Table of Contents

[REACTJS 2](#_Toc182935977)

[Introduction: 2](#_Toc182935978)

[**What is reactjs?** 2](#_Toc182935979)

[**Features** 2](#_Toc182935980)

[**COMMANDS IN REACT.JS** 2](#_Toc182935981)

[**What is Single Page Application(SPA)?** 2](#_Toc182935982)

[**When to Use React.js?** 2](#_Toc182935983)

[**What is jsx?** 3](#_Toc182935984)

[Components in Reactjs? 3](#_Toc182935985)

[**Functional Components:** 3](#_Toc182935986)

[**Class Components:** 4](#_Toc182935987)

[Import and Exports 4](#_Toc182935988)

[**Imports:** 5](#_Toc182935989)

[**Exports:** 5](#_Toc182935990)

[Types of Exports in Reactjs are: 5](#_Toc182935991)

[Named Exports : 5](#_Toc182935992)

[Default Exports: 6](#_Toc182935993)

[Props in ReactJs 7](#_Toc182935994)

[**By using Functional component**: 7](#_Toc182935995)

[MAP() METHOD 8](#_Toc182935996)

[How to use map( ) method with props in React 8](#_Toc182935997)

[Child props in React 10](#_Toc182935998)

[State() and useState() 11](#_Toc182935999)

[State() method used in class component 11](#_Toc182936000)

[useState() method is used in Functional Component 11](#_Toc182936001)

[Show and hide the details of bank by using useState() Method 12](#_Toc182936002)

[Back Ground Color Changer 14](#_Toc182936003)

[Counter App 16](#_Toc182936004)

[Props Destructuring 21](#_Toc182936005)

# **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