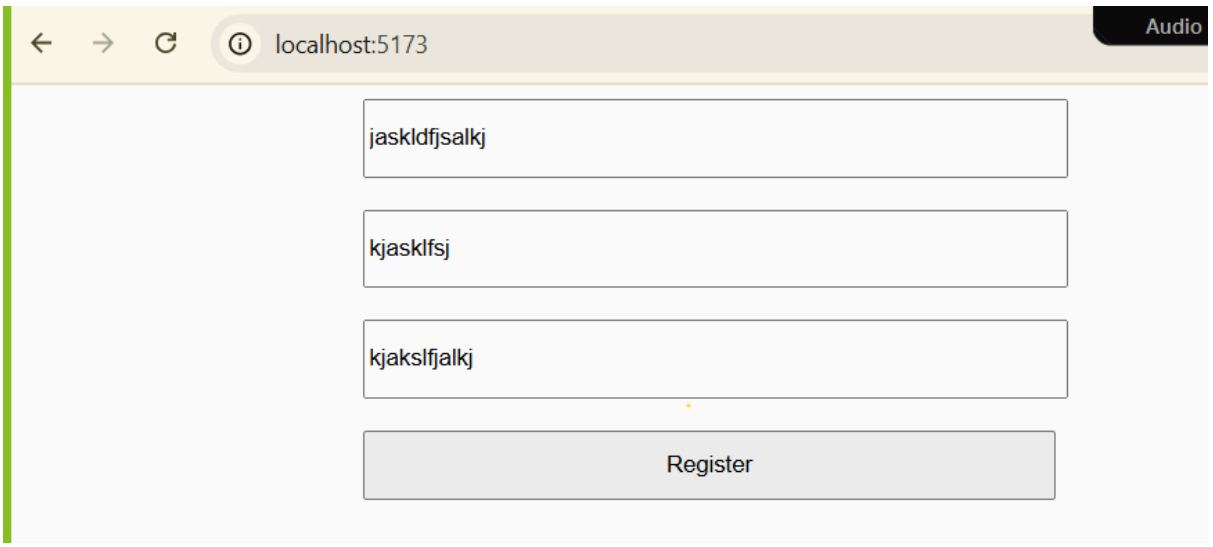
---

1. **Design the form (use previous example code)**
2. **Declare the state variable same name as with form control name**

Example with source code

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

```
let btnCss={
   width:"400px",
   height:"40px",
   marginLeft:"200px"
}
export default class App extends React.Component{
  constructor(){
      super(); //call parent constructor
       this.state={
         name:"",
         email:"",
         contact:""
       }
    }
    updateName=(e)=>{
         this.setState({name:e.target.value});
    }
    updateEmail=(e)=>{
       this.setState({email:e.target.value});
    }
    updateContact=(e)=>{
         this.setState({contact:e.target.value});
    }
   render(){
      return <>
     <input type='text' name='name' value={this.state.name}
style={btnCss}
      onChange={(e)=>this.updateName(e)} /> <br></br><br></br>
     <input type='text' name='email' value={this.state.email}
style={btnCss}
     onChange={(e)=>this.updateEmail(e)} /> <br></br><br></br>
    <input type='text' name='contact' value={this.state.contact}
style={btnCss}
    onChange={(e)=>this.updateContact(e)} /> <br></br><br></br>
    <input type='button' name='s' value='Register' style={btnCss}/>
<br></br>

      </>
   }
}
```



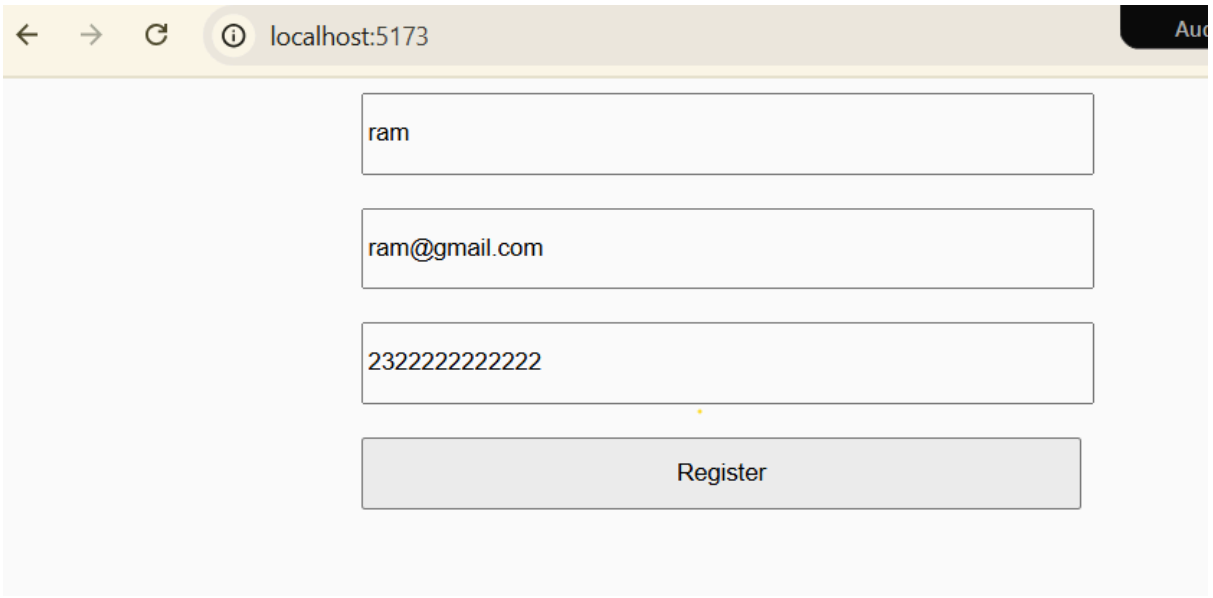
Note: if we think about above code we have three hander function because we have three field on form and every field we create one event handling function for name we have updatename,for email we have updateemail, for contact we have updatecontact
But we have 50 controls on your form then we are required to write 50 handler functions which is not possible in a real time scenario so we want to create a single handler function for all controls and manage its state.

Example with universal handler

```
import React from "react";
import ReactDOM from "react-dom";
let btnCss={
   width:"400px",
   height:"40px",
   marginLeft:"200px"
}
export default class App extends React.Component{
  constructor(){
      super(); //call parent constructor
       this.state={
         name:"",
         email:"",
         contact:""
       }
    }
    uniHandler=(e)=>{
         this.setState({[e.target.name]:e.target.value});
    }
   render(){
      return <>
```

```
     <input type='text' name='name' value={this.state.name}
style={btnCss}
      onChange={(e)=>this.uniHandler(e)} /> <br></br><br></br>
     <input type='text' name='email' value={this.state.email}
style={btnCss}
     onChange={(e)=>this.uniHandler(e)} /> <br></br><br></br>
    <input type='text' name='contact' value={this.state.contact}
style={btnCss}
    onChange={(e)=>this.uniHandler(e)} /> <br></br><br></br>
    <input type='button' name='s' value='Register' style={btnCss}/>
<br></br>
      </>
   }
}
```

Output



Conditional rending in React

Conditional rendering help us to display UI component using React based on condition
It is like as conditional operator

Syntax: condition ? option1:option2
Or
Condition && logic

Example: WAP program to implement following thing using React

Theme Toggle (Dark/Light Mode)

Description:

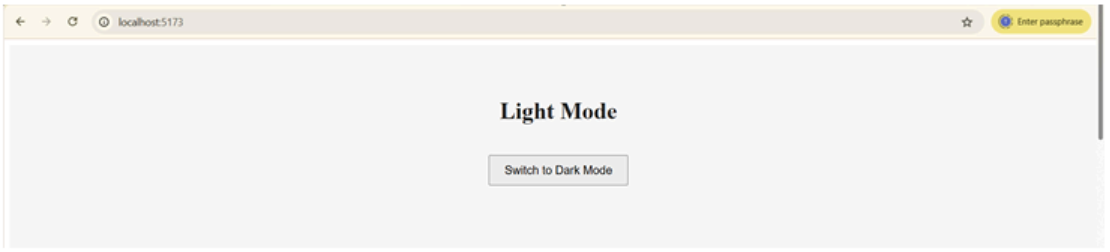
This component allows users to switch between dark and light themes. The theme state is managed using a class-based component. CSS classes are conditionally applied based on the current theme.

Features:

- Toggle button to switch between light and dark mode
- Uses `state` in class component to manage theme
- Applies appropriate CSS based on selected theme

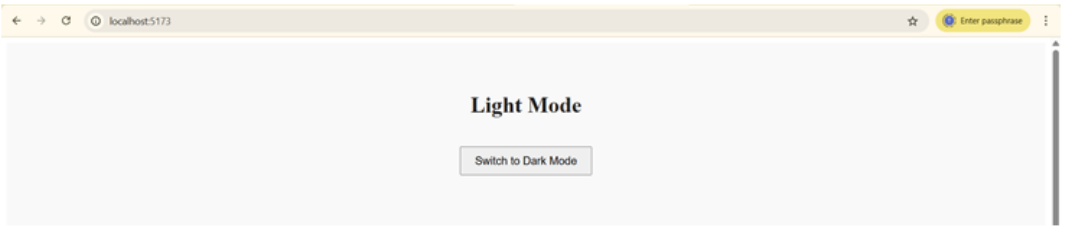
Output should look like as

When we run program then by default you have to following output



When user click on switch to dark Mode

Again when user click on switch to light mode then your output like as



Example with source code

Index.css

```css
.app{
  padding:50px;
  text-align: center;
  min-height: 100vh;
  transition: all 0.3s;
}
.app.dark{
  background-color: #1c1c1c;
  color:#f9f9f9
}
.app.light{
  background-color: #f9f9f9;
  color:#1c1c1c;
}
button{
  padding:10px 20px;
  font-size: 16px;
  margin-top: 20px;
  cursor: pointer;
}
```

App.jsx

```jsx
import React from "react";
import ReactDom from "react-dom";
class App extends React.Component
{   constructor(){
    super();
    this.state={
      isDarkMode:false
    }
   }
   toggleTheme=()=>{
    this.setState({isDarkMode:!this.state.isDarkMode});
   }
   render(){
    return <>
    <div className={this.state.isDarkMode?'app dark':'app light'}>
      <h1>{this.state.isDarkMode?'Dark Mode':'Light Mode'}</h1>
    <button onClick={this.toggleTheme}> Switch To
      {this.state.isDarkMode?'Dark Mode':'Light Mode'}</button>
    </div>
    </>
```

```
    }
}
export default App;
```

Example 3

Modal Popup using React Class Components

Description:

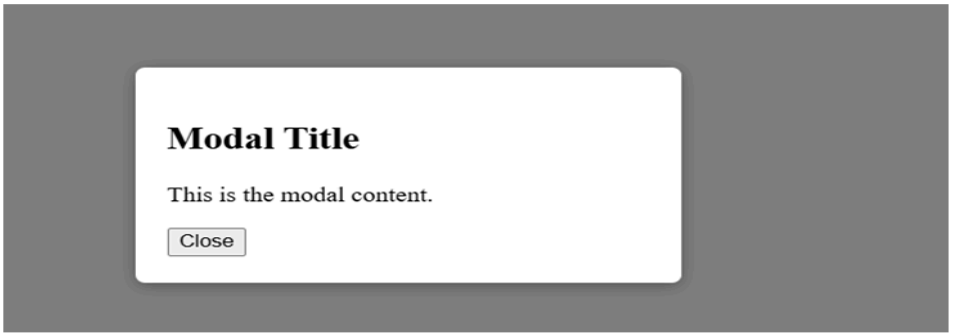
This React component demonstrates how to create and control a modal popup (dialog box) using class-based components. A modal is a UI element that overlays the main content and requires user interaction before returning to the main screen.

Features:

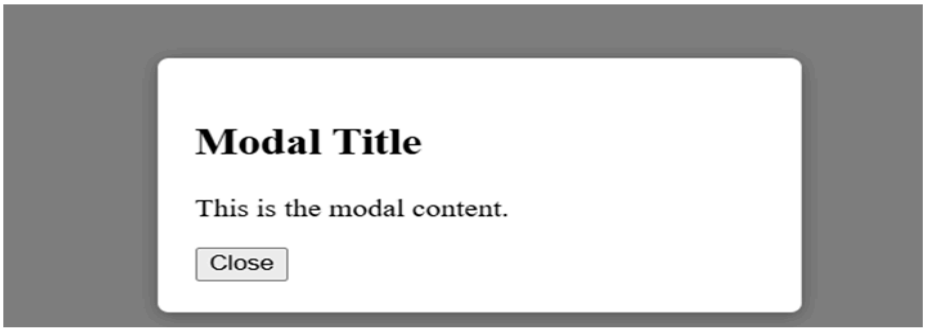
- **Modal visibility is managed using component state**
- **Modal opens/closes via buttons**
- **Simple, reusable modal component**
- **Uses plain CSS for styling and transition**

When we run the program then your output like as

After button click you modal should open like as



When user click on close the button then model should be close



Index.css

```css
.model-overlay{
  position: fixed;
  top:0;
  left:0;
  width: 100%;
  height: 100%;
  background-color: rgba(0,0,0,0.5);
  display: flex;
  justify-content: center;
  align-items: center;
}
.model-content{
  background-color: white;
  padding: 20px;
  border: 6px;
  min-width: 300px;
  box-shadow: 0 2px 10px rgba(0, 0,0,0.3);
}
```

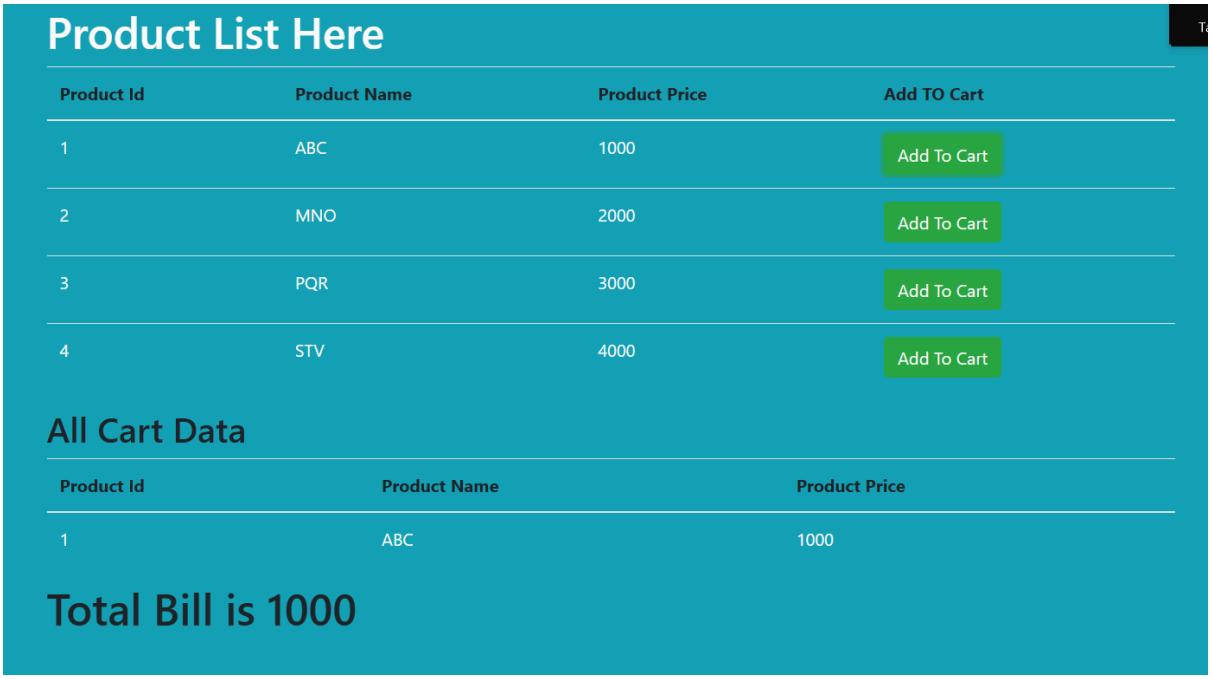
App.jsx

```jsx
import React from "react";
import ReactDom from "react-dom";

export default class App extends React.Component{
```

```
  constructor(){
    super();
    this.state={
       isModelOpen:false
    }
  }
  openModel=()=>{
    this.setState({isModelOpen:true});
  }
  closeModel=()=>{
    this.setState({isModelOpen:false})
  }
  render(){
    return<>
    <div>
       <button onClick={this.openModel}>Open Model</button>
        {this.state.isModelOpen && (<div className="model-overlay">
            <div className="model-content">
                <h2>Model Title</h2>
                <p>Model Content</p>
                <button onClick={this.closeModel}>Close</button>
            </div>
        </div>)}
        </div>
    </>
  }
}
```

Example : implement shopping Cart Demo using state variable

Note: you have to implement following types of demo

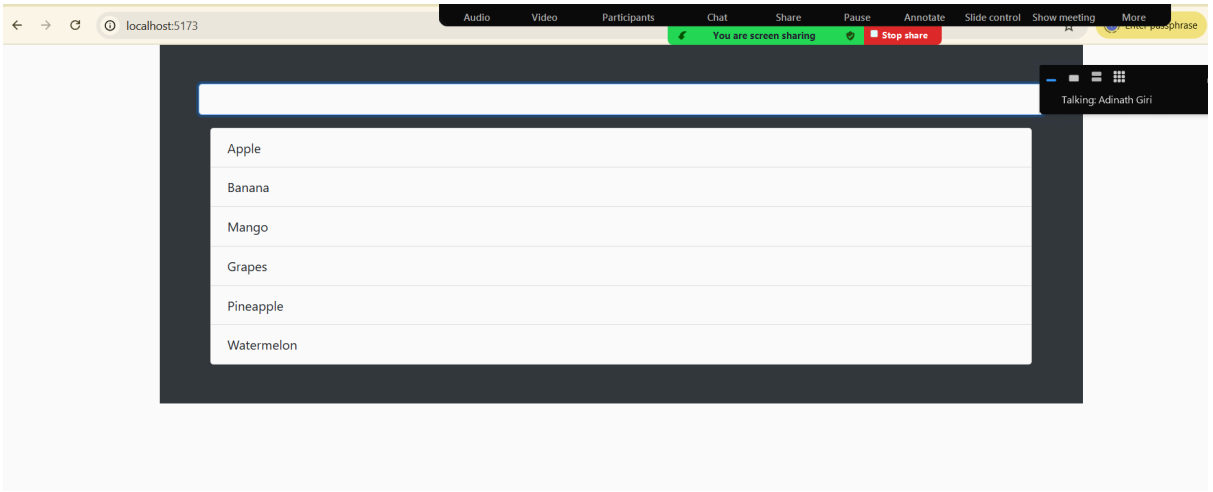



```
import React from "react";
import ReactDom from "react-dom";
export default class APP extends React.Component{

     constructor(){
       super();
       this.state={
           products:[{id:1,name:"ABC",price:1000},
                     {id:2,name:"MNO",price:2000},
                     {id:3,name:"PQR",price:3000},
                     {id:4,name:"STV",price:4000}
                   ],
              cart:[]
       }
     }
     addToCart=(product)=>{
      this.setState((prevState)=>({
        cart:[...prevState.cart,product]
      }));
     }
     getTotal=()=>{
       return this.state.cart.reduce((total,item)=>total+item.price,0);
     }
      render(){
       return  <>
        <div className="container bg-info p-5">
          <h1 className="text-white ">Product List Here</h1>
```

```
          <table className="table table-striped">
            <thead>
              <tr>
                <th>Product Id</th>
                <th>Product Name</th>
               <th>Product Price </th>
               <th>Add TO Cart</th>
              </tr>
            </thead>
          {

           this.state.products.map((product)=>(<tr
className="text-white">
            <td>{product.id}</td>
            <td>{product.name}</td>
            <td>{product.price}</td>
            <td>
              <button className="btn btn-success"
              onClick={()=>this.addToCart(product)} >Add To
Cart</button></td>

            </tr>))

          }
             </table>
             <h2>All Cart Data</h2>
             <table className="table table-striped">
            <thead>
              <tr>
                <th>Product Id</th>
                <th>Product Name</th>
               <th>Product Price </th>
               </tr>
            </thead>
             {
               this.state.cart.map((product)=>(<tr
className="text-white">
            <td>{product.id}</td>
            <td>{product.name}</td>
            <td>{product.price}</td>


            </tr>))
```

```
            }
            </table>
            <h1>Total Bill is {this.getTotal()}</h1>
        </div>

            </>
    }
}
```

Example : implement search filter using React class component




```
import React from "react";
import ReactDom from "react-dom";

export default class App extends React.Component{
  constructor(){
    super();
    this.state={
      query:"",
      items:[
            "Apple",
            "Banana",
            "Mango",
            "Grapes",
            "Pineapple",
            "Watermelon"
      ]
    }
  }
```

```
  handleChange=(e)=>{
    this.setState({query:e.target.value});

  }
  getFilterData=()=>{
    let {query,items}=this.state;
    return
items.filter(item=>item.toLowerCase().includes(query.toLowerCase()));//
app
  }
  render(){
    let filterData=this.getFilterData();
    return <>
      <div className="container bg-dark p-5">
         <div className="form-group">
        <input type='text' name='query' value={this.state.query}
        className="form-control" onChange={(e)=>this.handleChange(e)}/>
         </div>

        <div className="container">
        <ul className="list-group">
     {
      filterData.length>0 ?
      (filterData.map((item,index)=><li key={index}
className="list-group-item">{item}</li>))
           :(<li>Data not found</li>)
     }

        </ul>
        </div>
      </div>
    </>
  }
}
```

Props in React JS

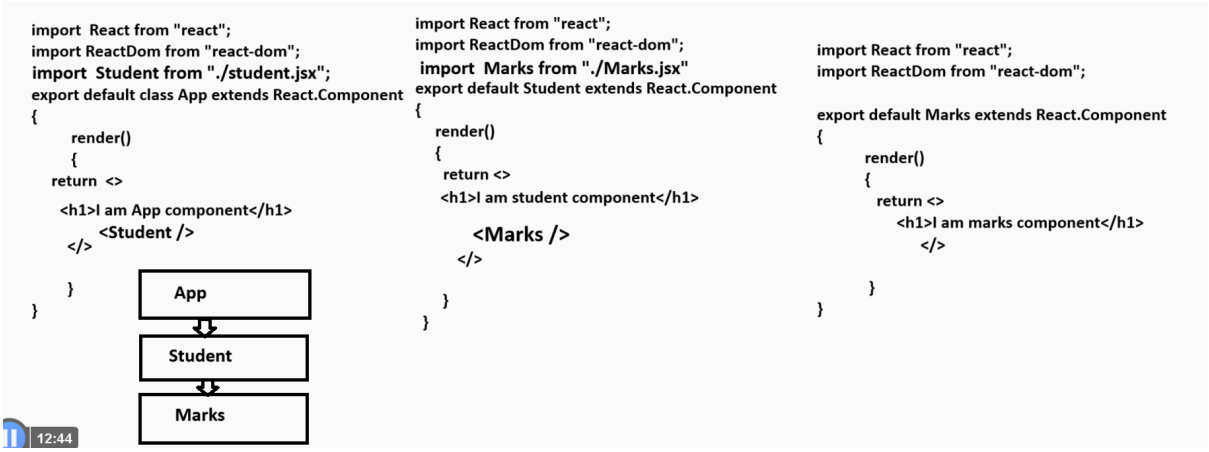
If we think about React props stands for properties are fundamental mechanism for for passing data from one component to another component
Means we can say property sender component known as parent component and received component known as child component

Q. What is the parent component in React?

A component use as JSX element to another component called as parent component

Q. What is the child component in React?

A component used by another component as a JSX element is called a child component.



If we think about above diagram we have App is main parent component and Student is child of App component and Marks is child of Student so if we want to share some data from App component student and student to marks then we can use props concept in React.

App.jsx

```jsx
import React from "react";
import ReactDom from "react-dom";
import Student from "./Student.jsx";
export default class App extends React.Component{

  render(){
    return <>
       <h1>I am App Component</h1>
       <Student/>
    </>
  }
}
```

Student.jsx

```jsx
import React from "react";
import ReactDom from "react-dom";
import Marks from "./Marks.jsx";
export default class Student extends React.Component{
```

```
    render(){
        return <><h1>I am student component </h1>
            <Marks />
        </>
    }
}
```

Marks.jsx

```
import React from "react";
import ReactDom from "react-dom";

export default class Marks extends React.Component{
    render(){
        return <><h1>I am Marks component</h1></>
    }
}
```

How to pass data from one component to another component using props

---

If we want to pass data from one component to another using props we have to use the following steps.

1. **Pass data to the component using key and value pairs.**

```
<Componentname  key="value"  />
```

Note: we pass data from parent component to child means we call child component in parent component

```
src > App.jsx > App
1  import React from "react";
2  import ReactDom from "react-dom";
3  import Student from "./Student.jsx";
4  export default class App extends React.Component{
5
6      render(){
7          return <>
8              <h1>I am App Component</h1>
9               <Student name="ABC" />
10         </>
11     }
12 }
```

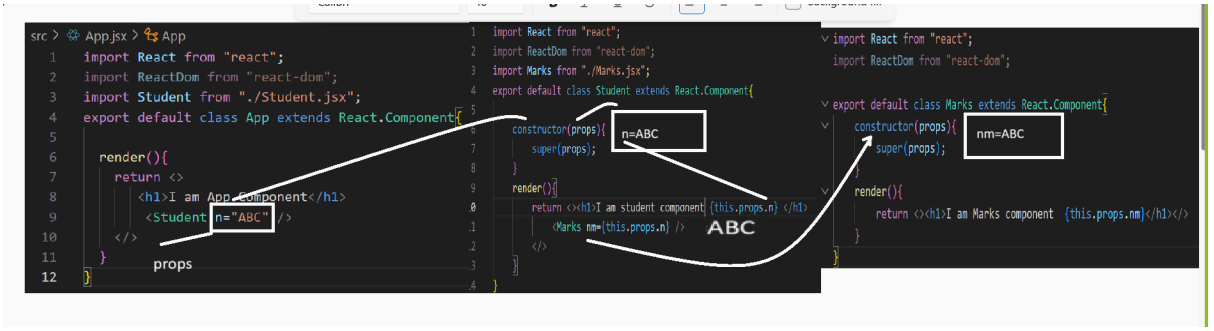
2. Create child component and define constructor in it and pass props as parameter to constructor and using super constructor pass to parent

```
import React from "react";
import ReactDom from "react-dom";
import Marks from "./Marks.jsx";
export default class Student extends React.Component{

    constructor(props){
        super(props);
    }
    render(){
        return <><h1>I am student component </h1>
            <Marks />
        </>
    }
}
```

Note: once we pass parameter as to props to parent we can read props in child component

If we want to read props in child component we have to use following syntax

Syntax: {props.name} or {props.key}



Example with source code

App.jsx

```
import React from "react";
import ReactDom from "react-dom";
import Student from "./Student.jsx";
export default class App extends React.Component{

  render(){
    return <>
       <h1>I am App Component</h1>
        <Student n="ABC" />
    </>
  }
}
```

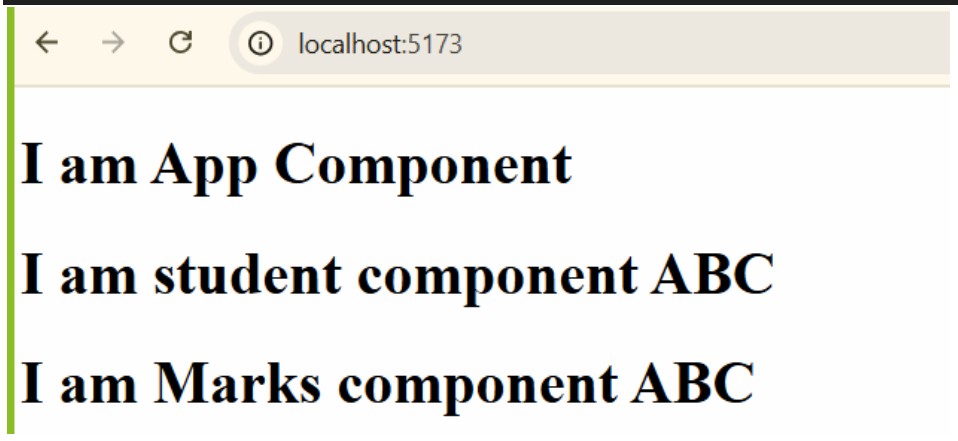
Student.jsx

```
import React from "react";
import ReactDom from "react-dom";
```

```
import Marks from "./Marks.jsx";
export default class Student extends React.Component{

    constructor(props){
        super(props);
    }
    render(){
        return <><h1>I am student component {this.props.n} </h1>
            <Marks nm={this.props.n} />
        </>
    }
}
```

Marks.jsx

```
import React from "react";
import ReactDom from "react-dom";

export default class Marks extends React.Component{
    constructor(props){
        super(props);
    }
    render(){
        return <><h1>I am Marks component
{this.props.nm}</h1></>
    }
}
```



Important points related with props

---

1. **Unidirectional data flow :** props facilitate a one way data flow meaning data is passed down from parent components to child components and child components cannot directly modify the props they receive.

2. **Read-only:** props are immutable within the child component means we can say the child component can read props from the parent but cannot modify them. Only the parent component that passed the props can alter the data.

3. **Similar to function arguments :** you can think of props as arguments to pass a function. Just as you pass argument to JavaScript function to customize behaviour

Q. What are the benefits of props?

---

1. **Reusability :** Components can be made generic and reused in different parts of application by passing varying data through props.

2. **Modularity :** props help us break down the UI into smaller , manageable components,promoting a modular and organized code base.

3. **Predictability :** the unidirectional data flow enforced by props makes it easier to understand how data moves through the application and predict component behaviour

Q. What is the meaning of prop drilling ?

Prop drilling means it is a process to share data between one component to another component called prop drilling or passing data from parent component to nested child component by passing props through intermediate component do not need to data themselves

Q. What is the difference between state variables and props?

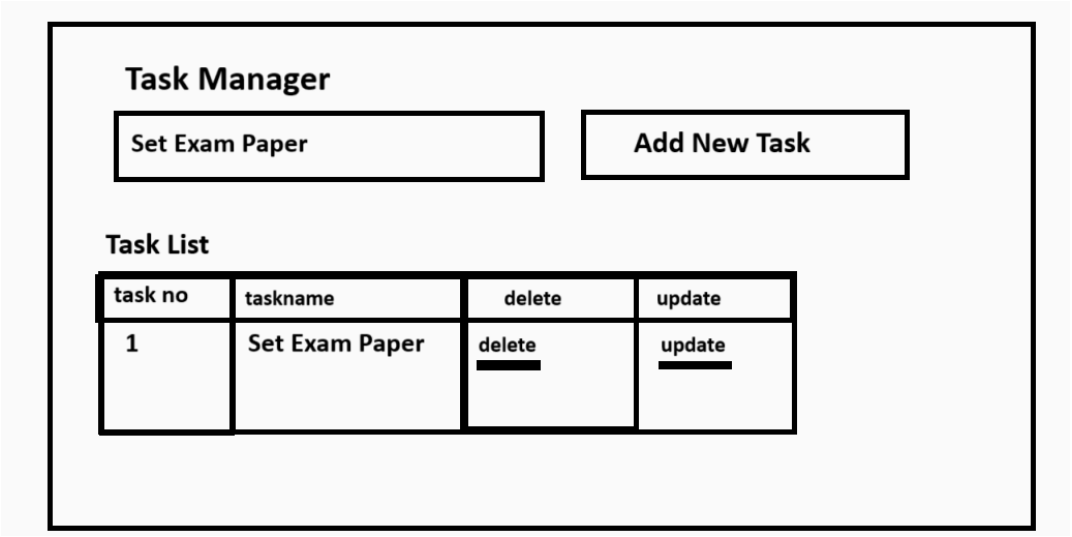
---

1. State variable can access only within same component and props can access outside of component
2. State variable can modify but props is read only in child component
3. State variable is used for manage the store within a same component and re-render component when we update state variable and when we update parent props then effect can found in child component also
4. State variable work instance variable in class but props work as function parameter
5. State variable not need to pass parent component using super but props need to pass as parameter to parent using super constructor

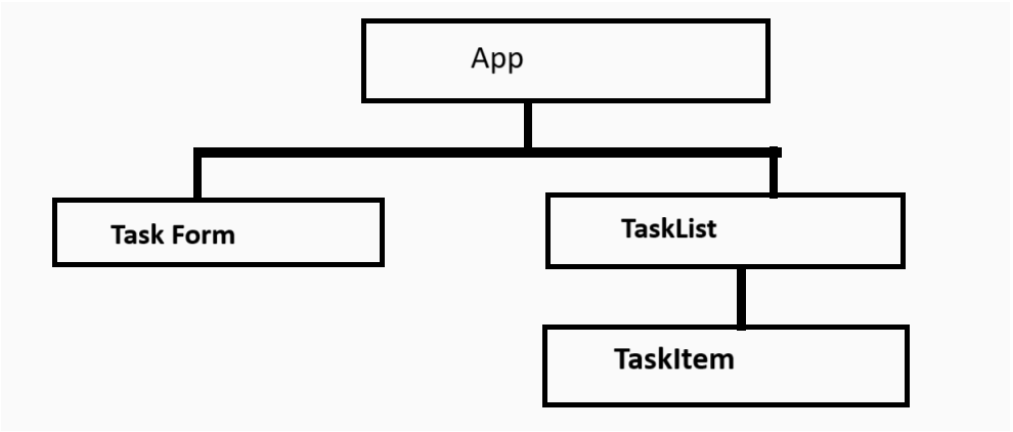
Task Management Application using React

Suppose we are working on a task management application and we want to perform the following operations.

1. **Add New Task**
2. **View Task**



Note: we want to use inheritance or parent and child relationship concept for achieve above task as well as we want to use state variable and props



Life cycle of React class component

Life cycle of Rect class component means it is a process that shows how components create and mount in DOM as well as update in DOM and remove from dom called as life cycle.

There are three stages in React class component life cycle

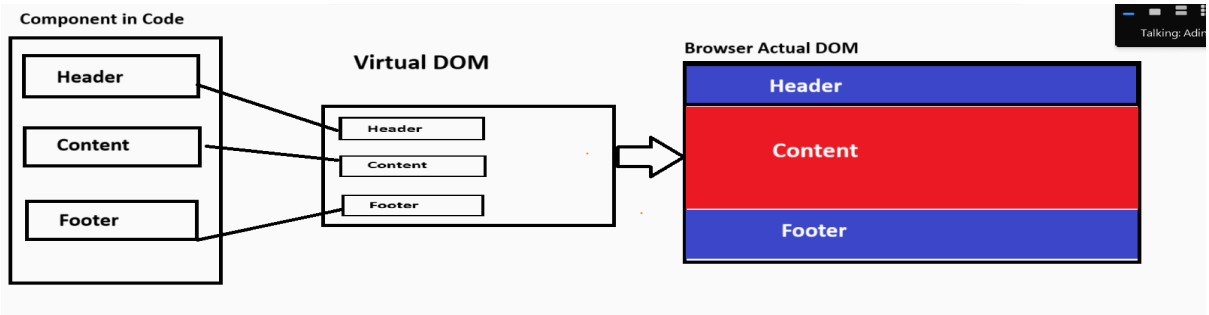
---

1. **Mounting phase :** Mounting phase means to decide how to create a class component and mount or place in a DOM tree called as mounting.

2. **Updating Phase :** updating phase means re-rendering process of component means when we change state variable or props from component and after that update that component in DOM called as updating phase.

3. **Unmounting Phase:** when we remove the component from DOM called as unmounting phase.

Mounting Phase

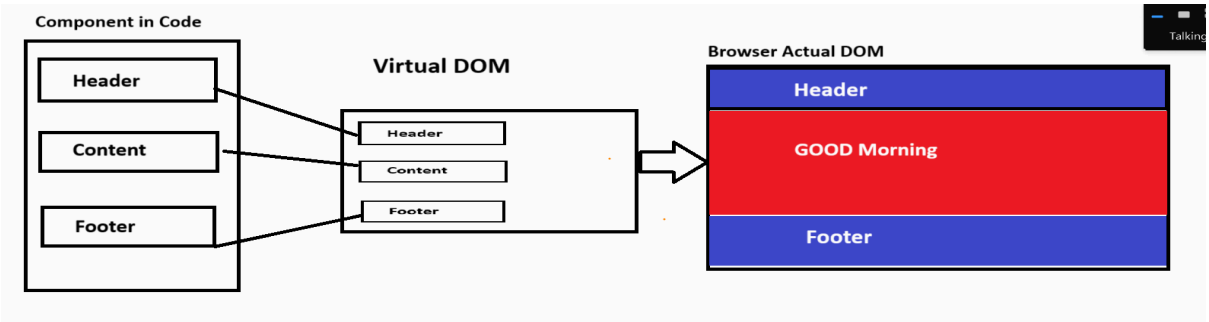


If think about mounting phase all source code component get stored in Virtual DOM and shift components from Virtual DOM to Actual DOM in browser

Updating Phase

When we change the state variable or props of a component then automatically the component gets updated or re-render.

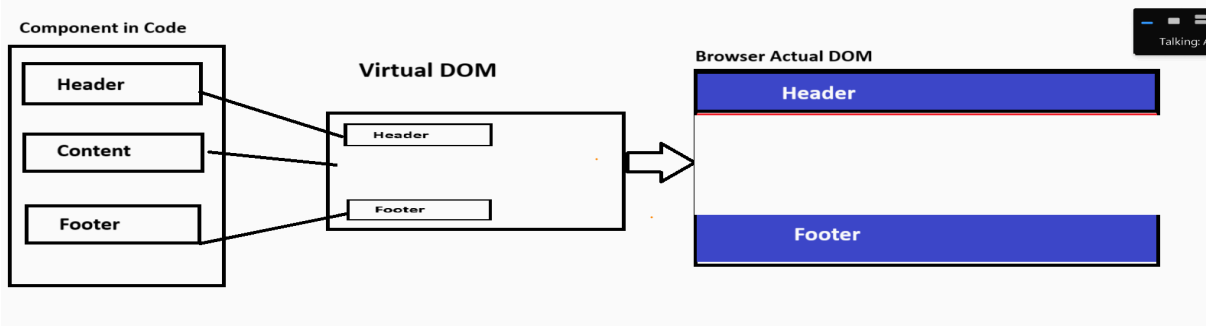
Suppose we change the content text from content component to good morning and suppose we store this text state variable means before update state variable value is content and after update state variable is good morning means change state of component so component update automatically in this case React not update browser DOM JUST update specified component from Virtual and Update only that in browser DOM means whose browser DOM not re-render so it may be improve performance page or user interface



Unmounting phase of React class component

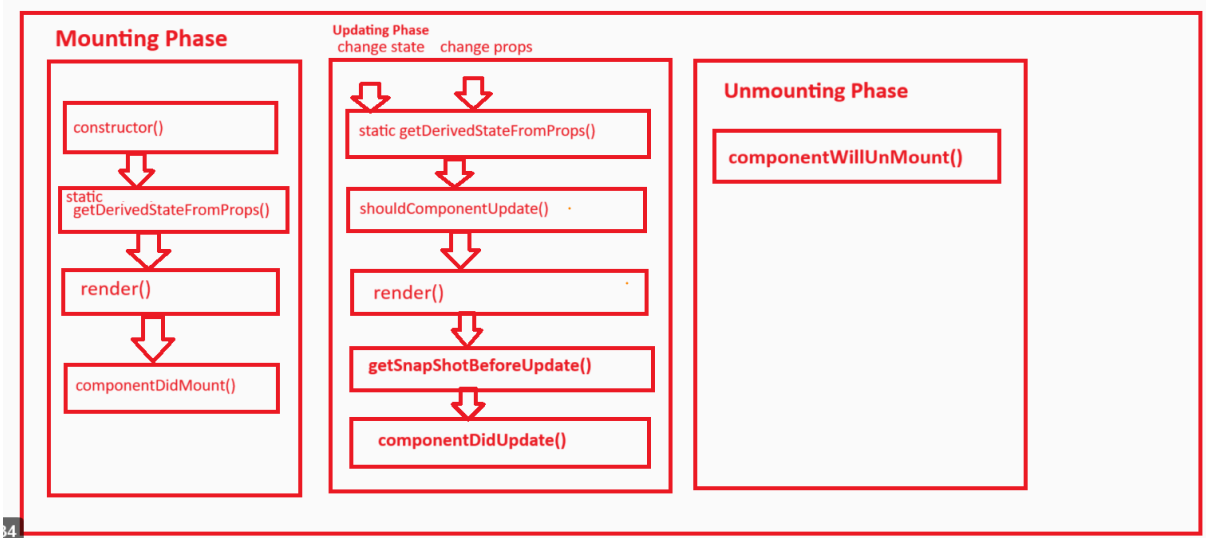
Remove the component from DOM called the unmounting phase.

Consider we want to remove the content component from DOM called as unmounting phase.



If you think about the life of the class , components in this process contain some inbuilt methods provided by React to us and every method has some purpose in every phase.

Methods involved in class component life cycle



Now we want to discuss about the mounting phase of class component

Method description in Mounting Phase

---

constructor(): constructor call when we use a component as JSX element and normally constructor is used for initialising the state variable or sending props to the parent component.

static getDerivedStateFromProps(props,state): this method call before render method and this method return updated state object or return null value.
This method is used in code in rare cases when the state variable is dependent on props changes and we cannot use this reference in this method.

render(): this method is mandatory in class components and returns JSX elements and calls every time when we update the state variable or props.

componentDidMount(): this method called once in life cycle means after first render. This is where AJAX Request and DOM or state update occur.
This method is used for integration purpose with other JavaScript Framework and any delay function like as setTimeout(),setInterval() as well as perform API calling or ajax call etc

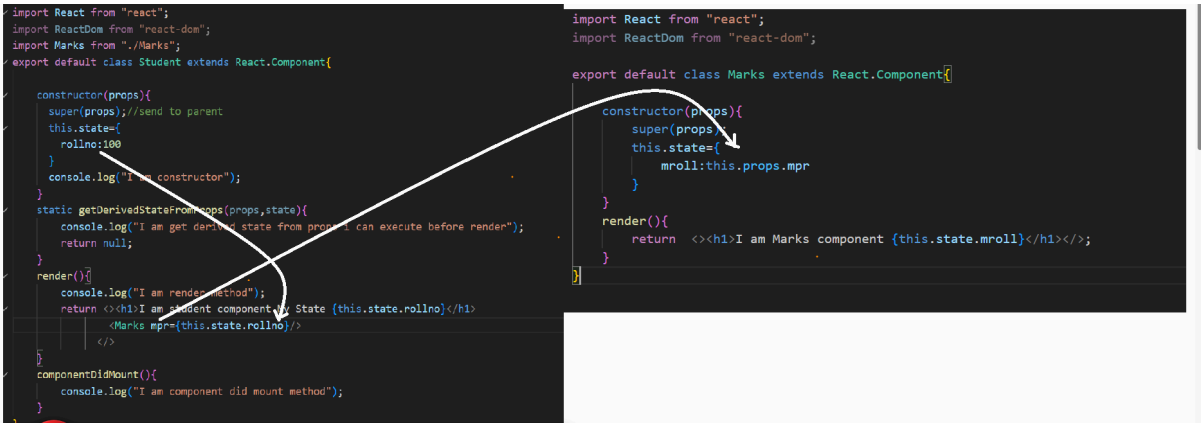
Example with source code

```
import React from "react";
import ReactDom from "react-dom";

export default class Student extends React.Component{
```

```
    constructor(props){
      super(props);//send to parent
      this.state={
        msg:"good Morning i am state variable"
      }
      console.log("I am constructor");
    }
    static getDerivedStateFromProps(props,state){
        console.log("I am get derived state from props i can execute
before render");
        return null;
    }
    render(){
        console.log("I am render method");
        return <><h1>I can return JSX element </h1>
                {this.state.msg}
          </>
    }
    componentDidMount(){
        console.log("I am component did mount method");
    }
}
```

Updating Phase of class component

We want to create one more component name as Marks in Project and we want to initialize state variable value send by props from parent component i.e Student
Means Student props is value of state variable in Marks component



We want to create one button in student component and when user click on button then we want to increase the value of state variable of student i.e rollno of student also we want to increase the state variable Marks component because state variable value of student is property value to Marks component and we initialize this value in state variable of Marks component

We have override getDerivedStateFromProps() in marks component and check if props and state variable value is not same then update props value to state variable and return object of state variable and after render() method get executed so we get updated value of state variable means here when we update the parent state and pass update state of parent as props to child then child state also get updated which is dependent props pass by parent.

Student.jsx

```
import React from "react";
import ReactDom from "react-dom";
import Marks from "./Marks";
export default class Student extends React.Component{

    constructor(props){
      super(props);//send to parent
      this.state={
        rollno:100
      }
      console.log("I am constructor");
    }
    static getDerivedStateFromProps(props,state){
        console.log("I am get derived state from props i can execute
before render");
        return null;
    }
    incCount=()=>{
      this.setState({rollno:this.state.rollno+1});
    }
    render(){
        console.log("I am render method");
        return <><h1>I am student component My State
{this.state.rollno}</h1>
                <Marks mpr={this.state.rollno}/>
                <br/>
                <button
                style={{width:"400px",height:"40px"}}
                 onClick={this.incCount}>Increase Value of Roll
No</button>
              </>
    }
    componentDidMount(){
        console.log("I am component did mount method");
    }
}
```

Marks.jsx

```
import React from "react";
import ReactDom from "react-dom";

export default class Marks extends React.Component{

    constructor(props){
        super(props);
        this.state={
            mroll:this.props.mpr
        }
    }
    static getDerivedStateFromProps(props,state){

        if(props.mpr!=state.mroll){ //101!=100
            return {mroll:props.mpr};
        }
        else{
            return null; //prop and state same hai
        }
    }
    render(){
        return  <>
        <h1>I am Marks component {this.state.mroll}
            </h1></>;
    }
}
```

shouldComponentUpdate() method

This method decides whether a component should update or not if component update returns true otherwise return false.

Example: we want to update Marks component whenever mroll value is less than equal 107 if mroll value is greater than 107 then Marks component should not update.

Marks.jsx

```
import React from "react";
import ReactDom from "react-dom";

export default class Marks extends React.Component{
```

```
    constructor(props){
        super(props);
        this.state={
            mroll:this.props.mpr
        }
    }
    static getDerivedStateFromProps(props,state){

        if(props.mpr!=state.mroll){ //101!=100
            return {mroll:props.mpr};
        }
        else{
            return null; //prop and state same hai
        }
    }
    shouldComponentUpdate(){
        if(this.state.mroll<107){
            return true;
        }
        else{
            return false;
        }
    }
    render(){
        return  <>
        <h1>I am Marks component {this.state.mroll}
            </h1></>;
    }

}
```

Note: if we think about above component Marks component update whenever state variable value is less than 107 after that not

getSnapShotBeforeUpdate(prevProps,prevState,snapshot): this method executes before componentDidUpdate() and holds the props and state variable value before updating it.

Marks.jsx

```
import React from "react";
import ReactDom from "react-dom";
```

```
export default class Marks extends React.Component{

    constructor(props){
        super(props);
        this.state={
            mroll:this.props.mpr
        }
    }
    static getDerivedStateFromProps(props,state){

        if(props.mpr!=state.mroll){ //101!=100
            return {mroll:props.mpr};
        }
        else{
            return null; //prop and state same hai
        }
    }
    shouldComponentUpdate(){
        if(this.state.mroll<107){
            return true;
        }
        else{
            return false;
        }
    }
    render(){
        return  <>
        <h1>I am Marks component {this.state.mroll}
            </h1></>;
    }
    getSnapshotBeforeUpdate(prevProps,prevState,n){
        console.log("Before update Props  "+prevProps.mpr);
        console.log("Before Update State "+prevState.mroll);

    }
}
```

componentDidUpdate(): this method call immediately after component update automatically
Normally this method is used in following cases

1. Fetching new data when props change
2. Reacting state update
3. Working with the DOM after change

Etc

Marks.jsx

```jsx
import React from "react";
import ReactDom from "react-dom";

export default class Marks extends React.Component{

    constructor(props){
        super(props);
        this.state={
            mroll:this.props.mpr
        }
    }
    static getDerivedStateFromProps(props,state){

        if(props.mpr!=state.mroll){ //101!=100
            return {mroll:props.mpr};
        }
        else{
            return null; //prop and state same hai
        }
    }
    shouldComponentUpdate(){
        if(this.state.mroll<107){
            return true;
        }
        else{
            return false;
        }
    }
    render(){
        return  <>
        <h1>I am Marks component {this.state.mroll}
            </h1></>;
    }
    getSnapshotBeforeUpdate(prevProps,prevState,n){
        console.log("Before update Props  "+prevProps.mpr);
        console.log("Before Update State "+prevState.mroll);

    }
    componentDidUpdate(){
        console.log("component updated successfully...");
    }
```

```
}
```

Unmounting phase of component

When we remove the component from DOM called as unmounting phase when we remove the component from dom then React call one inbuilt method known as componentWillUnmount() automatically

Example with source code

Main.jsx

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import Test from "./Test.jsx";
import App from './App.jsx'

let root=document.getElementById('root1');

let comp=createRoot(root);

comp.render(<Test />)

let removeComponent=()=>{
  console.log("remove method call");
    comp.unmount();
}
createRoot(document.getElementById('root')).render(<>
   <button onClick={removeComponent}>Remove Component</button>
</>);
```

Test.jsx

```
import React from "react";
import ReactDom from "react-dom";

export default class Test extends React.Component{
```

```
    render(){
        return<><h1>I am Test component</h1></>
    }
    componentWillUnmount(){
        console.log("component unmounted");
    }
}
```

How to perform routing using react

We can perform routing using React we have two ways

1. Using <a > href
2. **Using BrowserRouter :**

Now we want to discuss about <BrowserRouter> in React

BrowserRouter: BrowserRouter is a router component that acts as the foundation of client side routing in React applications. It uses the HTML5 History API to keep the UI in Sync with the URL leading to clean user-friendly URL without hash symbols

BrowserRouter manages the all routing context allowing other components like Link and Route to function correctly within the application.

It enables smooth single-page application (SPA) navigation without full page reloads.

How to perform Routing in React

Steps to perform routing in React

---

1. **Install the react-router-dom library in application**

```
D:\july 2024\reactprojects\routingapp>npm install react-router-dom

added 48 packages, and audited 202 packages in 14s

33 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

2. **Create different component for routing**
   Example: we want to create three components i.e three pages like as home,about,contact etc

Example with source code

Index.css

```css
ul{
  width:100%;
  background-color: black;
  padding:20px;
}
ul li{
  display:inline-block;
  padding:10px;
}

.navLink{
  text-decoration: none;
  color:white
}
```

Main.jsx

```jsx
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import "./index.css"
import App from './App.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App />
  </StrictMode>,
)
```

App.jsx

```jsx
import React from "react";
import ReactDom from "react-dom";
import Home from "./Home";
import About from "./About";
import Contact from "./Contact";
import { BrowserRouter, Routes,Route, NavLink, Link } from
"react-router-dom";
```

```
export default class App extends React.Component{

  render(){
    return <>
      <BrowserRouter>
       <NavLink className="navLink">
             <ul>
              <li><Link to="/" className="navLink">Home</Link>  </li>
              <li><Link to="/about" className="navLink">About</Link>
</li>
              <li><Link to="/contact"
className="navLink">Contact</Link>  </li>
              </ul>
         </NavLink>
         <Routes>
           <Route path="/" element={<Home/>} />
            <Route path="/about" element={<About/>} />
            <Route path="/contact" element={<Contact/>} />
         </Routes>

      </BrowserRouter>

    </>
  }
}
```

Home.jsx

```
import React from "react";
import ReactDom from "react-dom";

export default class Home extends React.Component{

    render(){
        return <h1>I am Home Component</h1>
    }
}
```

About.jsx

```
import React from "react";
```

```
import ReactDom from "react-dom";

export default class About extends React.Component{

    render(){
        return <h1>I am about component</h1>
    }
}
```

Contact.jsx

```
import React from "react";
import ReactDom from "react-dom";

export default class Contact extends React.Component{
    render(){
        return <h1>I am contact component </h1>
    }
}
```

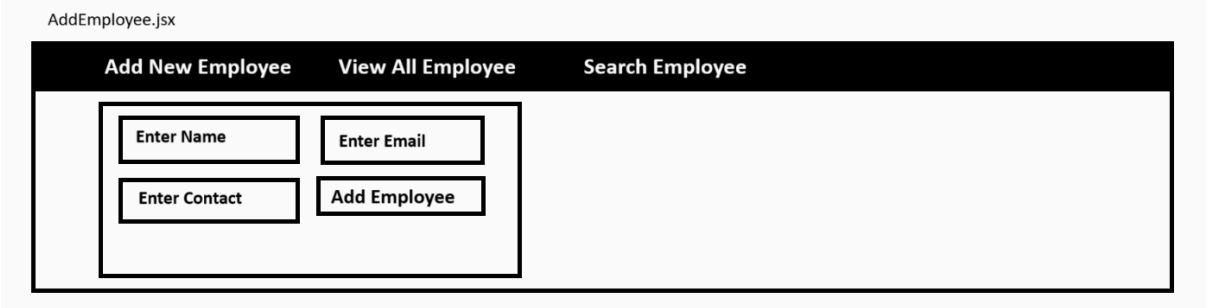
Example: we want to create CRUD Application using React + Node

1. **Add New Employee**
2. **View Employees**
3. **Delete Employee**
4. **Update Employee**
5. **Search Employee**

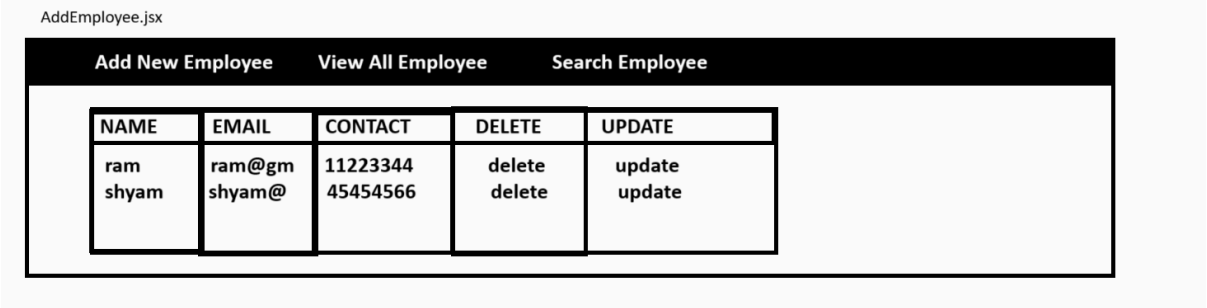
Home page

App.jsx

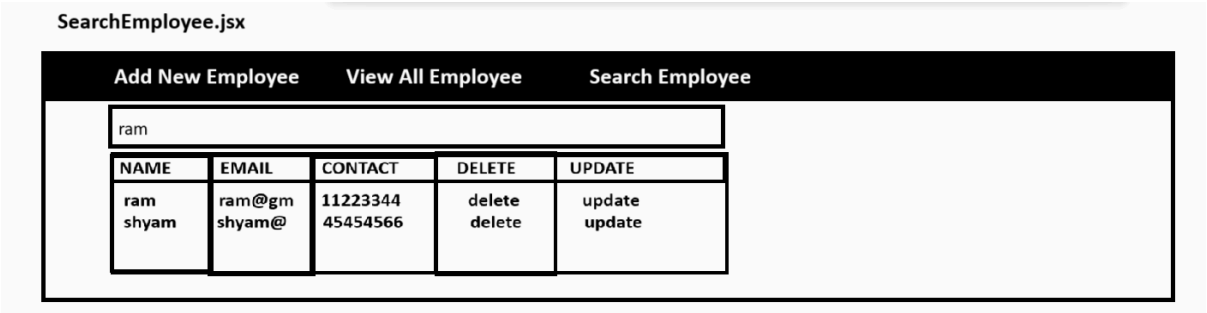
AddEmployee.jsx



ViewEmployee.jsx



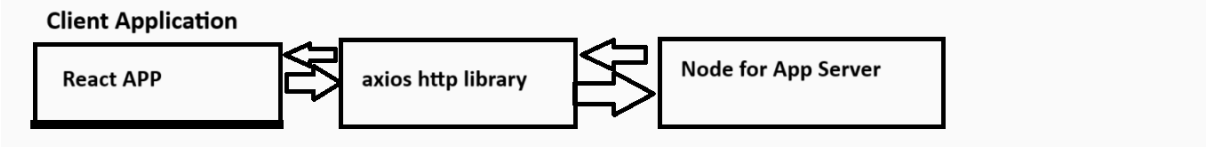
SearchEmployee.jsx



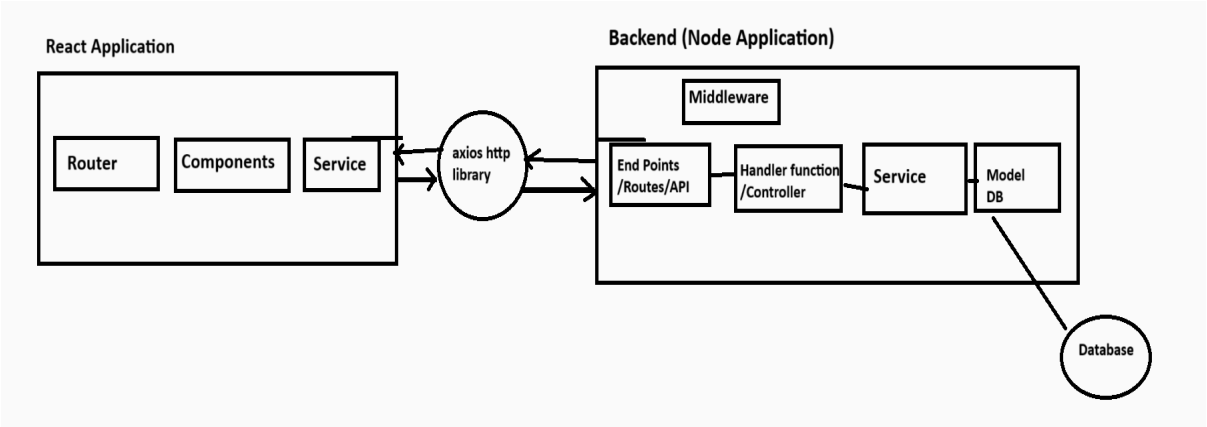
Tech Stack

1. **React + bootstrap + CSS - built user interface**
2. **Backend for creating REST API - Node**
3. **Axios library -** React is not support to http protocol directly and if we want to call REST API in react we need to support of http methods so here we use axios library which help us call REST API using http methods

Following diagram shows the structure of our application



Now we want to discuss about in depth code structure



Note: When we call the Node API or any server side API in React we need to manage the CORS Policy at server side or destination server

Q. What is CORS?

CORS stands for Cross Origin Resource Sharing. It is a security mechanism implemented by web browsers that allow web pages to request resources from different domains than the one that served pages.
It's crucial for enabling modern web applications to interact with APIs and other resources hosted on different servers,while also providing a layer of security against unauthorized access

Q. What is the security policy of CORS?

Browsers enforce a same-domain policy that restricts web pages from making requests to a different domain than the one they originated from.
This security feature measures to prevent malicious websites from accessing sensitive data on other sites.

CORS as an Exception

CORS provides a way for browsers to explicitly allow cross origin requests to enable control access to their resource from a specific domain.
CORS header send both request and response for verify request is allowed or not

Why use CORS?

Modern web applications often need to access data or services from different domains or server such as API hosted on separate server and want to access from another server then CORS implementation is essential

How to implement CORS using node

If we want to implement CORS using node we have to use a third middle ware name as cors and we must install it in the project.

Steps to use CORS in node application

---

1. **Install cors in application.**

```
D:\july 2024\reactprojects\nodeapp>npm install cors

added 2 packages, and audited 110 packages in 5s

20 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

D:\july 2024\reactprojects\nodeapp>
```

2. **We have to import cors and set it as application middleware.**

Example with source code

```
const bodyParser = require("body-parser");
let express=require("express");
let route=require("./routes/route.js");
let cors=require("cors");

let app =express();
app.use(bodyParser.urlencoded({extended:true}));
app.use(bodyParser.json());
app.use(cors());

app.use("/",route);
module.exports=app;
```

Configuration options using cors

```
const corsOptions={

origin: "origin server ip with port",
methods:['GET','POST'],
allowHeaders:['Content-Type','Authorization'],
credentials:true
};
app.use(cors(corsOptions));
```

Example with source code

```
const bodyParser = require("body-parser");
let express=require("express");
let route=require("./routes/route.js");
let cors=require("cors");

let app =express();
app.use(bodyParser.urlencoded({extended:true}));
app.use(bodyParser.json());
let corsOptions={
  origin:'http://localhost:5173',
  methods:['GET','POST']
};
app.use(cors(corsOptions));
app.use("/",route);
module.exports=app;
```

Function components in React

Function component are JavaScript functions that return a JSX element ,which is a template used to define a component structure
When returning a JSX element using a function of JavaScript and rendering on a web page called as function components.

Feature of function components

---

1. **Stateless (before hooks):** Originally , function components were stateless and used only for rendering UI based on props.
2. **Simple syntax:** They are defined as JavaScript functions leading to cleaner and more readable code.
3. **Return JSX:** function component can return JSX without overriding render() method

4. **No, this keyword:** unlike class component function components do not have this context or this keyword
5. **Hooks: with hooks** , functional components can manage state and side effects ,making them just as powerful class components.

How to create function component using react

If we want to create a function component using React we have two ways.

1. **Using normal function :**

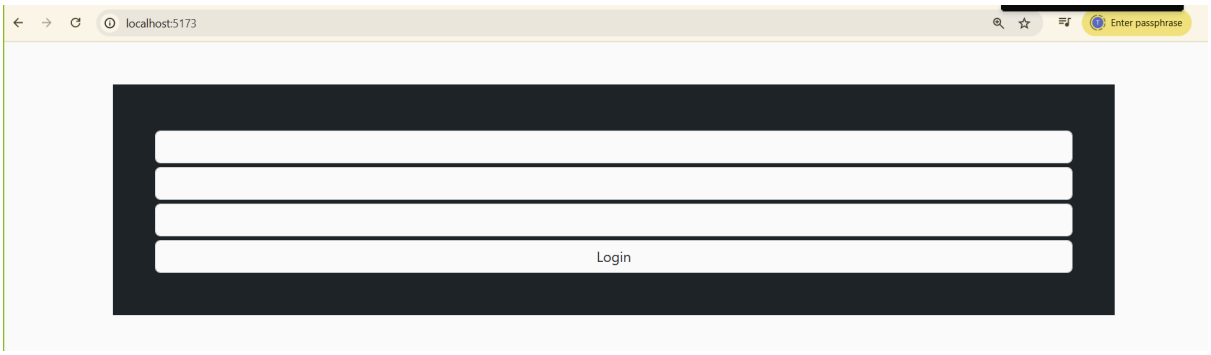
```
Syntax:
function  function name(props)
{
  return  <> JSX elements </>
}
export default function name;
```

2. **Using arrow function :**

```
Syntax:

let functionname=(props)=>{
  write here your logics
}
export default function name;
```

Example: we want to design login form using a function component

---



Login.jsx

```
import React from "react";
import ReactDom from "react-dom";
import "bootstrap/dist/css/bootstrap.min.css";
let Login=()=>{

    return <>
          <div className="container bg-dark mt-5 p-5">
```

```
          <div className="form-group mt-1">
            <input type='text' name='name' value=''
className="form-control" />
          </div>
            <div className="form-group  mt-1">
            <input type='text' name='email' value=''
className="form-control" />
          </div>
            <div className="form-group  mt-1">
            <input type='text' name='contact' value=''
className="form-control" />
          </div>
            <div className="form-group  mt-1">
            <input type='submit' name='s' value='Login'
className="form-control" />
          </div>
          </div>
        </>
}
export default Login;
```

App.jsx

```
import React from "react";
import ReactDom from "react-dom";
import Login from "./Login";
let App=()=>{

   return <><Login/></>
}
export default App;
```

Q. What is the difference between function component and class component in react?

Function component	Class component
It is simple JavaScript with arrow or without arrow function without render() method and return JSX elem	It is normal java class which acquire the React.Component class properties and override render() method and return JSX element
By default it is stateless component means we cannot use state variable in function component for that we required to us hooks	By default it is stateful component means we can declare state variable directly in class component not required any hooks

Function component can use hooks for manage the state variables and side effect of components	Class component cannot use hooks and can manage the state variable using java script object and side effect using life cycle methods
Function component cannot use this reference or keyword	Class component can use this keyword for manage state variable or props
Function component slightly faster because abstraction	Class component slightly slower because of internal logics

Q. What are the advantages and disadvantages of a function component?

Advantages	**Disadvantages**
Simpler to write and understand	No built in lifecycle methods (without hooks)
Better performance without extra overhead	May cause performance issue with frequent re-rendering
Support react hooks for managing state and effects	Manage complex state can be harder

Q. How to send props from one component to another?

App.jsx

```
import React from "react";
import ReactDom from "react-dom";
import A from "./A.jsx";
let App=()=>{

   return <><A/></>
}
export default App;
```

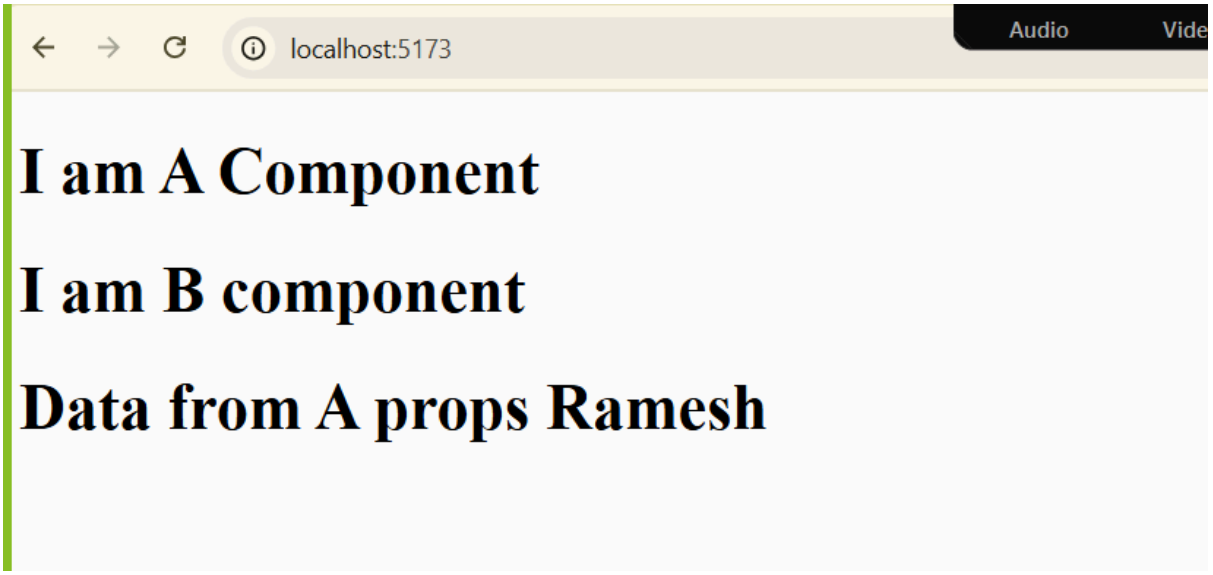
A.jsx

```
import React from "react";
import ReactDom from "react-dom";
import B from "./B.jsx";
let A=()=>{

    return <><h1>I am A Component</h1>
```

```
        <B name="Ramesh" />
     </>
}
export default A;
```

B.jsx

```
import React from "react";
import ReactDom from "react-dom";

let B=(props)=>{
    return <><h1>I am B component</h1>
         <h1>Data from A props {props.name}</h1>
    </>
}
export default B;
```



Hooks in react

Hooks is feature in function component which is used for to manage the state variable and life cycle methods of class component using function component

If we want to work with hooks in the function component we have following rules.

1. Hooks can only be called inside React function component
2. Hooks can be called at the top level of a component
3. Hooks cannot be conditional.

Types of hooks in React

1. useState
2. useEffect
3. useContext
4. useRef
5. useReducer
6. useCallBack
7. useMemo
8. CustomHook

Etc

Now we want to discuss about useState hook

The React useState hooks allows us to track state in a function component or manage state variable using a function component
State generally refers to data or properties that need to be tracked in application.

useState hook enables function components to hold and update dynamic data over time ,making them capable of remembering information across re-renders.

Syntax of useState hook

```
Syntax:  let  [variablename,setVariablename]=useState(initialvalue);
```

When we call useState takes an initial state value as argument and return an array contain two elements

```
Syntax:  let  [variablename,setVariablename]=useState(initialvalue);

  useState() hook 0th position means variable name and 1st position means function name

  variable is used for hold data and function is used for update data of using useState.
```

How to use useState hook in function component

1. import useState hook from react
2. Declare useState hook with variable and setter function
3. Modify the hook data using setter method

Function component

```
import React,{useState} from "react";
import ReactDom from "react-dom";
import "bootstrap/dist/css/bootstrap.min.css";
let Login=()=>{
  let[count,setCount]=useState(0);
  let incCount=()=>{
     setCount(count+1);
```

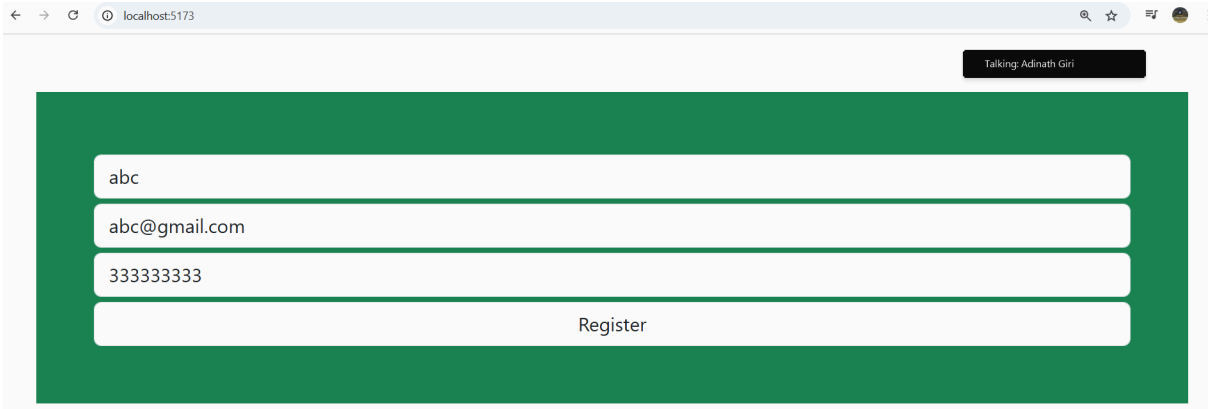
```
  }
    return <>
          <div className="container bg-dark mt-5 p-5">
           <div className="form-group  mt-1">
            <label className="form-control">{count}</label>
          </div>
            <div className="form-group  mt-1">
            <input type='button' name='s' value='Login'
className="form-control" onClick={incCount}/>
          </div>
          </div>
        </>
}
export default Login;
```

How to manage form using useState()

```
import React,{useState} from "react";
import ReactDom from "react-dom";
import "bootstrap/dist/css/bootstrap.min.css";
let Login=()=>{
   let[reg,setReg]=useState({
      name:"",
      email:"",
      contact:""
   });

   let handleForm=(e)=>{
      setReg({[e.target.name]:e.target.value});
     /* setReg(prevReg=>{
        return {...prevReg,[e.target.name]:e.target.value}
      });*/
   }
    return <>
          <div className="container bg-success mt-5 p-5">

             <div  className="form-group mt-1">
                 <input type='text' name='name' value={reg.name}
placeholder="Enter name "
                  className="form-control"
onChange={(e)=>handleForm(e)}/>
             </div>
```

```
            <div  className="form-group mt-1">
                <input type='text' name='email' value={reg.email}
placeholder="Enter Email "
                classNameName="form-control"
onChange={(e)=>handleForm(e)}/>
            </div>

            <div  className="form-group mt-1">
                <input type='text' name='contact' value={reg.contact}
placeholder="Enter Contact "
                classNameName="form-control"
onChange={(e)=>handleForm(e)}/>
            </div>

            <div  className="form-group mt-1">
                <input type='submit' name='s' value='Register'
classNameName="form-control"/>
            </div>
          </div>
        </>
}
export default Login;
```

Output



useEffect() hooks

The useEffect() hooks allow you to perform side effects in your components.
Some example of side effects are fetching data, directly updating DOM and timers etc
useEffect() accepts two arguments , the second argument is optional

Syntax: useEffect(call back function,<dependency>)

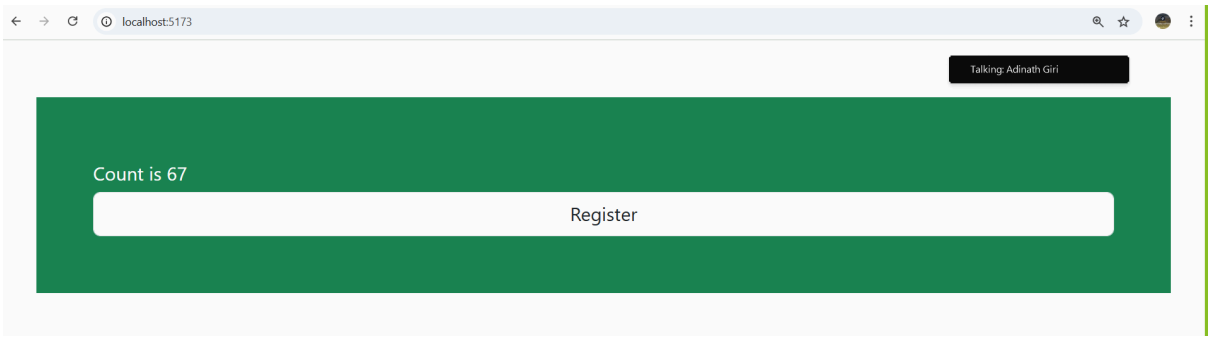
Note:useEffect() runs on every render means when update state variable or props then component get rendered and useEffect() get execute but we can control every time update of useEffect() by using dependency

useEffect() hook can manage the componentDidMount, componentDidUpdate() method functionality as well as componentWillUnmount()

Example of useEffect()

```
import React,{useEffect, useState} from "react";
import ReactDom from "react-dom";
import "bootstrap/dist/css/bootstrap.min.css";
let Login=()=>{
     let [count,setCount]=useState(0);
     useEffect(()=>{
      setTimeout(()=>{
        setCount(count+1);
      },1000);
     });
    return <>
          <div className="container bg-success mt-5 p-5">
             <div  className="form-group mt-1 text-white">
            <label className="form-group">Count is {count}</label>
             </div>
             <div  className="form-group mt-1">
                 <input type='submit' name='s' value='Register'
className="form-control"/>
             </div>
          </div>
        </>
}
export default Login;
```

Output



EXample with source code

```
import React,{useEffect, useState} from "react";
import ReactDom from "react-dom";
import "bootstrap/dist/css/bootstrap.min.css";
let Login=()=>{
     let [count,setCount]=useState(0);
     let[value,setValue]=useState(0);
 //componentDidMount(),0
      useEffect(()=>{
        setValue(value*count);
     },[count]);
     let incCount=()=>{
         setCount(count+1);
     }
    return <>
          <div className="container bg-success mt-5 p-5">
             <div  className="form-group mt-1 text-white">
            <label className="form-group">Count is {count}</label>
<br/>
            <label className="form-group mt-1">Value is {value}</label>
             </div>
             <div  className="form-group mt-1">
          <input type='submit' name='s' value='Increase counter'
className="form-control" onClick={incCount}/>
             </div>
          </div>
        </>
}
export default Login;
```

useReducer() : the useReducer() hook is similar to the useState hook.
It allows for custom state logic.
If you find yourself keeping track of multiple pieces of state that rely on complex logic then useReucer() is beneficial.

Syntax of reducer hook

Syntax: useReducer(<Reducer>,<initialState>);

The reducer function contains your custom state logic and initialState can be a simple value but generally will contain an object.

The useReducer() hook returns the current state and dispatch method

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

```
import ReactDom from "react-dom";
 let countReducer=(state,action)=>{
       switch(action.type){
          case "INC":
            return {count:state.count+1}
          case "DEC":
            return {count:state.count-1};
            default:
              return state;
       }
  }
let App=()=>{
  let [state,dispatch]=useReducer(countReducer,{count:0});
  return<>
   <h1>Counter is {state.count}</h1>
   <input type='button' name='s' value='Increment'
style={{width:"400px",height:"40px" ,marginRight:"20px"}}
   onClick={()=>dispatch({type:"INC"})}/>
     <input type='button' name='s' value='Increment'
style={{width:"400px",height:"40px"}}
     onClick={()=>dispatch({type:"DEC"})}/>
  </>
}
export default App;
```

Example for todo list

```
import React, { useReducer, useState} from "react";
import ReactDom from "react-dom";
let css={
  width:"400px",
  height:"40px",
  marginRight:"10px"
}                  //{type:"add",data:text,id:1}
let reducer=(state,action)=>{
  switch(action.type){
    case "add":
      let val=action.id+1;
      if(state.length!=0){
         val=state[state.length-1].srno+1;
      }
    return [...state,{msg:action.text,srno:val}];
    case "remove":
```

```
      return state.filter((item)=>item.srno!=action.id);

    case "clear":
      return [];
    default:
      return state;
  }
}

let App=()=>{
  let[text,setText]=useState("");
  let[todods,dispatch]=useReducer(reducer,[]);
  let addTodo=()=>{
      dispatch({type:"add",text,id:0})
  }
  let removeTodo=(did)=>{

    dispatch({type:"remove",id:did})
  }
  let clearTodo=()=>{
     dispatch({type:"clear"})
  }
  return <>
     <input type='text'  name='s' value={text} style={css}
onChange={(e)=>setText(e.target.value)}/>
     <input type='button' name='s1' value='Add' style={css}
onClick={addTodo}/>
     <input type='button' name='s1' value='Clear' style={css}
onClick={clearTodo}/>

   <br></br><br></br>
    {
      <table style={{width:"80%",borderStyle:"solid"}}>
        <tr
style={{textAlign:"center"}}><th>SRNO</th><th>Message</th><th>REMOVE</t
h></tr>
           {
            todods.map(item=>(<tr
style={{textAlign:"center"}}><td>{item.srno}</td><td>{item.msg}</td>
            <td><input type='button' name='n' value='X'
onClick={(e)=>removeTodo(item.srno)} /></td></tr>))
           }
```

```
        </table>
      }
    </>
}
export default App;
```

useMemo hooks

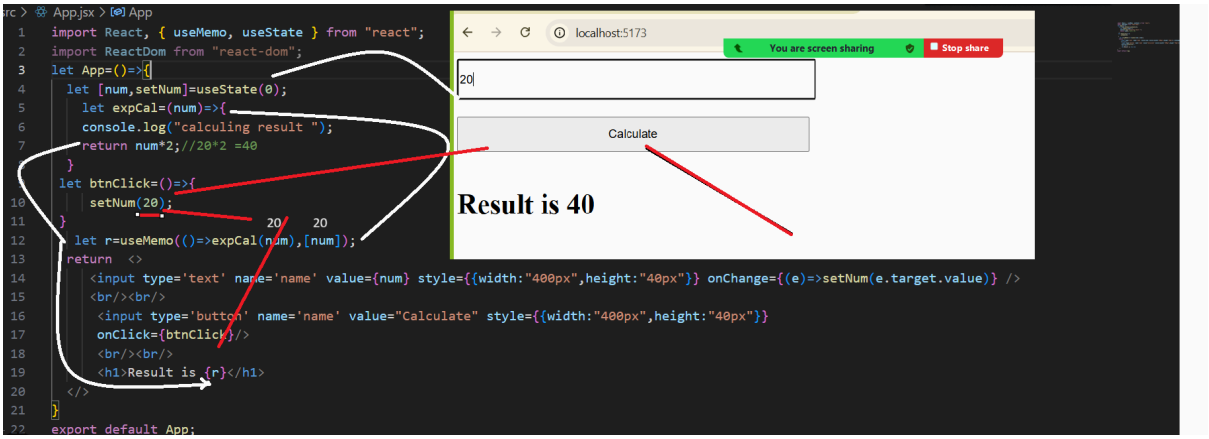
The useMemo hooks in React is a powerful performance optimization tool. It memorises the result of a function and recalculates the value only when a dependency changes. This helps prevent unnecessary re-renders and computations where a calculation is expensive or when you are working with a large dataset.

Syntax of memo hooks

Syntax of memo hooks

```
let memoriszevalue=useMemo(()=>{
    call back function
     return value;
  },[dependencies ]);
```

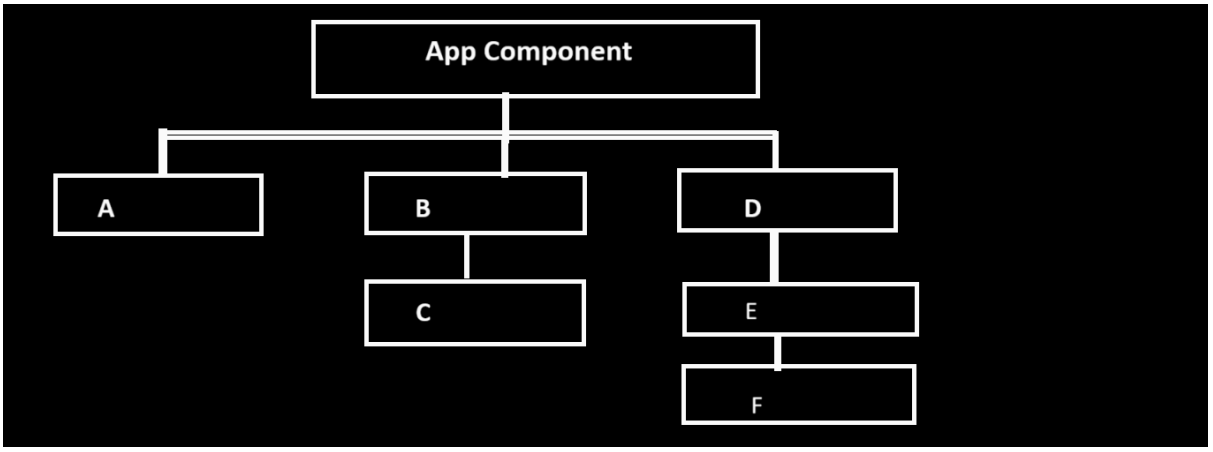
1. **memorizedValue:** the value returned by the function inside useMemo , which will be recomputed only when the value in dependencies array change
2. **Dependencies:** an array of values that useMemo() listen to when any of the value in this array change , the function is recomputed



Context Hook?

Q. What is context?

Context provide a ways to pass data from parent to child through component without prop drilling means using context we can skip intermediate parent to send props means we can directly send props to directly targeted child



If we think about above diagram ,we have three level of components and suppose component App component want to send data to F component so we must be send data to D and E component and after that means D and E not required data send by App but as props drilling rules we required to send via D and E component
So if we want to solve this problem we have to use context book using this concept we can avoid prop drilling problem

Now we want to create an example to send data from App Component to F component .

Steps to work with Context API

If we want to create context we have following steps

1. **Create React context**
   Syntax: let ref= React.createContext();
2. **Provide value to component for that we need to write provider**

```
export  let variablename=React.createContext();

<variablename.Provider value={value}>
    <ComponentName />
</variablename.Provider>>
```

Example with source code

```
import React from "react";
import ReactDom from "react-dom";
```

```jsx
import D from "./D.jsx";
export let userContext=React.createContext();
let App=()=>{

   return <><h1>I am App component</h1>

         <userContext.Provider value={"TECH HUB"}>
            <F/>
         </userContext.Provider>

   </>

}
export default App;
```

If we want to consume data at destination place we have to create Consumer

```jsx
import React from "react";
import ReactDom from "react-dom";
import { userContext } from "./App.jsx";
let F=()=>{

    return <>
       <userContext.Consumer>
        {
            user=>{
                return <h1>Data from APP {user}</h1>
            }
        }
       </userContext.Consumer>
    </>
}
export default F;
```

If we think about it above we use Context API but we are required to perform n number of processes or it is completed when we have nested context so better way you can use context hook.

Steps to use context hook

When we use the contextHook then it is similar to create context and providing value just the context hook help us to consume the context value easily

1. **Import useConext hook in user component**

App.jsx

```
import React from "react";
import ReactDom from "react-dom";
import D from "./D.jsx";
import F from "./F.jsx"
export let userContext=React.createContext();
let App=()=>{

   return <><h1>I am App component</h1>

         <userContext.Provider value={"TECH HUB"}>
            <F/>
         </userContext.Provider>

   </>

}
export default App;
```

F.jsx

```
import React ,{useContext} from "react";
import ReactDom from "react-dom";
import { userContext } from "./App.jsx";
let F=()=>{
   let data=useContext(userContext);
    return <>
       <h1>Data from App compoent {data}</h1>
    </>
}
export default F;
```

useRef() hook

useRef hook is used for

1. **Persist mutable values:** that does not cause re-renders when updated.
2. **Access DOM elements directly:** in functional components it returns a ref object with a refvariable.current property that you can read or update .

Benefits of useRef hook

a. **Avoid unnecessary re-renders :** unlike state, updating current does not trigger re-renders
b. **Access DOM elements** : can get a reference to many DOM elements in functional components.
c. **Store mutable values:** useful for keeping track of values across render(e.g times, previous values)
etc

Example with source code

```
import React, { useEffect, useState,useRef } from "react";
import ReactDom from "react-dom";
let App=()=>{
 let [count,setCount]=useState(0);
 let a=useRef(0);
 useEffect(()=>{
    a.current=a.current+1;
   console.log("i am effect function "+a.current);
 });
  return <>
      <h1>Counter is {count}</h1>
        <button onClick={(e)=>setCount(count+1)}
           style={{width:"400px",height:"40px"}}>Click Me</button>
    </>
}
export default App;
```

If we want to work with useRef we have some important points

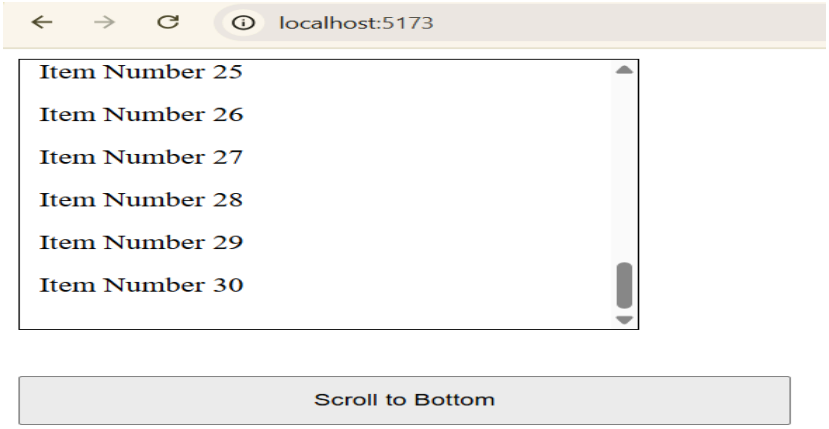
1. **Do not use useRef for reactive state: if we want React to update UI when a value changes ,use state (useState) instead.**
2. **Refs persist across renders:** the same ref object is kept for the component lifetime.
3. **Don't mutate DOM directly except for refs:** React handles DOM updates only use ref for focus,selection or third party DOM manipulations
4. **Assign ref to React elements or components :** The ref attribute can be attached to DOM elements or class component but not functional component etc

Scrolling Example using useRef

```
import React, { useEffect, useState,useRef } from "react";
import ReactDom from "react-dom";
let App=()=>{
  let containerRef=useRef(null);
   console.log(containerRef);
   let scrollToBottom=()=>{
     let div=containerRef.current;
```

```
    if(div){
     div.scrollTop=div.scrollHeight;
    }
  }
 return <>

      <div
      ref={containerRef}
        style={
             {
               height:"200px",
               width:"300px",
               border:"1px solid black",
               overflowY:"auto",
               padding:"10px"
             }
        }>
        {
         [...Array(30)].map((_,index)=>(<p>Item Number {index+1}</p>))
        }
      </div>
      <br></br><br></br>
       <button  onClick={scrollToBottom}
          style={{width:"400px",height:"40px"}}>Scroll to
Bottom</button>
    </>
}
export default App;
```

Output



useCallBack hook

---

useCallBack hook is a React Hook that returns a memorized version of a call function
That only changes if one of the dependencies has changed.

Syntax:

```
let memorizedCallBack= useCallBack(()=>{

},[  dependencies  ] );
```

()=>{}: here you can specify a callback which we want to memorise.
[dependencies] : an array of dependencies . the function is re-created only if any of the values changes.

Why use useCallback?

In React functions are re-created on every render. This can cause problem when passing function to child components as props,especially when those children are memorized with React.memo
Using a function inside effect hooks (useEffect, useMemo) or other hooks that depend on reference equality

Q. When to use useCallBack hook?

---

1. You are passing a callback to memorized child component
2. You want to prevent unnecessary re-renders
3. You are seeing performance issues with large or complex components.

Q. When not to use the useCallBack hook?

---

1. The component is small /simple and performance is not affected
2. The function does not rely on props or state - it can be defined outside of the component

Etc

React Redux Toolkit and Redux

Q. What is redux?

Redux is a JavaScript state management library used for managing and controlling application state
When we use redux with react it helps managing complex state logic outside of components , making state predictable and easier to debug

Redux Provide

---

1. A global state for application data
2. Action to describe state changes
3. Reducers to update state based action

Q. Why use Redux?

React components have local state using a useState() or useReducer() but as your app grow ,managing shared or nested between components become difficult to manage state
If want to solve this problem react suggest us use redux or use third party library or tool known as reactjs/tool and react-redux

Benefits of Redux

---

1. **Centralise state management :** all state in application manage at one place or centralize place.

2. **Predictable state changes :** with actions and reducers state changes are trackable
3. **Debugging Tools :** Redux dev tools help you inspect and time travel to changes
4. **Consistency in all components :** share data between unrelated components easily
5. **Middleware support :** great for logging,crash reporting , API calls (via Reduc-thunk/saga)

If we want to work with redux we have to know the following terms

1. **Store:** hold your app's entire data.

   Important points related with store

   ---

   a) Basically store is JavaScript object hold entire state of application
   b) It work like as centralize database for application
   c) There is only one store per redux application
   d) Store provide few important methods
      1. getState(): to read current state
      2. dispatch(action): to send action update state
      3. subscribe(listen): to listen state for changes.

2. **Action : describe an event**
   An action is a plain JavaScript object that describes what happens in an app.
   a. It must have property (a string) that identifies action
   b. It can optionally have payload property carrying data related to the action

3. **Reducer : a reducer is pure function that take two inputs**
   a. Current state
   b. An action object
   c. It return new state based on the actio**n**
   d. Reducer are not mutable the current state the create return

4. **Dispatch :** dispatch is a function available on the store when you dispatch on action you are sending that action to reducer
   The reducer process returns a new state. Dispatch is the only way to trigger state changes in redux.
5. **Subscribe :** you can subscribe to the store to listen changes whenever the state updates the subscribed listen function are called this is UI stays in sync with current state

How to use redux practically

---

1. Create react project
2. Install reduxjs/tool kit and react-redux dependencies

```
D:\july 2024\reactprojects\reduxcounterapp>npm install @reduxjs/toolkit@latest react-redux

added 10 packages, and audited 208 packages in 12s

34 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

D:\july 2024\reactprojects\reduxcounterapp>
```

Now we want to create simple counter application using redux means we want to create two buttons increment and decrement where increment increase value by 1 and decrement decrease value by 1

3. **Create reducer and action using tool kit**
   If we want to create reducer and action using tool kit we have createSlice function from @reduxjs/tookit library

Example with source code

```
import { createSlice } from "@reduxjs/toolkit"

 let counterSlice=createSlice({
    name:'counter',
    initialState:0,
    reducers:{
       ntate=>state-1
    }
 });

 export let {increment,decrement}=counterSlice.actions;
 export default counterSlice.reducer;
```

Q. What is createSlice?

It is a function for create reducer and action using react tool kit the benefit of this function is writing separate action type and creating reducer so we can createSlice

Properties:
name:'counter': this names the slice counter redux toolkit uses this name to generate action type like as counter/increment , counter/decrement etc

initialState:0 : this is the initial state value of the slice here sample number is 0

reducers: an object where each key is a reducer function(case reducer) that defines how to update state.
each reducer function state as first augment
How we can define two reducer

A. **Increment :** add 1 to from current state
B. **Decrement** : subtract 1 from current state

Q. Why this approach?

---

1. This approach encapsulates all related actions and reducer logics in one place ,reducing boilerplate
2. The function you can provide acts as a handler for specific actions.

Now we want to discuss about following code

```
export const {increment,decrement} = counterSlice.actions
export default counterSlice.reducer;
```

export const {increment,decrement} = counterSlice.actions : this line the action creator from slice to and export them

```
{
  type:‘counter/increment’
}
```

Why do we export it?

Export the component or other parts of your app can dispatch actions to update it

Example: dispatch(increment());

export default counterSlice.reducer: export reducer function as default export.
This reducer is function that reducer will use to update the state based on dispatch action

3. **Create store:**

Store.jsx

```
import { configureStore } from "@reduxjs/toolkit";
import counterReducer from "./CounterSlice.jsx";

const store=configureStore({
     counter:counterReducer
});
export default store;
```

Code explanations

import {configureStore} from “@reactjs/toolkit”;

This imports the configureStore function from redux toolkit and it is a simplified way to create a redux store.
Automatically setup store default like as

1. Redux dev tool support
2. Middleware like as redux-thunk for async logic
3. Better error checking development

Note: it replace the older createStore method from redux making stop setup easier faster

import counterReducer from “./CounterSlice.jsx”;

This import reducer function that you exported as the default from CounterSlice.jsx

Why is it used?

1. Redux needs a reducer to manage each slice for application state.
2. As per our example reducer handle counter related action like as increment or decrement

```
const store=configureStore({
     counter:counterReducer
});
```

What does it do?

---

1. **This create redux store and configure it with redux**
2. **This reducer is an object where the key (counter) become name of slice of state in the redux store**
3. **The value counterReducer is the reducer function responsible for updating that slice.**

Note: this function created the name of slice with reducer keyword with camel case format.

Example: our slice name is name:'counter' and export default counterSlice.reducer so this default export created as counterReducer when we import it then you use it like as

```
export default store;
```

We can use or import store with provider in application

Now we want to open the main.jsx file and import the store and provide it to the App using the provider shown in the following code.

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.jsx'
import { Provider } from 'react-redux'
import store from "./Store.jsx";
createRoot(document.getElementById('root')).render(
   <Provider>
     <App store={store}/>
   </Provider>
  )
```

Now we want to open App.jsx and customize following code
