Table of Contents

[REACTJS 2](#_Toc186215045)

[Introduction: 2](#_Toc186215046)

[**What is reactjs?** 2](#_Toc186215047)

[**Features** 2](#_Toc186215048)

[**COMMANDS IN REACT.JS** 3](#_Toc186215049)

[**What is Single Page Application(SPA)?** 3](#_Toc186215050)

[**When to Use React.js?** 3](#_Toc186215051)

[**What is jsx?** 3](#_Toc186215052)

[Components in Reactjs? 3](#_Toc186215053)

[**Functional Components:** 3](#_Toc186215054)

[**Class Components:** 4](#_Toc186215055)

[Import and Exports 5](#_Toc186215056)

[**Imports:** 5](#_Toc186215057)

[**Exports:** 5](#_Toc186215058)

[Types of Exports in Reactjs are: 5](#_Toc186215059)

[Named Exports : 6](#_Toc186215060)

[Default Exports: 7](#_Toc186215061)

[Props in ReactJs 7](#_Toc186215062)

[**By using Functional component**: 7](#_Toc186215063)

[MAP() METHOD 9](#_Toc186215064)

[How to use map( ) method with props in React 9](#_Toc186215065)

[Child props in React 11](#_Toc186215066)

[State() and useState() 11](#_Toc186215067)

[State() method used in class component 11](#_Toc186215068)

[useState() method is used in Functional Component 11](#_Toc186215069)

[Show and hide the details of bank by using useState() Method 13](#_Toc186215070)

[Back Ground Color Changer 14](#_Toc186215071)

[Counter App 17](#_Toc186215072)

[Props Destructuring 21](#_Toc186215073)

[Todo App 21](#_Toc186215074)

[Data Binding 21](#_Toc186215075)

[**One way Data Binding** 22](#_Toc186215076)

[**Two way data binding** 23](#_Toc186215077)

[Calculator app 24](#_Toc186215078)

[What is virtual dom 28](#_Toc186215079)

[Conditional Rendering 28](#_Toc186215080)

[Use Effect() hook 30](#_Toc186215081)

[What is a hook? 30](#_Toc186215082)

[Types of Behavior in useEffect: 30](#_Toc186215083)

[Ways to include local images/media files in jsx 33](#_Toc186215084)

[Routing Concept 33](#_Toc186215085)

[**Nested routing/child routing** 34](#_Toc186215086)

[Types of routing 38](#_Toc186215087)

[Static routes 38](#_Toc186215088)

[Dynamic routes 38](#_Toc186215089)

[Routing parameters 38](#_Toc186215090)

[Query Parameters 38](#_Toc186215091)

[How to use query parameters 38](#_Toc186215092)

[Route/Path Parameters 38](#_Toc186215093)

[How to use route parameters 38](#_Toc186215094)

# **REACTJS**

# **Introduction:**

## **What is reactjs?**

* React.js is a **JavaScript library** developed by Facebook, primarily used for building user interfaces (UIs) for web applications.
* It allows developers to create reusable, self-contained UI components, making complex application interfaces easier to build, maintain, and scale.

## **Features**

* **Component-Based Architecture:**
* The UI is broken down into reusable, modular components.
* **Virtual DOM:** A lightweight representation of the real DOM(Document Object Module), enabling efficient updates and rendering.
* **Declarative Syntax**: Allows developers to declare what they want the UI to look like, and React takes care of updating it.
* **Unidirectional Data Flow:** Data flows in a single direction, making state management more predictable and debugging easier.

**NOTE:-React.js creates a Single Page Application (SPA)**, we mean that React enables the development of web applications that load a **single HTML page initially** and then dynamically update content without fully reloading the page.

## **COMMANDS IN REACT.JS**

**npx create-react-app -app :** Create a new react application

**npm start :** Runs the application in development mode on a local server (default http://localhost:3000).

**npm install :** Installs all dependencies listed in package.json.

## **What is Single Page Application(SPA)?**

A **Single Page Application (SPA)** is a web application that loads a single HTML page initially and uses JavaScript to update the content dynamically as users interact with the app. Instead of navigating between different HTML pages, the SPA only changes the specific parts of the page that need updating, which provides a seamless user experience.

## **When to Use React.js?**

**Building Single-Page Applications (SPAs)**: React is excellent for SPAs where only parts of the page update rather than reloading the entire page. This creates a smooth, app-like experience for users.

**Developing Dynamic Web Applications**: If your app has interactive features (like social feeds, dashboards, or complex forms), React’s state management and component architecture make it ideal.

**Frequent Updates or Changing Data**: React’s Virtual DOM and component reusability make it efficient to update and manage dynamic data or data that changes frequently.

## **What is jsx?**

JSX (JavaScript XML), is a syntax extension for JavaScript often used with React. It allows you to write HTML-like code within JavaScript, making it easier to build and structure user interfaces.

# **Components in Reactjs?**

It has two components:

* Functional Components and
* Class Components

## **Functional Components:**

Functional components are JavaScript functions that accept props as an argument and return React elements.

**EX:**

import React from 'react';

import ReactDOM from 'react-dom/client';

const root=ReactDOM.createRoot(document.getElementById('root'));

function Header(){

let menu= [

    'demo', 'Welocome to ReactJs', 'About ReactJs'

];

    return(

    <header>

        <ul>

            <li>{menu[0]}</li>

            <li>{menu[1]}</li>

            <li>{menu[2]}</li>

        </ul>

    </header>

    );

}

  root.render(<Header/>

    );

## **Class Components:**

Class components are ES6 classes that extend **React.Component** and have a **render()** method, which returns React elements.

**EX:**

import React from 'react';

import ReactDOM from 'react-dom/client';

const root=ReactDOM.createRoot(document.getElementById('root'));

class Demo extends **React.Component**{

    render(){

        let demoDetails=[

            'ReactDemo', 'Welcome to ReactJs','About ReactJs'

        ]

        return (

                <header>

                    <ul>

                        <li>{demoDetails[0]}</li>

                        <li>{demoDetails[1]}</li>

                        <li>{demoDetails[2]}</li>

</ul>

                </header>

                )

    }

}

root.render(<Header/>

    );

# Import and Exports

It was introduced in js ES6 version

In React, **imports** and **exports** are used to manage and organize code by separating it into **modules or files.** This allows code to be reused across different parts of an application, keeping it modular and organized**.**

## **Imports:**

The purpose of the import statement is to bring in code from other files or modules into the current file, which helps to keep code **organized, modular, and reusable.**

## **Exports:**

### **Types of Exports in Reactjs are:**

### **Named Exports** :

Named exports allow multiple exports and must use curly braces in import.

When importing, you must use the exact names enclosed in **curly braces { }.**

EX:

**App.js**

Named Export

export let **emp**={

name:'Silpa',

sal:20000,

  addr:'nellore'

}

**Index.js**

import React from 'react';

import ReactDOM from 'react-dom/client';

**Here we are importing the named export in curly braces{emp}**

import **{emp}** from './App'

const root=ReactDOM.createRoot(document.getElementById('root'));

function Header(){

console.log(emp)

let menu= [

    'demo', 'Welocome to ReactJs', 'About ReactJs'

];

    return(

    <header>

        <ul>

            <li>{menu[0]}</li>

            <li>{menu[1]}</li>

            <li>{menu[2]}</li>

            </ul>

    </header>

    );

}

    root.render(<Header/>

    );

### **Default Exports:**

Default exports allow **only one default** export per file and do not require curly braces in import.

**EX:**

App.js

let x=12

export default x

Index.js

import React from 'react';

import ReactDOM from 'react-dom/client';

Default export

import **x** from './App'

const root=ReactDOM.createRoot(document.getElementById('root'));

function Header(){

console.log(x)

}

root.render(<Header/>

    );

# **Props in ReactJs**

The main purpose of props(Properties) in reactjs, we can pass some information from Parent Component to Child Component.

Props are **immutable**.

Props are basically used at the run time.

## **By using Functional component**:

Once these are defined we can get those values in the component using parameters based on the functional component. So directly we can display those values in components using props. parameter name.

**Ex:**

**App.js 🡪Parent Component**

import User from './User'

function App(){

    return(

        < > **🡪fragments**

        <User name='Silpa' age='22' Address='hyd'/>

        <User name='Janu' age='3' Address='Chennai'/>

         </>

        )

}

export default App

**User.js 🡪Child component**

function User(props){

    console.log(props)

    return(

        <div>

        <h1>Name:{props.name}</h1>

        <h2>age:{props.age}</h2>

        <h3>Address:{props.Address}</h3>

        </div>

    )

}

**o/p:**

Name:Silpa

age:22

Address:hyd

Name:Janu

age:3

Address:Chennai

**By using Class Component:**

In class components directly we can get through the constructor or use this. props. parameter name, but constructors are not mandatory.

**App.js -🡪parent Component**

import User from './User'

function App(){

    return(

        <>

        <User name='Silpa' age='22' Address='hyd'/>

        <User name='Janu' age='3' Address='Chennai'/>

         </>

        )

}

export default App

**User.js -🡪Child Component**

import React from "react"

class User extends React.component{

    render(){

        return(

                    <div>

                    <h1>Name:{this.props.name}</h1>

                    <h2>age:{this.props.age}</h2>

                    <h3>Address:{this.props.Address}</h3>

                    </div>

                )

    }

}

export default User

**o/p:**

Name:Silpa

age:22

Address:hyd

Name:Janu

age:3

Address:Chennai

# **MAP() METHOD**

## How to use map( ) method with props in React

When we try to call the Same component with different values it creates a lot of time and increases the lines of code.

To overcome this issue we have to prefer the **Map** method.

By using that **key** react library will re-render the elements fastly.

**User.js 🡪Child Component**

function User(props) {

    console.log(props);

    return (

        <div>

            <h1>id: {props.id}</h1>

            <h1>Name: {props.name}</h1>

            <h2>Age: {props.age}</h2>

            <h2>Address: {props.Address}</h2>

        </div>

    );

}

export default User;

**App.js 🡪Parent Component**

import User from './User'

function App(){

    let users=[

        {

            id:'10',

            name:'Silpa',

            age:23,

            Address:'Nellore'

    },

    {

        id:'11',

        name:'Janvi',

        age:3,

        Address:'Chennai'

    }

]

    return(

        < >

        {

            users.map(

            (u)=>{

                return<User key ={u.id} id={u.id} name={u.name} age={u.age} Address={u.Address}/>

                }

        )

    }

        </>

    )

}

export default App

**index.js 🡪root component**

import React from "react";

    import ReactDOM from "react-dom/client";

    import A from'./App'

    const root=ReactDOM.createRoot(document.getElementById('root'));

      root.render(<A/>

        );

**O/p:**

id: 10

Name: Silpa

Age: 23

Address: Nellore

id: 11

Name: Janvi

Age: 3

Address: Chennai

# **Child props in React**

* We can access this child props using **“props.children property”.**
* Whatever we write b/n two tags in the child component either function or class component. In this case, we have to fetch those children using props.
* Children Property (props.children) in the child component are called “Child props”.

**EX:**

**App.js 🡪 child component**

function App(props){

    console.log(props)

    return(

        <div>

            <h1>This is Heading</h1>

            <p>This is Para</p>

            <p>{props.children}</p>

        </div>

    )

}

export default App

**index.js**

import React from "react";

    import ReactDOM from "react-dom/client";

    import A from'./App'

    const root=ReactDOM.createRoot(document.getElementById('root'));

      root.render(

      <A name='ss'>This is App Component</A>

        );

**o/p:**

This is Heading

This is Para

This is App Component

# **State() and useState()**

State() method used in class component and

useState() method is used in Functional Component.

useState is nothing but it is a method in this method we have to pass the default values for which variables are dynamically changing.

**EX:**

**App.js**

import React, { useState } from "react";

function App() {

  const [product, updateProduct] = useState({

    pname: "Poco M3",

    price: 15000,

  });

  return (

    <>

      <h1>Product: {product.pname}</h1>

      <p>Price: {product.price}</p>

      <h1>Product: {JSON.stringify(product)}</h1>

      <input id="pname" placeholder="Enter new price" />

      <button

        onClick={() => {

          let mprice = document.getElementById("pname").value;

          updateProduct({

            ...product,

            price: mprice,

          });

        }}

      >

        Update

      </button>

    </>

  );

}

export default App;

**index.js**

import React from "react";

    import ReactDOM from "react-dom/client";

    import A from'./App'

    const root=ReactDOM.createRoot(document.getElementById('root'));

      root.render(<A/>

      )

**O/P**

Product: Poco M3

Price: 20000

Product: {"pname":"Poco M3","price":"20000"}

update

Enter the new value

### **Show and hide the details of bank by using useState() Method**

**App.js**

import React, { useState } from 'react';

import BankDetails from './BankDetails';

function App() {

  const [showDetails, setShowDetails] = useState(false);

  const Details = () => {

    setShowDetails(!showDetails);

  };

return (

    < >

      <h1>Bank Information</h1>

      <BankDetails

        bankName="SBI"

        bankAddress="Gudur"

        showDetails={showDetails} />

      <button onClick={Details}>  Bank Details</button>

    </>

  );

}

export default App;

**index.js**

import React from "react";

    import ReactDOM from "react-dom/client";

    import A from'./App'

    const root=ReactDOM.createRoot(document.getElementById('root'))

root.render(<A/>

      )

BankDetails.js

import React from 'react';

function BankDetails({ bankName, bankAddress, showDetails }) {

  return (

 <div style={{ marginTop: '20px' }}>

    {showDetails && (

        <>

          <p><strong>Bank Name:</strong> {bankName}</p>

          <p><strong>Bank Address:</strong> {bankAddress}</p>

        </>

      )}

 </div>

 );

}

export default BankDetails;

**O/P:**

Bank Information

Bank Name: SBI

Bank Address: Gudur

Bank details

Button(when we click this button it will show the bank details and if we click the button 2nd time it will hide the bank details

# **Back Ground Color Changer**

**App.js**

import './App.css'

function App(){

  return(

    <>

    <div className="Wrapper"></div>

    <h1>backGround Colour Changer</h1>

    <input type="color" onChange={

      (e)=> {

       document.body.style.background= e.target.value

      }

    }/>

    </>

  )

}

export default App;

**index.js**

import React from "react";

    import ReactDOM from "react-dom/client";

    import A from'./App'

    const root=ReactDOM.createRoot(document.getElementById('root'));

      root.render(<A/>)

**App.css**

.Wrapper{

  display: flex;

    justify-content: center;

    align-items: center;

    flex-direction: column;

    gap: 10px;

}

**global.css 🡪created in public**

\*{

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

 body {

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

}

**Index.html**

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta name="theme-color" content="#000000" />

    <meta

      name="description"

      content="Web site created using create-react-app"

    />

    <link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

    <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

    <link rel="stylesheet" href="./global.css">

    <title>React App</title>

  </head>

  <body>

    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="root"></div>

  </body>

</html>

# Counter App

**Enterprise Applications Context**

In the context of **enterprise applications like e-commerce platforms**, the principles learned from a counter app are expanded:

1. **Cart Management**:
   * A counter can represent the quantity of items in a shopping cart.
   * Buttons can allow users to increase or decrease the item count dynamically.
2. **Inventory Control**:
   * Helps manage stock levels in real-time, displaying the count of available products.

**Example Use in E-commerce:**

Imagine you're purchasing a product:

1. You see a "Quantity" field with + and - buttons.
2. Clicking these buttons updates the total cost in real-time.
3. This feature is essentially a counter app adapted for an enterprise-level requirement.

**Ex:**

**App.js**

import './App.css';

import { useState } from 'react';

function App() {

  const [counter, setCounter] = useState(0); // Correct state initialization

  const stock = 12; // Max stock count

  return (

    <div className="Wrapper">

      <h1>CounterApp</h1>

      {/\* Input to change background color \*/}

      <input

        type="color"

        onChange={(e) => {

          document.body.style.background = e.target.value;

        }}

      />

      {/\* Button to decrement counter \*/}

      <button

        className="minus"

        disabled={counter === 0}

        onClick={() => {

          if (counter > 0) {

            setCounter(counter - 1);

          }

  }}

      >   - </button>

      {/\* Dynamically displaying the counter value \*/}

      <p className="counter">{counter}</p>

      {/\* Button to increment counter \*/}

      <button

        className="plus"

        disabled={counter === stock}

        onClick={() => {

          if (counter < stock) {

            setCounter(counter + 1);

          }

        }}

      >    + </button>

    </div>

  );

}

export default App;

**index.js**

import React from "react";

    import ReactDOM from "react-dom/client";

    import A from'./App'

    const root=ReactDOM.createRoot(document.getElementById('root'));

      root.render(

      <A/>

      )

**global.css**

\*{

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

 body {

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

}

**App.css**

.Wrapper {

  display: flex;

  flex-direction: column;

  justify-content: center;

  align-items: center;

  gap: 10px;

}

.counter {

  font-size: 20px;

}

.Wrapper button {

  border: none;

  outline: none;

  padding: 10px 15px;

  font-size: 21px;

  font-style: oblique;

  border-radius: 5px;

  cursor: pointer;

  color: white;

}

button.minus {

  background-color: blueviolet;

}

button.plus {

  background-color: red;

}

**Index.html**

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta name="theme-color" content="#000000" />

    <meta

      name="description"

      content="Web site created using create-react-app"

    />

    <link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

    <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

    <link rel="stylesheet" href="./global.css">

    <title>React App</title>

  </head>

  <body>

    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="root"></div>

  </body>

</html>

# **Props Destructuring**

The main purpose of Props Destructuting in react is used to pass and receive the values in a simple way.

When we send any object data using props, we can generally send individual information; if we need to send entire object information using props, we have to prefer the spread operator in the parent component.

• Ex: <Component {…product} />

We can maintain it easily, which improves readability using this concept.

# **Todo App**

# Data Binding

Data binding in React refers to the process of storing data or information in one place and displaying it in the browser. There are two types of data binding in React: one-way and two-way data binding.

* **One-way data binding** means data flows in a single direction—from the JavaScript file to the HTML.
* **Two-way data binding** allows data to flow in both directions, from the JavaScript file to the HTML and back from the HTML to the JavaScript file. This is achieved using the state concept in React.

## **One way Data Binding**

* **One-way data binding** means data flows in a single direction—from the JavaScript file to the HTML.
* Using the state concept we can achieve the one-way data binding.
* State concept is compulsory to achieve the one-way data binding.

**App.js**

import { useState } from "react"

function OneWayDataBinding(){

  let[num,upDateNum]=useState(10);

  return(

    <>

       <h1>{num}</h1>

       <button onClick={()=>

          upDateNum(20)

       }

       >click</button>

    </>

  )

}

export default OneWayDataBinding;

**index.js**

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import OneWayDataBinding from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

    <OneWayDataBinding />

);

## **Two way data binding**

* **Two-way data binding** allows data to flow in both directions, from the JavaScript file to the HTML and back from the HTML to the JavaScript file. This is achieved using the state concept in React.
* When we update in js it should be effect in html or if we update the value in html it should be update in js also.
* In React by default it is not possible to update the javascript value when we update the value in html. We have to update the state value to achieve two-way data binding.
* The **state concept in React** is a core feature used to manage dynamic data within a component.
* **State allows React components to create and manage their own data, and when the state changes, the component re-renders automatically to reflect the updated data in the UI.**

**App.js**

import { useState } from "react"

function TwoWayDataBinding(){

  let[num,upDateNum]=useState(10);

  return(

    <>

       <h1>{num}</h1>

       <button onClick={()=>

          upDateNum(num)

       }

       >click</button>

<br></br>

       <input type='text' onChange= { (e)=>{

          let temp=e.target.value;

          upDateNum(temp)

       }

}

       value={num}/>

    </>

  )

}

export default TwoWayDataBinding;

**index.js**

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import TwoWayDataBinding from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

    <TwoWayDataBinding />

);

# **Calculator app**

**App.js**

import React, { useState } from "react";

import Keypad from "./Keypad";

import "./App.css";

function App() {

  let [input, setInput] = useState("");

  function HandleClick(value) {

    setInput(input + value);

  }

  function Calculate() {

    try {

      let outputVal = eval(input);

      setInput(String(outputVal));

    } catch (e) {

      setInput("Error");

    }

  }

  function HandleClear() {

    setInput("");

  }

  return (

    <div className="container">

      <h1>Calculator App Using React</h1>

      <div className="Calculator">

        <input type="text" value={input} className="output" readOnly />

        <Keypad HandleClick={HandleClick} Calculate={Calculate} HandleClear={HandleClear} />

      </div>

    </div>

  );

}

export default App;

**Keypad.js**

import React from "react";

function Keypad({ HandleClick, Calculate, HandleClear }) {

  return (

    <div className="keypad">

      <div className="row">

        <button className="digit" onClick={() => HandleClick("7")}>7</button>

        <button className="digit" onClick={() => HandleClick("8")}>8</button>

        <button className="digit" onClick={() => HandleClick("9")}>9</button>

        <button className="operator" onClick={() => HandleClick("/")}>/</button>

      </div>

      <div className="row">

        <button className="digit" onClick={() => HandleClick("4")}>4</button>

        <button className="digit" onClick={() => HandleClick("5")}>5</button>

        <button className="digit" onClick={() => HandleClick("6")}>6</button>

        <button className="operator" onClick={() => HandleClick("\*")}>\*</button>

      </div>

      <div className="row">

        <button className="digit" onClick={() => HandleClick("1")}>1</button>

        <button className="digit" onClick={() => HandleClick("2")}>2</button>

        <button className="digit" onClick={() => HandleClick("3")}>3</button>

        <button className="operator" onClick={() => HandleClick("-")}>-</button>

</div>

      <div className="row">

        <button className="digit" onClick={() => HandleClick("0")}>0</button>

        <button className="fun-key" onClick={Calculate}>=</button>

        <button className="fun-key" onClick={HandleClear}>C</button>

        <button className="operator" onClick={() => HandleClick("+")}>+</button>

      </div>

    </div>

  );

}

export default Keypad;

**index.js**

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

    <App />

);

**App.css**

.container{

  display: flex;

  justify-content: center;

  align-items: center;

  flex-direction: column;

  height: 100vh;

}

.output{

  outline: none;

  border: none;

  height: 70px;

  width: 320px;

  font-size: 32px;

  padding: 2px 10px;

  color: white;

  background-color: #333433;

  border-top-right-radius: 10px;

  border-top-left-radius: 10px;

}

  button{

  width: 85px;

  height: 90px;

  font-size: 27px;

  cursor: pointer;

  border:none;

  color: black;

  border-top:1px solid #666;

  border-right:1px solid #666;

}

.operator{

  border-right: none;

  background-color: orange;

}

.operator:hover{

  background-color:skyblue;

}

.fun-key:hover{

  background-color: lightcoral;

}

# What is virtual dom

# **Conditional Rendering**

Conditional Rendering means based upon condition value we have to show the information in browser using JSX.

Here we have 2 types of conditions

One is like using ternary operator

Other one is using double and operator.

**App.js**

import'./App.css';

function App(){

  return(

    <>

    <Header></Header>

    {

      // isAdmin && <Admin></Admin>

      isAdmin?<Admin></Admin>:null

    }

</>

  )

}

  let loggedIn=true;

  // let loggedIn=false;

  let isAdmin=true;

  // let isAdmin=false;

  function Header(){

    return(

      <ul>

        <li>Home</li>

        <li>AboutUs</li>

        <li>ContactUs<li>

{

        loggedIn && (<li>Logout</li>)

        // loggedIn? (<li>Logout</li>) : (<li>Login</li>)

  }

      </ul>

    )

  }

  function Admin(){

    return(

      <h1>This is Admin</h1>

    )

  }export default App;

**index.js**

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

    <App />

);

**App.css**

ul{

  background-color: blue;

  color: white;

  display: flex;

  align-items: center;

  font-size: 22px;

  padding: 22px;

  gap: 21px;

  list-style-type: none;

}

# **Use Effect() hook**

Actually, The hook concept was introduced in react 16th version. The hooks are used in functional component.

Hook is nothing but a **“method”.**

## What is a hook?

* A hook is a function provided by React to handle specific functionalities.
* While it is possible to continue work without using **useEffect,** using it can help resolve certain issues. Some use cases for useEffect include:
  1. Fetching data from an API.
  2. Making changes to a component after it has been loaded.
  3. Removing a component from the DOM.
  4. Adding events to the DOM.

**Syntax:**

**useEffect(callback, [dependencies(optional)])**

* We need to import **useEffect** from the **React library.**

### Types of Behavior in useEffect:

1. If the second argument is omitted, the useEffect will be called every time the component renders.

**Ex:**

**useEffect(**

**( )=>{**

**console.log(“use effect”)**

**}**

**);**

1. If the second argument is an empty array ([]), useEffect will run only once, after the component has rendered for the first time.

**EX:**

**useEffect(**

**( )=>{**

**console.log(“use effect”)**

**}, [ ]**

**);**

1. If the second argument contains dependencies (e.g., [dependency]), the useEffect will be called whenever any of the specified dependencies change.

**Ex:**

**useEffect(**

**( )=>{**

**console.log(“use effect”);**

**getProducts( )**

**}[ searchInput]**

**);**

**EX:**

**TestEffect.js**

import { useEffect, useState } from "react";

import React from "react";

function TestEffect(){

    const [timer, setTimer]=useState(0);

    const [counter, setCounter]=useState(0);

    useEffect(

        ()=>{

            console.log("Inside use Effect");

        },

[counter]

    )

    setInterval(

        ()=> {

            setTimer(timer+1)

        },40000) ;

    return(

        <>

        <h1>Timer:{timer}

            {

                console.log("Inside Component")

            }

        </h1>

        <button onClick={()=>setTimer(timer+1)

        }>Update Timer</button>

        <button onClick={()=>setCounter(counter+1)

        }>Update Counter</button>

        </>

         )

        }

        export default TestEffect;

**App.js**

import React from "react";

import TestEffect from "./TestEffect";

function App(){

  return(

    <TestEffect></TestEffect>

  )

}

export default App;

**index.js**

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

    <App />

);

# Ways to include local images/media files in jsx

1.We can fetch the images inside component from public folder or src folder only. Outside src or public folder we can't fetch images.

2. If it is in public directly we can use it.

3. If it is in src folder so we have to import from it then we can fetch it.

# Routing Concept

* 1. **Install Router Library:**

npm install react-router-dom

* 1. **Configure routes using BrowserRouter :**

Ex: <BrowserRouter>

<Header />

<Routes>

<Route path="/" element={<Home />} />

<Route path="about" element={<About />} />

</Routes>

</BrowserRouter>

* 1. Using the Link component

we have **to create Navigation links**

Ex: <ul>

<li>

<Link to="/">Home</Link>

</li>

<li>

<Link to="/about">About</Link>

</li>

</ul>

## **[Nested routing/child routing](https://www.youtube.com/create_channel?upsell=comment" \o "" \t ")**

We have to give children routes in the Parent component like this using Route Tag.

Ex:

<Route path="/user" element={<User />}>

<Route path="orders" element={<Orders />} />

<Route path="profile" element={<Profile />} /> </Route>

We have to import the Outlet, Link in the children's tag.

Once we Imported we have to call the **Outlet tag** in the component then only a particular child URL will call.

We have any no.of routes inside of the parent component it is called Child routes or nested routing.

We have to declare the nested routing like this.

Ex:

<Route>

<Route path="/user" element={<User />}>

<Route path="orders" element={<Orders />} />

<Route path="profile" element={<Profile />} />

</Route>

We have to import the outlet, Link classes from the react-router-dom library.

Once Importing is done we have to call the Outlet tag in the child component.

Ex:

<button><Link to="orders">Orders</Link></button>

<button><Link to="profile">Profile</Link></button>

<h3>This is User Component</h3>

<Outlet />

[Ex:](https://www.youtube.com/create_channel?upsell=comment" \o "" \t ")

**App.js**

import React from "react";

import { BrowserRouter, Routes, Route, Link } from "react-router-dom";

import Home from "./Home";

import ContactUs from "./ContactUs";

import AboutUs from "./AboutUs";

import Login from "./Login";

import User, {Orders, Profile} from "./User"; // Ensure correct import

function Header() {

  return (

    <ul>

      <li><Link to="/">Home</Link></li>

      <li><Link to="/about">About Us</Link></li>

      <li><Link to="/contact">Contact Us</Link></li>

      <li><Link to="/login">Login</Link></li>

      <li><Link to="/user">User</Link></li>

    </ul>

  );

}

function App() {

  return (

    <BrowserRouter>

      <Header />

      <Routes>

        <Route path="/" element={<Home />} />

        <Route path="about" element={<AboutUs />} />

        <Route path="contact" element={<ContactUs />} />

        <Route path="login" element={<Login />} />

        <Route path="user" element={<User />} >

        <Route path="order" element={<Orders />} />

        <Route path="profile" element={<Profile />} />

        </Route>

      </Routes>

    </BrowserRouter>

  );

}

export default App;

**Home.js**

import React from "react";

export default function Home() {

  return <h1>This is the Home Component</h1>;

}

**About.js**

import React from "react";

export default function AboutUs() {

  return <h1>This is the About Us Component</h1>;

}

**ContactUs.js:**

import React from "react";

export default function ContactUs() {

  return <h1>This is the Contact Us Component</h1>;

}

**Login.js:**

import React from "react";

export default function Login() {

  return <h1>This is the Login Page Component</h1>;

}

**User.js**

import React from "react";

import { Outlet } from "react-router-dom";

import { Link } from "react-router-dom";

export default function User() {

  return (

    <>

    <div>

        <button><Link to="order">Orders</Link></button>

        <button><Link to="profile">Profile</Link></button>

    </div>

      <h1>This is User Component</h1>

      <h1>This is Nested Routed Demo</h1>

      <Outlet />

    </>

  );

}

export function Orders() {

  return (

    <h1>This is  user Orders Component</h1>

  );

}

export function Profile() {

  return (

    <h1>This is Profile Component</h1>

  );

}

# Types of routing

## Static routes

<BrowserRouter>

<Routes>  
 <Route path=”/” element={< Home />} />

<Route path=”about” element={<About />} />

</Route>

</Routes>

</BrowserRouter>

## Dynamic routes

<Route path=”user/:id” element= {<User />} />

# Routing parameters

## Query Parameters

### How to use query parameters

**Import { useLocation} from “react-router-dom”**

**To access Query Parameters:**

**const location =useLoaction( );**

**const queryParams = new URLSearchParams(loaction.search);**

**const page\_no = queryParams.get(‘page’);**

## Route/Path Parameters

### How to use route parameters

* **Import { useParams} from “react-router-dom”**
* **To access route parameters:**

**Const {id} = useParams( );**