//birectional 🡺Increment number

Parent.jsx

 import { useState } from "react";

import Child from "./Components/Child"

const Parent = () =>

{

let[count,setCount]=useState(0)

   let Increment=()=>{

        return setCount(count+1);

        }

return (

    <div>

 <Child data={{count,Increment}} />

    </div>

  )

}

export default Parent

child.jsx

const Child = (props) => {

return (

    <div>

<h1>{props.data.count}</h1>

      <button onClick={()=> props.data.Increment()}>Increment</button>

    </div>

  )

}

export default Child

//Sending json object

Parent.jsx

import Child from "./Components/Child"

import JSON from "./Components/data.json"

const Parent = () =>

{

 return (

    <div>

 <Child data={JSON}/>

    </div>

  )

}

export default Parent

//child

import { Fragment } from "react";

const Child = (props) => {

 console.log(props);

  return (

    <div>

 {

        props.data.map(data=>

        {

return <Fragment key={data.id}>

    <img src={data.avatar\_url}></img>

</Fragment>

        })

      }

    </div>

  )

}

export default Child

//passing object without custom data

//parent

import Child from "./Components/Child"

// import JSON from "./Components/data.json"

const Parent = () =>

{

    /\*sending object explicilty \*/

    let obj={

        name:"moni",

        id:123,

        isAvailable:true,

        isUndefined:undefined,

        isNull:null,

        isSymbol:Symbol("monika"),}

 return (

    <div>

<Child {...obj}/>

    </div>

  )

}

export default Parent

//child

const Child = ({name,id,isNull,isAvailable,isUndefined,isSymbol}) => {

   return (

    <div>

      <h1>My name is {name}</h1>

      <h3>My Id is {id}</h3>

      <h3>It is {isAvailable==true ? "Available":"Not Available"}</h3>

      <h3> {isNull==null ? "It is null":"It is not null"}</h3>

      <h3> {isUndefined==undefined ? "It is undefined":"It is not undefined"}</h3>

      <h3>{isSymbol}</h3>

</div>

  )

}

export default Child

//nested object destructing

Parent

import Child from "./Components/Child"

// import JSON from "./Components/data.json"

const Parent = () =>

{

let obj={

    subject:"java",

    trainer:"shubam",

    place:{

        branch:"btm",

        area:"banglore",

        cost:{

            avg:2500,

            total:{

                sal:6700

            }

        }

    }

}

  return (

    <div>

<Child {...obj}/>

    </div>

  )

}

export default Parent

//child

const Child = ({ subject,trainer,place:{branch,area,cost:{avg,toatal:{sal}}}) => {

  return (

    <div>

 <h1>{subject}</h1>

       <p>{trainer}</p>

       <p>{branch}</p>

       <p>{area}</p>

       <p>{avg}</p>

       <p>{sal}</p>

    </div>

  )

}

export default Child

PROPS USING CLASS BASED COMPONENTS

//PARENT

import  { Component } from 'react'

import Child from './Components/Child'

export default class Parent extends Component {

  render() {

    return (

      <div>

        <Child name="monika" />

      </div>

    )

  }

}

//CHILD

import  { Component } from 'react'

import Child from './Components/Child'

export default class Parent extends Component {

  render() {

    return (

      <div>

        <Child name="monika" />

      </div>

    )

  }

}

import  { Component } from 'react'

import Child from './Components/Child'

export default class Parent extends Component {

    state=

    {

        count:0,

    }

    Increment=()=>

    {

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

    }

    render() {

    return (

      <div>

      <h1>{this.state.count} from parent</h1>

      <hr/>

        <Child name="monika" val={this.Increment} />

      </div>

    )

  }

}

/\* eslint-disable react/prop-types \*/

import  { Component } from 'react'

export default class Child extends Component {

  render() {

    console.log(this

        )

    console.log(this.props.name);

    return (

      <div>

        <h1>My name is {this.props.name}</h1>

        <button onClick={() => this.props.val()}>Increment from child</button>

      </div>

    )

  }

}

//PROP.CHILD🡪generic boxes

//PARENT

import React from 'react'

import Child from './Components/Child'

const Parent = () => {

  return (

    <div>

      <Child name="moni"/><h3> I am from Chennai</h3><button>click</button>

     <Child name="kavi"/><h3> I am from Andhra</h3><button>click</button>

     <Child name="priya"/><h3> I am from Banglore</h3><button>click</button>

     <Child name="kani"/><h3> I am from Trichy</h3><button>click</button>

    </div>

  )

}

export default Parent

//CHILD

import React from 'react'

const Child = ({name,children}) => {

  return (

    <div>

    <h1>{name}</h1>

    <p>{children}</p>

    </div>

  )

}

export default Child

//prop-eg

/\* eslint-disable no-unused-vars \*/

import React from 'react'

import Child from './Components/Child'

import Login from './Components/Login'

import Signup from './Components/Signup'

const Parent = () => {

  return (

    <div>

    <Child>

    <Login>

     <h1>Login</h1>

    </Login>

    <Signup>

    <h1>SignUp</h1>

    </Signup>

</Child>

    </div>

  )

}

export default Parent

/\* eslint-disable react/prop-types \*/

/\* eslint-disable no-unused-vars \*/

import React, { Fragment } from 'react'

import Login from './Login'

import Signup from './Signup'

const Child = ({children}) => {

  return (

    <Fragment>

           {children}

    </Fragment>

  )

}

export default Child

/\* eslint-disable react/prop-types \*/

/\* eslint-disable no-unused-vars \*/

import React from 'react'

const Login = ({children}) => {

  return (

    <div>

    <h1>{children}</h1>

      <form>

        <div>

            <label htmlFor='username'>Username</label>

            <input type="text" placeholder='username'></input>

        </div>

        <div>

            <label htmlFor='password'>Password</label>

            <input type="password" placeholder='password'></input>

        </div>

        <div>

           <button>Login</button>

        </div>

      </form>

    </div>

  )

}

export default Login

/\* eslint-disable no-unused-vars \*/

/\* eslint-disable react/prop-types \*/

import React from 'react'

const Signup = ({children}) => {

  return (

    <div>

       <h1>{children}</h1>

      <form>

        <div>

            <label htmlFor='username'>Username</label>

            <input type="text" placeholder='username'></input>

        </div>

        <div>

            <label htmlFor='password'>Password</label>

            <input type="password" placeholder='password'></input>

        </div>

        <div>

           <button>SignUp</button>

        </div>

      </form>

    </div>

  )

}

export default Signup

//prop-types

//parent

/\* eslint-disable no-unused-vars \*/

import React from 'react'

import Child from './Components/Child'

const Parent = () => {

  return (

    <div>

   <Child name={1} company="q spiders" sal={12000}/>

   <Child name={"monika"} company="j spiders" sal={15000}/>

   <Child name={"priya"} company="py spiders" sal={17000}/>

    </div>

  )

}

export default Parent

//child

/\* eslint-disable react/prop-types \*/

/\* eslint-disable no-unused-vars \*/

import PropTypes from "prop-types";

const Child = (props) => {

  return (

    <div>

      <h1>{props.name}</h1>

    </div>

  )

}

export default Child

Child.propTypes={

  name:PropTypes.string,

  company:PropTypes.string,

  salary:PropTypes.number,

}

Warning: Failed prop type: Invalid prop `name` of type `number` supplied to `Child`, expected `string`. at Child (<http://localhost:5173/src/Components/Child.jsx?t=1703824558214:19:99>) //to avoid developers bug

STATE:

A state is an object that holds data and information related to a React component. It can be used to store, manage, and update data within the application, which in turn allows for dynamic changes to user interfaces. For example, if a button needs to change its text based on user input, then this would be done by updating the state with new values.

STATE IN CLASS BASED COMPONENT

/\* eslint-disable no-unused-vars \*/

import React, { Component } from 'react'

export default class App extends Component {

    constructor()

    {

        super();

        this.state={

            name:"monika",

        }

    }

  render() {

    return (

      <div>

        <h1> am class based Component</h1>

      </div>

    )

  }

}

STATE IN FUNCTIONAL BASED COMONENT(HOOKS)

/\* eslint-disable no-unused-vars \*/

import React from 'react'

const App = () => {

   //hooks

    let name=React.useState("monika");

  return (

    <div>

    </div>

  )

}

export default App

UPDATE THE STATE PROPERTY

We cannot update the state property directly

setState(object) is a method to update the state property and it is asynchronous not update immediately instead it may dfer the update until later for better performance.

The setState() MethodState can be updated in response to event handlers, server responses, or prop changes. This is done using the setState() method. The setState() method enqueues all of the updates made to the component state and instructs React to re-render the component and its children with the updated state.

setState() allows us to change state in React class component

import  { Component } from 'react'

export default class App extends Component {

    state={

        username:"monika",

    }

  render() {

    return (

      <div>

        <h1>My name is {this.state.username}</h1>

       {/\* Do not mutate state directly. Use setState().eslintreact/no-direct-mutation-state\*/

       /\* setstate(object)\*/}

        <button onClick={()=>{this.setState({username:"moni"})}}>update</button>

      </div>

    )

  }

}

Usestate hook is the basic hook in fbc to provide statefull object in fbc.

Usestate is the react hook that allows us to add state to the fbc and returns an array with two values (intial state )and (function to update intial state)

Usestate=>[undefined,function(){}]

/\* eslint-disable no-unused-vars \*/

import {useState} from 'react'

const App = () => {

   //array destructing

    let[name,changename]=useState("monika");

  return (

    <div>

      <h1>My name is {name}</h1>

      <button onClick={()=>changename("moni")}>update</button>

    </div>

  )

}

export default App

usestate with object

/\* eslint-disable no-unused-vars \*/

import {useState} from 'react'

const App = () => {

    let[user,setuser]=useState({username:"monika"})

  return (

    <div>

     <h1>{user.username}</h1>

     <button onClick={()=>setuser({username:"moni"})}>Update</button>

    </div>

  )

}

export default App

import  { Component, Fragment } from 'react'

export default class App extends Component {

    state={

        languages:["java","js","sql","html"]}

  render() {

    return (

      <div>

        {this.state.languages.map((val,index) =>

        {

            return (<Fragment key={index}>

            <li>{val}</li>

            </Fragment>);

        })}

      </div>

    )

  }

}

States as Boolean

import  { Component } from 'react'

export default class App extends Component {

    state={

        isClickable:false

      }

  render() {

    if(this.state.isClickable === true)

    {

return <>

  <p>I am clickable</p>

</>

    }

    else

    {

      return <>

      <p>I am not  clickable</p>

    </>

    }

  }

}

//output: I am not clickable

States as Boolean in functional based component

import React, { useState } from 'react'

const App = () => {

  let[isClickable,changeClickable]=useState(false);

  if(isClickable === true)

  {

    return <>

      <h1>I am clickable</h1>

    </>

  }

  else

  {

return <>

  <h1>I am not Clickable</h1>

</>

  }

}

export default App

# //output: I am not Clickable

States as an object in functional

import  { Fragment, useState } from 'react'

const App = () => {

  let[data,setData]=useState(

{name:"monika",

 age:23,

 skills:["java","react","sql"],

 isEmployee:true,

})

  return (

    <div>

      <h1>{data.name}</h1>

      <h1>{data.age}</h1>

      <h1>My Skills:{data.skills.map((sub,index)=>{

        return <Fragment key={index}>

        <li>{sub}</li>

       </Fragment>

      })}</h1>

      <h1>{data.isEmployee === true? "Employee":"Not an Employee"}</h1>

    </div>

  )

}

export default App

//output:

# monika

# 23

# My Skills:

#  java

#  react

#  sql

# Employee

States as an Object in class based

/\* eslint-disable react/jsx-no-undef \*/

import { Component } from 'react'

import { Fragment } from 'react'

export default class App extends Component {

 state= {name:"monika",

 age:23,

 skills:["java","react","sql"],

 isEmployee:true,

}

  render() {

    return (

      <div>

        <h1>{this.state.name}</h1>

        <h1>{this.state.age}</h1>

        <h1>My Skills:{this.state.skills.map((sub,index)=>{

          return <Fragment key={index}>

          <li>{sub}</li>

          </Fragment>

        })}</h1>

        <h1>{this.state.isEmployee === true? "Employee":"Not an Employee"}</h1>

      </div>

    )

  }

}

//ouput:

# monika

# 23

# My Skills:

#  java

#  react

#  sql

# Employee

//array object in fbc

/\* eslint-disable no-unused-vars \*/

import  { Component, Fragment, useState } from 'react'

import { ReactDOM } from 'react'

const  App =()=>  {

  let [state,setState]=useState({

    courses:[{

      course\_id:1,

      course\_name:"mernstack",

      duration:4,

      trainers:["shashi","dixith","diwakar"],

      price:45000,

      branch:"BTM",

      course\_details:[{

        frontend:["html","css","js"],

        backend:["nodesjs","mysql"],

        database:["github","mangodb","mysql"],

      }]

    },{

      course\_id:2,

      course\_name:"javaFullstack",

      duration:4,

      trainers:["dixith","diwakar"],

      price:49000,

      branch:"HSR",

      course\_details:[{

        frontend:["html","css","js"],

        backend:["sprinboot","mysql"],

        database:["mysql","mysql"],

      }]

    },{

      course\_id:3,

      course\_name:"pythonfullstack",

      duration:8,

      trainers:["shashi","dixith"],

      price:50000,

      branch:"BTR",

      course\_details:[{

        frontend:["html","css","js"],

        backend:["Django","mysql"],

        database:["github","sql","mysql"],

      }]

    }]

  })

    return <section id="courseblock">

      <article>

        <main>

          <header>

            <h2>List of courses</h2>

          </header>

          <aside className='container'>

            {

             state.courses.map(course=>{

                return(

                  <div className='list' key={course.id}>

                  <h1>{course.course\_name}</h1>

                  <p><span>Branch: </span>

                  <span>{course.branch}</span></p>

                  <p><span>Duration: </span>

                  <span>{course.duration + " months"}</span></p>

                  <p><span>Price: </span>

                  <span>&#8377;{course.price}</span></p>

                  <p><span>Trainers </span>

                  <div className='trainer'>{course.trainers.map((trainer,index)=>

                  {

                   return <Fragment key={index}>

                   <p>{trainer}</p>

                   </Fragment>

                  })}

                  </div></p>

                  </div>

                )

              })

            }

          </aside>

        </main>

      </article>

    </section>

}

export default App;

//json

[

    {

      "course\_id": 1,

      "course\_name": "mernstack",

      "duration": 4,

      "trainers": [

        "shashi",

        "dixith",

        "diwakar"

      ],

      "price": 45000,

      "branch": "BTM",

      "course\_details": [

        {

          "frontend": [

            "html",

            "css",

            "js"

          ],

          "backend": [

            "nodesjs",

            "mysql"

          ],

          "database": [

            "github",

            "mangodb",

            "mysql"

          ]

        }

      ]

    },

    {

      "course\_id": 2,

      "course\_name": "javaFullstack",

      "duration": 4,

      "trainers": [

        "dixith",

        "diwakar"

      ],

      "price": 49000,

      "branch": "HSR",

      "course\_details": [

        {

          "frontend": [

            "html",

            "css",

            "js"

          ],

          "backend": [

            "sprinboot",

            "mysql"

          ],

          "database": [

            "mysql",

            "mysql"

          ]

        }

      ]

    },

    {

      "course\_id": 3,

      "course\_name": "pythonfullstack",

      "duration": 8,

      "trainers": [

        "shashi",

        "dixith"

      ],

      "price": 50000,

      "branch": "BTR",

      "course\_details": [

        {

          "frontend": [

            "html",

            "css",

            "js"

          ],

          "backend": [

            "Django",

            "mysql"

          ],

          "database": [

            "github",

            "sql",

            "mysql"

          ]

        }

      ]

    }

  ]

//fbc

/\* eslint-disable no-unused-vars \*/

import  { Component, Fragment, useState } from 'react'

import { ReactDOM } from 'react';

import JSON from "./data.json"

const  App =()=>  {

  let [state,setState]=useState({

  courses:JSON,

  });

    return <section id="courseblock">

      <article>

        <main>

          <header>

            <h2>List of courses</h2>

          </header>

          <aside className='container'>

            {

             state.courses.map(course=>{

                return(

                  <div className='list' key={course.id}>

                  <h1>{course.course\_name}</h1>

                  <p><span>Branch: </span>

                  <span>{course.branch}</span></p>

                  <p><span>Duration: </span>

                  <span>{course.duration + " months"}</span></p>

                  <p><span>Price: </span>

                  <span>&#8377;{course.price}</span></p>

                  <p><span>Trainers </span>

                  <div className='trainer'>{course.trainers.map((trainer,index)=>

                  {

                   return <Fragment key={index}>

                   <p>{trainer}</p>

                   </Fragment>

                  })}

                  </div></p>

                  </div>

                )

              })

            }

          </aside>

        </main>

      </article>

    </section>

}

export default App;

//css

\*{

    box-sizing: border-box;

    padding: 0;

    margin:0;

}

body{

    background-color: rgb(45, 20, 78);

    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

    color: white;

}

#courseblock

{

    background-color: rgb(64, 26, 168);

}

#courseblock article{

    background-color: rgb(64, 26, 168);

    height:100%;

    margin:0 auto;

    padding:10px;

    display: flex;

    width:90%

}

#courseblock article main{

    width:100%;

}

.container{

    width:100%;

    display: flex;

    flex-wrap:wrap;

    gap:10px;

    height:100%;

    padding: 10px;

}

.container .list{

    flex-basis: 32%;

    background-color: rgb(54, 19, 78);

}

#courseblock header h2{

    color: yellow;

    font-weight: bold;

    padding:10px 0;

    text-transform: uppercase;

}

.list h1{

color:rgb(238, 42, 7);

font-weight: 100;

text-transform: capitalize;

border-bottom:2px solid grey;

padding-bottom: 10px;

margin: 5px 0;

}

.list p{

    color: grey;

    font-size:14px;

    border-bottom: 1px solid rgb(2, 32, 5);

    padding:10px 0;

}

.trainer{

    display: flex;

    gap:5px

}

.trainer >p{

    background: rgb(206, 34, 140);

    font-size: 11px;

    color: black;

    padding:4px 10px;

    margin:4px 0;

    border-radius: 5px;

    border-bottom: 1px solid yellow;

    text-transform: capitalize

    ;

}

//increment app cbc

/\* eslint-disable no-undef \*/

import { Component } from 'react'

export default class App extends Component {

  state={

    count: 0,

    name:"monika",

    loading:false,

  };

  Increment=() =>

  {

    this.setState({ count:this.state.count + 1 ,loading:true,name:"kavi"})

  };

  Decrement=() =>

  {

    this.setState({ count:this.state.count - 1 ,loading:true,name:"priya"})

  };

 Reset=() =>

  {

    this.setState({ count: 0 ,loading:true,name:"mani"})

  };

  render() {

    return (

      <div>

      <p>{this.state.name}</p>

        <h1>{this.state.count}</h1>

        <div className='button'>

        <button onClick={this.Increment}>Increment</button>

        <button onClick={this.Decrement}>Decrement</button>

        <button onClick={this.Reset}>Reset</button>

      </div>

      </div>

    )

  }

}

//increment app fbc

/\* eslint-disable no-undef \*/

import { useState } from 'react'

const App=()=>{

  let[state,setState]=useState( {

    count: 0,

    name:"monika",

    loading:false,

  })

  let Increment=() =>

  {

    setState({ count:state.count + 1 ,loading:true,name:"kavi"})

  };

 let  Decrement=() =>

  {

   setState({ count:state.count - 1 ,loading:true,name:"priya"})

  };

 let

 Reset=() =>

  {

   setState({ count: 0 ,loading:true,name:"mani"})

  };

    return (

      <div>

      <p>{state.name}</p>

        <h1>{state.count}</h1>

        <div className='button'>

        <button onClick={Increment}>Increment</button>

        <button onClick={Decrement}>Decrement</button>

        <button onClick={Reset}>Reset</button>

      </div>

      </div>

    )

}

export default App;

//add to cart in fbc

import { useState } from 'react'

const App = () => {

  let[cart,setCart]=useState(0);

  let addToCart=()=>

  {

    setCart((prevalue)=>prevalue+200)//200

    setCart((prevalue)=>prevalue+100)//200+100

    setCart((prevalue)=>prevalue+100)//300+100=400

  }

  return (

    <div>

      <h1>Cart:{cart}</h1>

      <button onClick={addToCart}>Add to cart</button>

    </div>

  )

}

export default App

Lifecycle methods

Why?

React is an component based architecture enable us to reusable and interative UI

Phases

React Component

Error Handling

Unmounting(death of component)

Updating

Important methods:

* getDerivedStateFromProps
* shouldComponentUpdate
* getSnapShotBeforeUpdate
* componentDidUpdate
* Render-required

Mounting

(birth of component)-create component and insert into dom

Important methods:

* Constructor
* getDerivedStateFromProps
* Render-required
* ComponentDidMount

Constructor:

/\* eslint-disable no-unused-vars \*/

import  { Component } from 'react'

export default class App extends Component {

  constructor()

  {

    super();

    console.log("I am constructor and render only once")

    console.log("best place to intialize state object and bind this keyword")

  }

  static getDerivedStateFromProps(props,state)

  {

  console.log("rare used method")

    console.log("I am getDerivedStateFromProps execute after constructor and before render method amd renndering again and again based on sate and props")

  }

  componentDidMount()

  {

    console.log("I am componentDidMount i will render only once after all mounting life cycle ")

    console.log("best place for calling network call or ajax call or interact with dom")

  }

  render() {

    console.log("I am render method and render again and again")

    console.log("best place for rendering jsx and compose child component")

    return (

      <div>

        App

      </div>

    )

  }

}

//shouldComponentUpdate

Returns Boolean==if component update returns true and render method will execute after execution of shouldComponentUpdate

if component update returns false and render method will not execute after execution of shouldComponentUpdate

import  { Component } from 'react'

export default class App extends Component {

  state={

    count:0,

  }

 handleClick=()=>

  {

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

  }

shouldComponentUpdate(nextProp,nextState)

{

  console.log("I am shouldComponentUpdate");

  let x=this.state.count ===nextState.count;

  return x;

}

  render() {

    console.log("I am render method and render again and again")

    console.log("best place for rendering jsx and compose child component")

    return (

      <div>

       <h1>Count:{this.state.count}</h1>

       <button onClick={this.handleClick}>Click here</button>

      </div>

    )

  }

}

//getSnapShotBeforeUpdate

import  { Component } from 'react'

export default class App extends Component {

  state={

    count:0,

    course:"java"

  }

handleClick=()=>

  {

    this.setState({course:"Nodejs"})

  }

getSnapshotBeforeUpdate(prevProps,prevState)

{

let test=(document.getElementById("demo").innerHTML=`Before the Update ${prevState.course}`)

return test;

}

componentDidUpdate()

{

  let test=(document.getElementById("demo1").innerHTML=`Update value ${this.state.course}`)

  return test;

}

  render() {

    console.log("I am render method and render again and again")

    console.log("best place for rendering jsx and compose child component")

    return (

      <div>

       <button onClick={this.handleClick}>Click here <strong>{this.state.course}</strong></button>

       <h1><b>{this.state.course}</b></h1>

       <aside id="demo"></aside>

       <aside id="demo1"></aside>

      </div>

    )

  }

}

Events in react

React events are written in camelCase syntax:

onClick instead of onclick.

React event handlers are written inside curly braces:

onClick={shoot}  instead of onClick="shoot()".

In dom we handle default implicitly but in react we handle explicitly.

import React from 'react'

const App = () => {

 const  handleEventsInReact=()=>

  {

alert("Hi this is React Event")

  }

  return (

    <div>

     <h1>React js</h1>

     <button onClick={handleEventsInReact}>Click</button>

    </div>

  )

}

export default App

Default behavior of dom is page refreshing automatically

to prevent that in inline

handle()=>{

console.log(“hello”)

}

<form onsumbit=”handle(); return false”></form>🡪prevent behaviuour

<form onsumbit=”handle(); return true”></form>🡪default behaviuour

This type of syntax we cannot achieve in React.

In react you must call preventDefault explicitly to prevent browser default

/\* eslint-disable no-unused-vars \*/

import React from 'react'

const App = () => {

 const  handleSubmit=(e)=>

  {

console.log(e);//synthetic event

    e.preventDefault();//you must call preventDefault explicitly to prevent browser default

console.log("Hi this is React Event");

  }

  return (

    <div>

    <form onSubmit={handleSubmit}>

      <input type="text" placeholder="Enter name"></input>

      <button>submit</button>

    </form>

    </div>

  )

}

export default App

Handling events with React elements is very similar to handling events on DOM elements. There are some syntax differences:

* React events are named using camelCase, rather than lowercase.
* With JSX you pass a function as the event handler, rather than a string.

For example, the HTML:

<button onclick="activateLasers()">

Activate Lasers

</button>

is slightly different in React:

<button onClick={activateLasers}> Activate Lasers

</button>

Another difference is that you cannot return false to prevent default behavior in React. You must call preventDefault explicitly. For example, with plain HTML, to prevent the default form behavior of submitting, you can write:

<form onsubmit="console.log('You clicked submit.'); return false">

<button type="submit">Submit</button>

</form>

In React, this could instead be:

function Form() {

function handleSubmit(e) {

e.preventDefault(); console.log('You clicked submit.');

}

return (

<form onSubmit={handleSubmit}>

<button type="submit">Submit</button>

</form>

);

}

SyntheticBase events are similar to dom events but in dom we write code explicitly for browser compatibility but synthetic events default taken care of browser compatibility

//Synthetic Event

e is a synthetic event. React defines these synthetic events according to the [W3C spec](https://www.w3.org/TR/DOM-Level-3-Events/), so you don’t need to worry about cross-browser compatibility. React events do not work exactly the same as native events. See the [SyntheticEvent](https://legacy.reactjs.org/docs/events.html) reference guide to learn more.

When using React, you generally don’t need to call addEventListener to add listeners to a DOM element after it is created. Instead, just provide a listener when the element is initially rendered.

Your event handlers will be passed instances of SyntheticEvent, a cross-browser wrapper around the browser’s native event. It has the same interface as the browser’s native event, including stopPropagation() and preventDefault(), except the events work identically across all browsers.

Objectreference

Call();

let object1=

{

    obj\_id:5,

    obj\_name:"object1",

}

let product=

{

    pro\_id:6,

    pro\_name:"watch",

}

let person=

{

    obj\_id:70,

    obj\_name:"monika",

}

let win=window;

function callAnyObjectWhenEverYouNeeded()

{

    return this;

}

let result=callAnyObjectWhenEverYouNeeded();

console.log(result);

output: window {window: Window, self: Window, document: document, name: '', location: Location, …}

apply();

let object1=

{

    obj\_id:5,

    obj\_name:"object1",

}

let product=

{

    pro\_id:6,

    pro\_name:"watch",

}

let person=

{

    obj\_id:70,

    obj\_name:"monika",

}

let win=window;

function callAnyObjectWhenEverYouNeeded()

{

    return this;

}

let result=callAnyObjectWhenEverYouNeeded.call(person);

console.log(result);

//output: {obj\_id: 70, obj\_name: 'monika'}

Eg:2

let data1={

    num:200,

}

let data2={

    num:2000,

}

function addNumber(x,y,z)

{

    return this.num + x + y + z;

}

let result=addNumber.apply(data2,[10,30,30]);

console.log(result)

//output: 2070

//bind();returns function

let data1={

    num:200,

}

let data2={

    num:2000,

}

function addNumber(x,y,z)

{

    return this.num + x + y + z;

}

// let result=addNumber.apply(data2,[10,30,30]);

// console.log(result)

let result=addNumber.bind(data2,10,30,30);

console.log(result());

//

binding cbc

import  { Component } from 'react'

export default class App extends Component {

  constructor()

  {

    super();

    this.state={

      count:0

    }

      this.Increment=this.Increment.bind(this);//binding this keyword

  }

  Increment()

  {

    console.log(this);//window

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

  }

  Decrement=()=>//! recommended way

  {

    console.log(this)//window

   this.setState({count:this.state.count -1})

  }

  render() {

    return (

      <div>

        <h1>{this.state.count}</h1>

        <button onClick={this.Increment}>Increment with constructor bind</button>

        {/\* 2nd way using arrow function \*/}

        <button onClick={()=>{this.setState({count:this.state.count +1})}}>Increment with arrow function</button>

        <button onClick={this.Decrement}>Decrement with arrow function and recommended way </button>

        <button onClick={this.Increment.bind(this)}>Increment with inline bind</button>

      </div>

    )

  }

}

Custom Events in js

Syntax

Const even=new Event()

Eg:

//html

<button id=”btn”></button>

//js

const qspEvent =new Event(“qspEvent”)

//listening event

let btn=document.querySeelector(“#btn”);

btn.addEventListener(“qspEvent”,e=>{console.log(e)

console.log(e.target)});

//dispatch the event

btn.dispatchEvent(qspEvent);

//custom event in react

const App = () => {

  let customEvent=()=>

  {

    let color=new Event("red");

    let body=document.body;

    body.addEventListener("red",e=>{

      e.target.style.background = "red";

      console.log(e.target)

    });

    body.dispatchEvent(color);

  }

  return (

    <div>

      <button onClick={customEvent}>click</button>

    </div>

  )

}

export default App

const App = () => {

let[display,setDisplay]=useState(false);

  return (

    <div>

      <button onMouseEnter={()=>{setDisplay(true)}} onMouseLeave={()=>{setDisplay(false)}}>hover me get something</button>

    {display && (<div><h1>❤</h1></div>)}

    </div>

  )

}

export default App

Execute multiple function in oneclick

const App = () => {

  let show=()=>

  {

    console.log("show")

  }

  let showMenu=()=>

  {

    console.log("showMenu")

  }

  let showLogo=()=>

  {

    console.log("showLogo")

  }

  return (

    <div>

      <button onClick={()=>{show(),showMenu(),showLogo()}}>Multiple use</button>

    </div>

  )

}

export default App

Event Handler Props

./app.jsx

import React, { useState } from 'react'

import Button from './Button'

const App = () => {

    return (

    <div>

      <button onClick={()=>alert("This is normal event handler")}>Click me</button>

      <span onClick={()=>alert("This is normal event handler with span")} >Click me with span</span>

      <Button onTest={()=> alert("This is Event handler props")}>Play</Button>

      <Button onTest={()=> alert("This is Event handler props ")}>Play</Button>

    </div>

  )

}

export default App

./Button.jsx

const Button = ({children,onTest}) => {

  return (

    <button onClick={onTest}>{children}</button>

  )

}

export default Button

//Event propagation

./app.jsx—bubbling phase

import React from 'react'

import "./style.css"

const App = () => {

  return (

    <section  className="section"onClick={()=>console.log("This is Grandparent")}>Grandparent

      <div  className="parent"onClick={()=>console.log("This is Parent")}>Parent

        <div className='child' onClick={()=>console.log("This is Child")}>

Child

        </div>

      </div>

    </section>

  )

}

export default App

//stop event propagation

import React from 'react'

import "./style.css"

const App = () => {

  return (

    <section  className="section"onClick={(e)=>{e.stopPropagation(); console.log("This is Grandparent") }}>Grandparent

      <div  className="parent"onClick={(e)=>{e.stopPropagation();console.log("This is Parent")}}>Parent

        <div className='child' onClick={(e)=>{e.stopPropagation();console.log("This is Child")}}>

Child

        </div>

      </div>

    </section>

  )

}

export default App

style.css

\*{

    box-sizing: border-box;

    padding: 0;

    margin:0;

}

 body{

    background-color: rgb(45, 20, 78);

    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

    color: white;

 }

 .section{

    background-color: orangered;

    width:80%;

    height:100px;

    margin:10px auto;

 }

.parent{

    background-color:blue;

    width:80%;

    height:100px;

    margin:10px auto;

}

.child{

    background-color:rgb(25, 160, 54);

    width:80%;

    height:100px;

    margin:10px auto;

}

//Event propagation- capturing Phase

import React from 'react'

import "./style.css"

const App = () => {

  return (

    <section  className="section" onClickCapture={(e)=> console.log("This is Grandparent") }>Grandparent

      <div  className="parent" onClickCapture={(e)=>console.log("This is Parent")}>Parent

        <div className='child' onClickCapture={(e)=>console.log("This is Child")}>

Child

        </div>

      </div>

    </section>

  )

}

export default App

//stop propagation

import React from 'react'

import "./style.css"

const App = () => {

  return (

    <section  className="section" onClickCapture={(e)=> { e.stopPropagation();console.log("This is Grandparent")} }>Grandparent

      <div  className="parent" onClickCapture={(e)=>{ e.stopPropagation(); console.log("This is Parent")}}>Parent

        <div className='child' onClick={(e)=>{e.stopPropagation(); console.log("This is Child")}}>

Child

        </div>

      </div>

    </section>

  )

}

export default App

//prevent form event propagation

import React from 'react'

import "./style.css"

const App = () => {

  return (

    <form onSubmit={(e)=>{ e.preventDefault();console.log("Hello form")}}

    >

      <input type="text" placeholder='Enter name'></input>

      <button>Submit</button>

    </form>

  )

}

export default App

key Event

import React from 'react'

import "./style.css"

const App = () => {

  let handleKeyboardEvent=(e)=>

  {

    // if(e.key === "Enter")

    // {

    //   console.log(`you have been pressed ${e.key}`)

    // }

    // if(e.keyCode === 13)

    // {

    //   console.log(`you have been pressed ${e.keyCode}`)

    // }

  //   if(e.keyCode === 32)

  //  {

  //    console.log(`you have been pressed ${e.keyCode}`)

  //  }

  if(e.key === "a")

    {

      console.log(`you have been pressed ${e.key}`)

  }

  else if(e.key === "b")

    {

      console.log(`you have been pressed ${e.key}`)

  }

  else

  {

    console.log(`some other key`)

}

  }

  return (

    <div>

   <input type="text"

    // onKeyPress={handleKeyboardEvent}

    // onKeyDown={handleKeyboardEvent}

    onKeyUp={handleKeyboardEvent}

    >

    </input>

    </div>

  )

}

export default App

//keyboard press

import React from 'react'

import "./style.css"

import {useState} from 'react'

const App = () => {

let [state, setState] = useState("");

let handleKeyboardEvent=(e)=>

  {

  if(e.key === "a")

    {

      console.log(`you have been pressed ${e.key}`)

      setState(e.key)

  }

  else if(e.key === "b")

    {

      console.log(`you have been pressed ${e.key}`)

      setState(e.key)

  }

  else

  {

    console.log(`some other key`)

    setState(e.key)

}

  }

  return (

    <div>

    <h1>{state}</h1>

   <input type="text" value={state} onKeyUp={handleKeyboardEvent}>

    </input>

    </div>

  )

}

export default App

Conditional Rendering:

Rendering the UI based on conditions.

Else if :

import React, { useState } from 'react'

const App = () => {

  const [mode,setMode]=useState("a");

  const [color,setColor]=useState("green")

  if(mode === "a" && color==="green")

  {

    return <h1 style={{color:color}}>Mode is A</h1>

  }

  else if(mode === "b" && color==="red")

  {

    return <h1 style={{color:color}} >Mode is B</h1>

  }

  else if(mode ==="c" && color==="blue")

  {

    return <h1 style={{color:color}}>Mode is C</h1>

  }

  else

  {

    return <h1>No Matching Keyboard</h1>

  }

}

export default App

immediate invoke function

//app.jsx

/\* eslint-disable no-unused-vars \*/

import { useState } from "react"

import Notifications from "./Components/Messages/Notifications"

const App = () => {

  let [state,setState]=useState([

    {

      text:"info text",

      status:"info",

    },

    {

      text:"warning text",

      status:"warning",

    },

    {

      text:"error text",

      status:"error",

    }

  ])

  return (

    <div>

     <Notifications  state={state}/>

    </div>

  )

}

export default App

//Notifications.jsx

/\* eslint-disable no-unused-vars \*/

/\* eslint-disable react/prop-types \*/

import { Fragment } from "react"

import Info from "./Info"

const Notifications = ({state}) => {

  return (

    <div>

     {

        state.map((data,index)=>{

            return <Info key={index} {...data} />

        })

     }

    </div>

  )

}

export default Notifications

//Info.jsx

/\* eslint-disable no-unused-vars \*/

/\* eslint-disable react/prop-types \*/

const Info = ({text,status}) =>{

return (

<div>

   {

    (function()

    {

        switch(status)

        {

            case "info":

            return text;

            case "warning":

            return text;

            case "error":

            return text;

            default:

            return "no status found";

        }

    })()

   }

</div>

)

}

export default Info;

shor-circuit AND operator

display content or nothing

//prevent null mapping

import { Fragment } from "react";

import { useState } from "react";

const App = () => {

  let[data,setData]=useState(null);

  (async function()

  {

    let res=await window.fetch("https://api.github.com/users");

    let finalVal= await res.json();

    setData(finalVal)

  })();

  return (

    <div>

      {

        data === null ? "Loading...":data.map(val=>{

          return(<Fragment key={val.id}>

           <li>{val.login}</li>

          </Fragment>)

        })

      }

    </div>

  )

}

export default App

React Refs

Creating ref

import {createRef} from 'react'

const App = () => {

let inputRef=createRef();

let passwordRef=createRef();

let changeData=(e)=>

{

e.preventDefault();

let username=inputRef.current.value;

let password=passwordRef.current.value;

console.log(username);

console.log(password);

}

console.log(inputRef)

  return (

  <form>

    <input  ref={inputRef} type="text" placeholder='Enter username'></input>

    <input ref={passwordRef} type="password" placeholder='Enter password'></input>

    <button onClick={changeData}>Submit</button>

  </form>

  )

}

export default App

useRef in fdc

import {useRef} from 'react'

const App = () => {

let inputRef=useRef();

let changeData=()=>

{

  inputRef.current.placeholder="Enter username";

inputRef.current.style.background="#efefef";

}

console.log(inputRef)

  return (

  <div>

    <input  ref={inputRef} type="text" onClick={changeData}></input>

  </div>

  )

}

export default App

**React forms**

This form has the default HTML form behavior of browsing to a new page when the user submits the form. If you want this behavior in React, it just works. But in most cases, it’s convenient to have a JavaScript function that handles the submission of the form and has access to the data that the user entered into the form. The standard way to achieve this is with a technique called “controlled components”.

In React, there are two ways of handling form data:

* **Controlled Components:** In this approach, form data is handled by React through the use of hooks such as the useState hook.
* **Uncontrolled Components:** Form data is handled by the Document Object Model (DOM) rather than by React. The DOM maintains the state of form data and updates it based on user input.
* In the first approach, you let the bike take control. You sit on the bike and let it decide the direction and speed. You might try to make it go in a certain direction by leaning your body, but ultimately, the bike decides where to go.
* This is similar to uncontrolled components in React. You place a form element in the component, and the DOM takes control of it. The DOM decides the state of the input element and updates it based on a user's input.
* In the second approach, you take control of the bike. You hold the handlebars and pedal, and you decide where to go and how fast to ride. You can easily slow down or speed up as needed.
* This is similar to controlled components where a React component takes control of the form data, and maintains the state of form elements. The component decides when and how to update the state, and it re-renders itself based on the state changes.

**uncontrolled state**

import {useRef} from 'react'

//uncontrolled component->react ref-->use ref attribute in input

const App = () => {

let emailRef=useRef();

let passwordRef=useRef();

let handleClick=e=>

{

e.preventDefault();

let email=emailRef.current.value;

let password=passwordRef.current.value;

console.log(email);

console.log(password);

}

  return (

    <section>

      <article>

        <div>

        <label htmlFor='email'>Email</label>

          <input type="email" placeholder='Enter email'id="email" ref={emailRef} />

        </div>

        <div>

        <label htmlFor='password'>Password</label>

          <input type="password" placeholder='Enter password'id="password" ref={passwordRef} />

        </div>

        <div>

        <button onClick={handleClick}>Login</button>

        </div>

      </article>

    </section>

  )

}

export default App

## Controlled Components in React

In React, a controlled component is a component where form elements derive their value from a React state.

When a component is controlled, the value of form elements is stored in a state, and any changes made to the value are immediately reflected in the state.

To create a controlled component, you need to use the value prop to set the value of form elements and the onChange event to handle changes made to the value.

The value prop sets the initial value of a form element, while the onChange event is triggered whenever the value of a form element changes. Inside the onChange event, you need to update the state with the new value using a state update function.

**change event**

The change event occurs when the value of an element has been changed (only works on <input>, <textarea> and <select> elements).

Controlled components

import {useState}from 'react'

//controlled components-->declarative

const App = () => {

  //step-1-->Intialize state object

  //step-2-->Add value Attribute to the input or form elements

  //step-3-->state mutation  or state updation //!add onchange event to the elements

  let[email,setEmail]=useState("");

  let[password,setPassword]=useState("");

  let handleSubmit=(e)=>

  {

    e.preventDefault();

    console.log({email,password})

  }

  return (

    <div>

      <form onSubmit={handleSubmit}>

        <input type="email" placeholder='Enter email' value={email} onChange={(e)=>setEmail(e.target.value)}></input>

        <input type="password" placeholder='Enter password' value={password} onChange={(e)=>setPassword(e.target.value)}></input>

        <button>Login</button>

      </form>

    </div>

  )

}

export default App

single onchange function in fbc

import {useState}from 'react'

//controlled components-->declarative

const App = () => {

  //step-1-->Intialize state object

  //step-2-->Add value Attribute to the input or form elements

  //step-3-->state mutation  or state updation //!add onchange event to the elements

 let[state,setState]=useState({

  email:"",

  password:"",

  loading:false,

 })

 let {email,password,loading}=state;

 let handleChange=(e)=>

 {

  let{name,value}=e.target;

  //explicitly merging the data-->...state

  setState({...state,[name]:value});

 }

  let handleSubmit=(e)=>

  {

    e.preventDefault();

    console.log({email,password})

  }

  return (

    <div>

      <form onSubmit={handleSubmit}>

        <input type="email" placeholder='Enter email' name="email" value={email} onChange={handleChange}></input>

        <input type="password" placeholder='Enter password'name="password" value={password} onChange={handleChange}></input>

        <button>{loading ?"loading":"submit"}</button>

      </form>

    </div>

  )

}

export default App

note:\*name attribute is mandatory

hoc:

//reusability

//composability

const Hoc = (WrappedComponent) => {

    let fun=function()

    {

        return <WrappedComponent/>

    }

    return fun;

}

export default Hoc;

//!hoc is like a function that takes component as a arguement and return enhanced version of the component

//example /

//hoc.jsx

const Hoc = (WrappedComponent) => {

   return function fun()

    {

        return <WrappedComponent profile={{name:"monika",age:21,company:"fireflink"}}/>;

    }

}

export default Hoc;

//!hoc is like a function that takes component as a arguement and return enhanced version of the component

//profile.jsx

/\* eslint-disable react/prop-types \*/

/\* eslint-disable react-refresh/only-export-components \*/

import Hoc from "../hoc/Hoc";

const Profile = ({profile}) => {

    console.log(profile);

  return (

  <div>

    <h1>{profile.name}</h1>

  </div>

  )

}

export default Hoc(Profile)

//Navbar.jsx

/\* eslint-disable react-refresh/only-export-components \*/

/\* eslint-disable react/prop-types \*/

import Hoc from "../hoc/Hoc"

const Navbar = ({profile}) => {

  return (

    <div>

    <h1>Navbar</h1>

      <h1>{profile.age} is coming from hoc</h1>

    </div>

  )

}

export default Hoc(Navbar)

//App.jsx

import Profile from './components/auth/Profile'

import Navbar from './components/navbar/Navbar'

const App = () => {

  return (

    <div>

      <Profile/>

      <Navbar/>

    </div>

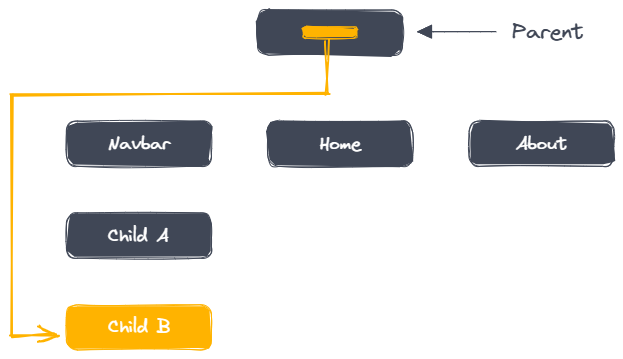
  )

}

export default App

Context Api

Context API allows data to be passed through a component tree without having to pass props manually at every level. This makes it easier to share data between components.

A diagram illustrating how Context API works

The React Context API is a way for a React app to effectively produce global variables that can be passed around. This is the alternative to "prop drilling" or moving props from grandparent to child to parent, and so on. Context is also touted as an easier, lighter approach to state management using Redux.

Context API is a (kind of) new feature added in version 16.3 of React that allows one to share state across the entire app (or part of it) lightly and with ease.

CONTEXT API

Provider

Context Type

displayname

createContext()

Global state

components

components

consumer

REACT API version 16.3 onwards

Step 1:create context

Step2:create provider

Step3:In provider use (value) prop

import { createContext } from "react";

console.log(createContext);

let myContext =createContext("");

console.log(myContext);

1. **$typeof**: Symbol(react.context)
2. **Consumer**: {$$typeof: Symbol(react.context), \_context: {…}, …}
3. **Provider**: {$$typeof: Symbol(react.provider), \_context: {…}}
4. **\_currentRenderer**: null
5. **\_currentRenderer2**: null
6. **\_currentValue**: ""
7. **\_currentValue2**: ""
8. **\_defaultValue**: null
9. **\_globalName**: null
10. **\_threadCount**: 0

1. //usecase of contextApi
2. // 1:theme design like dark or light
3. // 2:user Authentication
4. // 3:acess data from external resources
5. // 4:multilingual support

REFER FREECODECAMP

STYLED COMPONENTS

Css in js

Limitation of inline styles with js

* We cannot use pseudo selectors, media queries,key frames
* Library for css in js-styled components,emotion js,jss,Aphrodite,radium,styled jsx
* npm add styled-components

eg:

import styled from "styled-components"

console.log(styled);

const H1=styled.h1`

color:red;

font-size:20px;`

const H2=styled.h2`

color:yellow;

font-size:25px;`

const Form=styled.form`

width:500px;

background-color:#eee;

color:white;

padding:10px;

margin:20px;`

const Input=styled.input`

width:90%;

border:1px solid #efefef;

padding:10px;`

const Button=styled.button`

border:none;

background:purple;

color:white;

padding:10px;

margin:10px 0;`

const App = () => {

  return (

    <div >

      <H1>This is H1</H1>

      <H2>This is H2</H2>

      <Form>

        <Input type="text" placeholder="Enter name"></Input>

        <Button>View More</Button>

      </Form>

    </div>

  )

}

export default App

CSS FRAMEWORKS

Install tailwind css

**https://tailwindcss.com/docs/guides/vite**

npm install -D tailwindcss postcss autoprefixer

Install tailwind configuration

npx tailwindcss init –p

copy the tailwind.config.js and paste it in folder tailwind.config

/\*\* @type {import('tailwindcss').Config} \*/

export default {

  content: [

    "./src/\*\*/\*.{js,jsx,ts,tsx}",

  ],

  theme: {

    extend: {},

  },

  plugins: [],

}

**Use tailwind css**

**Copy**

@tailwind base;

@tailwind components;

@tailwind utilities;

And paste it in created root css file

Bootstrap

React bootstrap website

Material UI

https://mui.com/material-ui/getting-started/installation/

npm install @mui/material @emotion/react @emotion/styled

npm install @fontsource/roboto

npm install @mui/icons-material

**google web fonts**

<link rel="preconnect" href="https://fonts.googleapis.com" />

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />

<link

rel="stylesheet"

href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500;600;700&display=swap"

/>

**Paste in index.html**

**Usage**

import Button from '@mui/material/Button';

const App = () => {

  return (

    <div>

       <Button variant="contained">Hello world</Button>

    </div>

  )

}

export default App

**Ant design**

[**https://ant.design/**](https://ant.design/)

**UseEffect**

**React side-effect**

A React side-effect occurs when we use something that is outside the scope of React.js in our React components e.g. Web APIs like localStorage.

* **When we talk about side effects in the context of React.js, we are referring to anything that is outside the scope of React**
* **So calling any native Web APIs will be considered as a side effect as it’s not within the React universe**
* **Making a HTTPS request to an external API is another example of a side effect and the list goes on…**
* **We usually manage React side effects inside the useEffect hook (part of the React Hooks API)**

import { useEffect, useState } from "react"

const App = () => {

let[width,setWidth]=useState(window.innerWidth);

const updateWidth=()=>

{

  setWidth(window.innerWidth);

}

  useEffect(()=>{

    window.addEventListener('resize',updateWidth)//adding to the event stack

    //remove eventlistener

    //cleanup function

    return ()=>//removing from  the event stack

    {

      window.removeEventListener('resize',updateWidth)

    }

  },[width])

  console.log(width)

  if(width>600)

  {

    document.body.style.background="green";

  }

  else

  {

    document.body.style.background="red";

  }

  return (

    <div>

    </div>

  )

}

export default App

**to use Api’s**

**npm install axios-instead of window.fetch**

Axios is a promise-based HTTP library that lets developers make requests to either their own or a third-party server to fetch data. It offers different ways of making requests such as GET , POST , PUT/PATCH , and DELETE

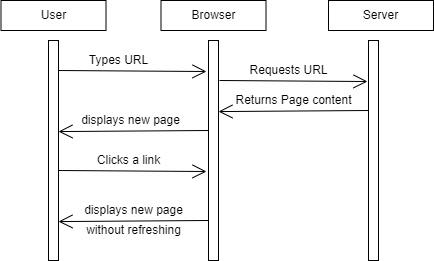
**REACT ROUTER**

[**https://reactrouter.com/en/main**](https://reactrouter.com/en/main)

CLIENT-SIDE ROUTING

Client-side routing is one of the most important features to make use of while developing single-page web applications. In web development, routing describes the process by which a user clicks a link on a webpage and is directed to a new page with different content and a unique URL

ROUTING



Includes everything in react router and adds a few react native APis

Includes everything in react router and adds a few dom specific APis

It contains the core functionality of react router

Route-matching alogorithm

React-router-native

React-router-dom

React-router

url

https://localhost5173/login

React router

# HTTP

**Hypertext Transfer Protocol (HTTP)** is an [application-layer](https://en.wikipedia.org/wiki/Application_Layer) protocol for transmitting hypermedia documents, such as HTML. It was designed for communication between web browsers and web servers, but it can also be used for other purposes. HTTP follows a classical [client-server model](https://en.wikipedia.org/wiki/Client%E2%80%93server_model), with a client opening a connection to make a request, then waiting until it receives a response. HTTP is a [stateless protocol](https://en.wikipedia.org/wiki/Stateless_protocol), meaning that the server does not keep any data (state) between two requests.

# HTTP response status codes

HTTP response status codes indicate whether a specific [HTTP](https://developer.mozilla.org/en-US/docs/Web/HTTP) request has been successfully completed. Responses are grouped in five classes:

1. [Informational responses](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#information_responses) (100 – 199)
2. [Successful responses](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#successful_responses) (200 – 299)
3. [Redirection messages](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#redirection_messages) (300 – 399)
4. [Client error responses](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#client_error_responses) (400 – 499)
5. [Server error responses](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#server_error_responses) (500 – 599)

npm add axios react-router-dom react-moment react-hot-toast react-icons uuid

**json server**

**npm install -g json-server**

**json-server –version**

**start backend servor**

**json-server --watch backend/db.json --port=5000**

[**https://blog.logrocket.com/**](https://blog.logrocket.com/)

# Optional chaining (?.)

The **optional chaining (?.)** operator accesses an object's property or calls a function. If the object accessed or function called using this operator is [undefined](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/undefined) or [null](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/null), the expression short circuits and evaluates to [undefined](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/undefined) instead of throwing an error.

**Advance hooks**

//hooks are javascript functions

**Rules:**

* **Only call hooks at the top level**
* **Don’t call hooks inside loops or conditions or nested function**
* **Don’t call hooks from regular javascript function.**
* **Instead call hooks from the react functions Component**
* **Call hooks from custom hooks**

**Hooks starts with ----use----**

**Rules of Hooks**

You are probably here because you got the following error message:

Console

X Hooks can only be called inside the body of a function component.

There are three common reasons you might be seeing it:

1. You might be **breaking the Rules of Hooks**.
2. You might have **mismatching versions** of React and React DOM.
3. You might have **more than one copy of React** in the same app.

## Breaking Rules of Hooks

Functions whose names start with use are called [Hooks](https://react.dev/reference/react) in React.

**Don’t call Hooks inside loops, conditions, or nested functions.** Instead, always use Hooks at the top level of your React function, before any early returns. You can only call Hooks while React is rendering a function component:

* ✅ Call them at the top level in the body of a [function component](https://react.dev/learn/your-first-component).
* ✅ Call them at the top level in the body of a [custom Hook](https://react.dev/learn/reusing-logic-with-custom-hooks).

function Counter() {

// ✅ Good: top-level in a function component

const [count, setCount] = useState(0);

// ...

}

function useWindowWidth() {

// ✅ Good: top-level in a custom Hook

const [width, setWidth] = useState(window.innerWidth);

// ...

}

* 🔴 Do not call Hooks inside conditions or loops.
* 🔴 Do not call Hooks after a conditional return statement.
* 🔴 Do not call Hooks in event handlers.
* 🔴 Do not call Hooks in class components.
* 🔴 Do not call Hooks inside functions passed to useMemo, useReducer, or useEffect.

function Bad({ cond }) {

if (cond) {

// 🔴 Bad: inside a condition (to fix, move it outside!)

const theme = useContext(ThemeContext);

}

// ...

}

function Bad() {

for (let i = 0; i < 10; i++) {

// 🔴 Bad: inside a loop (to fix, move it outside!)

const theme = useContext(ThemeContext);

}

// ...

}

function Bad({ cond }) {

if (cond) {

return;

}

// 🔴 Bad: after a conditional return (to fix, move it before the return!)

const theme = useContext(ThemeContext);

// ...

}

function Bad() {

function handleClick() {

// 🔴 Bad: inside an event handler (to fix, move it outside!)

const theme = useContext(ThemeContext);

}

// ...

}

function Bad() {

const style = useMemo(() => {

// 🔴 Bad: inside useMemo (to fix, move it outside!)

const theme = useContext(ThemeContext);

return createStyle(theme);

});

// ...

}

class Bad extends React.Component {

render() {

// 🔴 Bad: inside a class component (to fix, write a function component instead of a class!)

useEffect(() => {})

// ...

}

}

**Refer:** **https://react.dev/warnings/invalid-hook-call-warning**

**useReducer-when it is a complex state object go with useReducer**

**useReducer is the alternative to the useState hook**

**useState- when it is a basic state object go with useState**

**contextApi with useReducer is the best option to create complex state Logic in react application.**

**For state management like Redux,Recoil,Mobx useReducer is the best option compare to useState**

The useReducer Hook is used to store and update states, just like the useState Hook. It accepts a reducer function as its first parameter and the initial state as the second. useReducer returns an array that holds the current state value and a dispatch function to which you can pass an action and later invoke it.

import { useReducer } from "react"

const App = () => {

  let intialState={

    name:"Monika",

    company:"FireFlink",

    salary:20000,

  }

  let[state,dispatch]=useReducer(()=>{},intialState);

  return (

    <div>

    <h1>{state.name}</h1>

    <h1>{state.company}</h1>

    <h1>{state.salary}</h1>

    </div>

  )

}

export default App

import { useReducer } from "react"

const App = () => {

  let initialState={

   count:0,

  }

  let reducer=(state,action)=>

  {

console.log(state);

console.log(action);

// console.log(action.type);

switch(action.type)

{

  case "INCREMENT":

    return{

      count:state.count+1,

    }

    case "DECREMENT":

    return{

      count:state.count-1,

    }

    case "RESET":

    return{

      count:0,

    }

    default:

      return state;

}

  }

  let[state,dispatch]=useReducer(reducer,initialState);

  return (

    <div>

    <h1>{state.count}</h1>

    <button onClick={()=>dispatch({type:"INCREMENT"})}>Increment</button>

    <button onClick={()=>dispatch({type:"DECREMENT"})}>Decrement</button>

    <button onClick={()=>dispatch({type:"RESET"})}>Reset</button>

    </div>

  )

}

export default App

## What are authentication and authorization?

In simple terms, authentication is the process of verifying who a user is, while authorization is the process of verifying what they have access to.

| **Authentication** | **Authorization** |
| --- | --- |
| Determines whether users are who they claim to be | Determines what users can and cannot access |
| Challenges the user to validate credentials (for example, through passwords, answers to security questions, or facial recognition) | Verifies whether access is allowed through policies and rules |
| Usually done before authorization | Usually done after successful authentication |
| Generally, transmits info through an ID Token | Generally, transmits info through an Access Token |
| Generally governed by the OpenID Connect (OIDC) protocol | Generally governed by the OAuth 2.0 framework |
| Example: Employees in a company are required to authenticate through the network before accessing their company email | Example: After an employee successfully authenticates, the system determines what information the employees are allowed to access |

**Portals**

React Portals provide a way to render components into a DOM node outside the component hierarchy. Typically, when you render a component in React, it gets inserted into the DOM as a child of its parent component. However, there are scenarios where you might need to render a component at a different location in the DOM, such as when creating a modal or dropdown menu.

Helps to render the Component that are not a part of root component

Memorization

Memoization is an optimization technique for accelerating computer programs by caching the results of heavy function calls and returning them when similar inputs are encountered repeatedly. Simply, React memoization is similar to caching

In programming, **memoization is an optimization technique** that makes applications more efficient and hence faster. It does this by storing computation results in cache, and retrieving that same information from the cache the next time it's needed instead of computing it again.

In simpler words, it consists of storing in **cache** the output of a function, and making the function check if each required computation is in the cache before computing it.

A **cache** is simply a temporary data store that holds data so that future requests for that data can be served faster.

Using memo will cause React to skip rendering a component if its props have not changed.

**REACT MEMO**

Components in React are designed to re-render whenever the state or props value changes. However, this can impact your application's performance because, even if the change is only intended to affect the parent component, every other child component attached to the parent component will re-render. When a parent component re-renders, so do all of its child components.

**//App.jsx**

import Count from "./Count"

import { useState } from "react";

const App = () => {

  let [count,setCount]=useState(0);

  let Increment=()=>

  {

    setCount(count+1)

  }

  return (

    <div>

  <h1>I am parent Component{""} <strong style={{color:"green"}}>{count}</strong></h1>

  <button onClick={Increment}>Increment</button>

  <hr/>

  <Count/>

    </div>

  )

}

export default App;

**//Count.jsx**

import React from "react"

const Count = () => {

  return (

    <div>

    {console.log("I am child Component")}

        <h1>I am child Component</h1>

    </div>

  )

}

export default React.memo(Count)

React useMemo() hook is a function that caches the value produced from an expensive function used inside a React component. It accepts the expensive function and works by storing the value produced from the function when that is passed the same arguments repeatedly.

useMemo(()=>{},[])

Error boundaries are react component that catch javascript eroors anywhere in their child component tree and log those errors and display fallback ui instead of the component tree that crashed

Error boundaries catch errors during rendering during lifecycle methods

Error boundaries do not catch error for :

//event handlers

//asynchronous code like setTimeout

//server side render

Lazy Loading..

**Lazy loading**(also called on-demand loading) is an optimization technique for the online content, be it a website or a web app.  
Instead of loading the entire web page and rendering it to the user in one go as in bulk loading, the concept of lazy loading assists in loading only the required section and delays the remaining, until it is needed by the user.

Code splitting:

Javascript,css and html can be split into smaller chunks this enables sending of minimal code required to the server

Improving page load time

Lazy loading..on demand loading or user required loading(dynamic loading)

Const Login=lazy(()=>import(“./Login.jsx”)

Eg: import { lazy } from 'react';

const MarkdownPreview = lazy(() => import('./MarkdownPreview.js'));

<Suspense fallback={<Loading />}>

<h2>Preview</h2>

<MarkdownPreview />

</Suspense>

If we want pass ref from one compoennet to another component we need forwardRef

In child export React.forwardRef(child)

Destructure ref

And pass to element ref={ref}

REDUX

**actions**

Comp2

Comp1

data

1)create store

import { configureStore } from "@reduxjs/toolkit";

export const store=configureStore({

    devTools:true,

    reducer:

    {

        customers:

    }

})

2)create slice

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

const initialState = [];

const customerSlice = createSlice({

    name: "customer",

    initialState,

    reducers: {

        },

3)create reducer function

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

const initialState = [];

const customerSlice = createSlice({

    name: "customer",

    initialState,

    reducers: {

addCustomer: (state, action) => {

            state.push(action.payload);

        },

4)Exporting Action creators and Reducer

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

const initialState = [];

const customerSlice = createSlice({

    name: "customer",

    initialState,

    reducers: {

        addCustomer: (state, action) => {

            state.push(action.payload);

        },

        deleteCustomer(state,action)

        {

            const deleteIndex=action.payload;

            return state.filter((val,index)=>index!==deleteIndex)

        }

    }

});

export const { addCustomer,deleteCustomer } = customerSlice.actions;

export default customerSlice.reducer;

5)Sharing Redux store using Provider

import CustomerAdd from "./Components/CustomerAdd"

import CustomerView from "./Components/CustomerView"

import { Provider } from "react-redux"

import { store } from './Store';

const App = () => {

  return (

    <Provider store={store}>

      <div>

      <CustomerAdd/>

      <CustomerView/>

    </div>

    </Provider>

  )

}

export default App

6)Redux devtool extensions

https://chromewebstore.google.com/detail/redux-devtools/lmhkpmbekcpmknklioeibfkpmmfibljd

7)Dispatch actions

import { useState } from "react"

import {useDispatch } from "react-redux";

import { addCustomer } from "../slice/CustomerSlice";

const CustomerAdd = () => {

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

    const dispatch=useDispatch();

 function addCustomerData()

    {

        if(input)

        {

            dispatch(addCustomer(input))

            setInput("");

        }

    }

  return  <>

     <div>

        <h1>Add customer</h1>

        <input type="text"  value={input}onChange={(e)=>setInput(e.target.value)}/>

        <button onClick={addCustomerData}>Add</button>

    </div>

     </>

}

export default CustomerAdd

8)action and state changes

9)Getting data from store

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

const initialState = [];

const customerSlice = createSlice({

    name: "customer",

    initialState,

    reducers: {

        addCustomer: (state, action) => {

            state.push(action.payload);

        },

        deleteCustomer(state,action)

        {

            const deleteIndex=action.payload;

            return state.filter((val,index)=>index!==deleteIndex)

        }

    }

});

export const { addCustomer,deleteCustomer } = customerSlice.actions;

export default customerSlice.reducer;

import { useDispatch, useSelector } from "react-redux";

import { deleteCustomer } from "../slice/CustomerSlice";

/\* eslint-disable react/prop-types \*/

const CustomerView = () => {

   const customers=useSelector((state)=>state.customers)

  return (

    <div>

    <h1>Customer View</h1>

    <ul>

        {

            customers.map((customer,index)=>{

                return <>

                    <li>{customer}</li>

                </>

            })

        }

    </ul>

    </div>

  )

}

export default CustomerView

10)Creating reducer for delete action

import { useDispatch, useSelector } from "react-redux";

import { deleteCustomer } from "../slice/CustomerSlice";

/\* eslint-disable react/prop-types \*/

const CustomerView = () => {

    const dispatch=useDispatch();

   const customers=useSelector((state)=>state.customers)

   function deleteHandler(index)

   {

    dispatch( deleteCustomer(index))

   }

  return (

    <div>

    <h1>Customer View</h1>

    <ul>

        {

            customers.map((customer,index)=>{

                return <>

                    <li>{customer} <button onClick={()=>deleteHandler(index)}>delete</button></li>

                </>

            })

        }

    </ul>

    </div>

  )

}

export default CustomerView