**What are Components in ReactJS?**

Earlier, the developers write more than thousands of lines of code for developing a single page application. These applications follow the traditional DOM structure, and making changes in them was a very challenging task. If any mistake found, it manually searches the entire application and update accordingly. The component-based approach was introduced to overcome an issue. In this approach, the entire application is divided into a small logical group of code, which is known as components.

the entire application is divided into a small logical group of code, which is known as components.

Components are like pure javascript functions that help make the code easy by splitting the logic into reusable independent code.

 A functional component is just a plain JavaScript function that returns JSX.

A class component is a JavaScript class that extends React.Component which has a render method.

In Components Folder 🡺 create a new File with the name of FcuntionalComponent.jsx

import React from "react";

/\* arrow function \*/

// const FunctionalComponent= () => {

//     return <h1> Hello world from Functional Component with arrow fucntion</h1>

// };

/\* Javascript Function \*/

function FunctionalComponent()

{

    return <h1> Hello world from Functional Component with Javascript fucntion</h1>

}

export default FunctionalComponent;

app.js

import FunctionalComponent from './Components/FcuntionalComponent';

inside the return Method

<h2> Example for Fucntional Component</h2>

    <p> functional component is just a plain JavaScript function that returns JSX.</p>

    <FunctionalComponent />

Functional component with Properties

FunctionalComponents.jsx

import React from "react";

import { ReactDOM } from "react";

 //example :3

 //Functional component with Properties

const FunctionalComponent = ({name}) =>{

     var myStyle={

         color: 'green'

     };

     var heading={

         color:'grey',

         textDecorationLine: 'underline'

     }

     return <div><h1 style={heading}>Functional component with Properties </h1>

      <h2 style={myStyle} > welcome :{name}</h2> </div> ;

 }

 export default FunctionalComponent;

app.js

How to pass the value for props in app.js file

import FunctionalComponent from './Components/FcuntionalComponent';

    <FunctionalComponent name='Geetha' />

Another approach with props

const FunctionalComponent = (props) =>{

     var myStyle={

         color: 'green'

     };

     var heading={

         color:'grey',

         textDecorationLine: 'underline'

     }

     return <div><h1 style={heading}>Functional component with Properties </h1>

      <h2 style={myStyle} > welcome :{props.name}</h2> </div> ;

 }

# **React State**

The state is an updatable structure that is used to contain data or information about the component. The state in a component can change over time. The change in state over time can happen as a response to user action or system event. A component with the state is known as stateful components. It is the heart of the react component which determines the behavior of the component and how it will render. They are also responsible for making a component dynamic and interactive.

A state must be kept as simple as possible. It can be set by using the **setState()** method and calling setState() method triggers UI updates. A state represents the component's local state or information. It can only be accessed or modified inside the component or by the component directly. To set an initial state before any interaction occurs, we need to use the **getInitialState()** method.

**For example**, if we have five components that need data or information from the state, then we need to create one container component that will keep the state for all of them.

## Handling state

Now we all know that we cannot avoid dealing with state variables in a React project. Handling state was only doable in a class component until recently, but from React 16.8, React Hook [useState](https://reactjs.org/docs/hooks-state.html) was introduced to allow developers to write stateful functional components. Here we are going to make a simple counter that starts from 0, and one click on button will increment the count by 1.

### Handling state in functional components

//Example :4 Handling a state with functional component

 const FunctionalComponent =() => {

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

    // var count=0;

     return (

         <div>

             <p> Count:{count}</p>

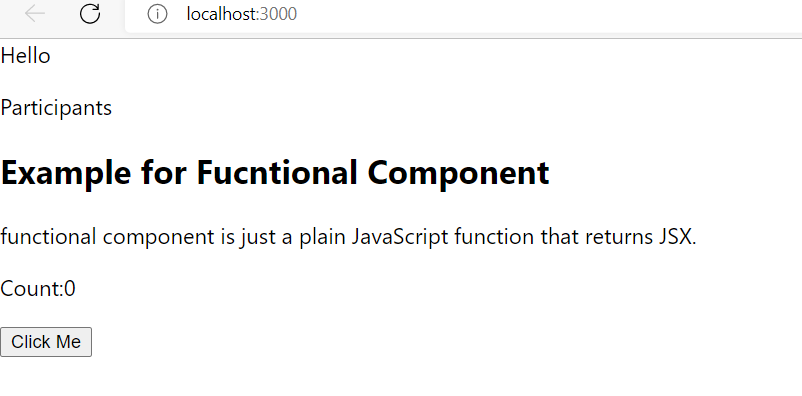
           {/\* //  <button onClick={count+1}> Click Me</button> \*/}

             <button onClick={ () => setCount(count+1)}> Click Me</button>

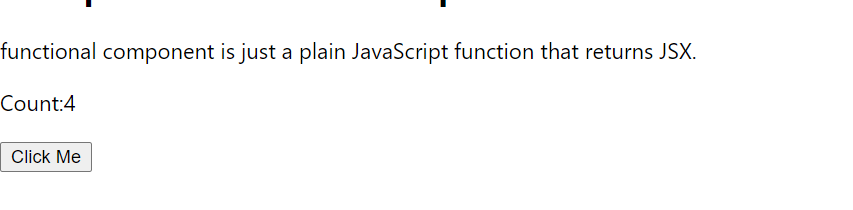
         </div>

     );

 };



In Every click count value gets incremented by1



Example for Class Component

omponents are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML via a render() function.

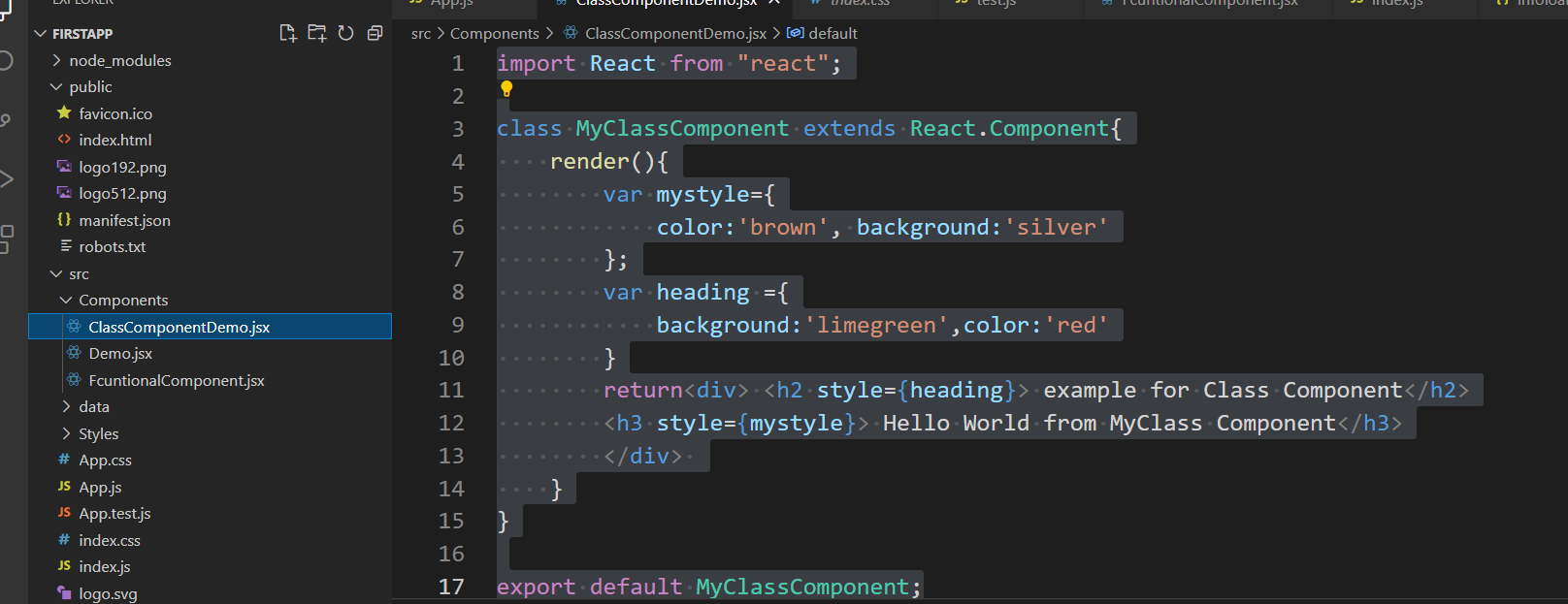
Create a Class Component

When creating a React component, the component's name must start with an upper case letter.

The component has to include the extends React.Component statement, this statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.

The component also requires a render() method, this method returns HTML.

Create a newFile with the name of ClassComponentDemo.jsx



Example without using destructuring

import React from "react";

class MyClassComponent extends React.Component{

    render(){

        var mystyle={

            color:'brown', background:'silver'

        };

        var heading ={

            background:'limegreen',color:'red'

        }

        return<div> <h2 style={heading}> example for Class Component</h2>

        <h3 style={mystyle}> Hello World from MyClass Component</h3>

        </div>

    }

}

export default MyClassComponent;

app.js

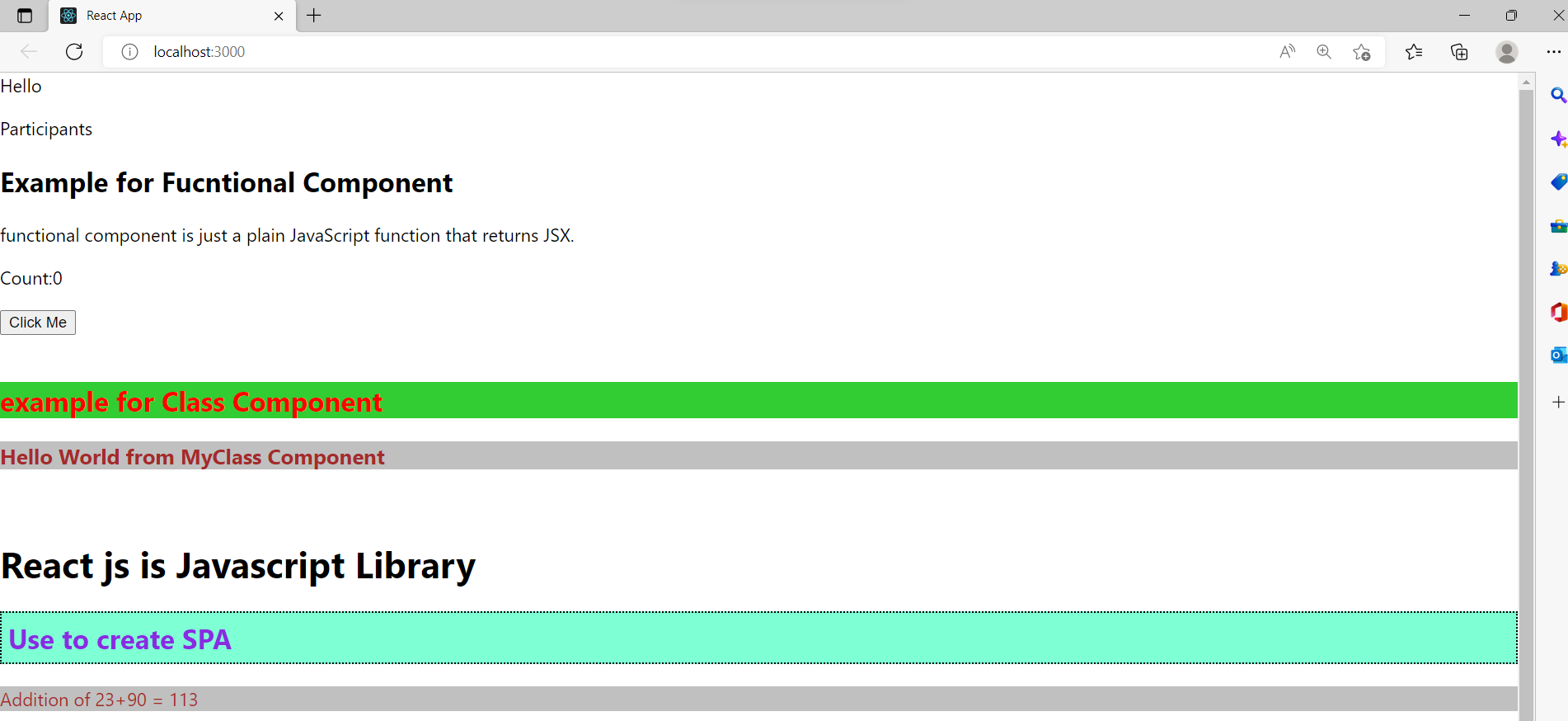
import MyClassComponent from './Components/ClassComponentDemo';

within the return method

 <br/>

      <MyClassComponent />

    <br/>



Approach :2

import React,{Component} from "react";

class MyClassComponent extends Component{

    render(){

        var mystyle={

            color:'brown', background:'silver'

        };

        var heading ={

            background:'limegreen',color:'red'

        }

        return<div> <h2 style={heading}> example for Class Component</h2>

        <h3 style={mystyle}> Hello World from MyClass Component</h3>

        </div>

    }

}

export default MyClassComponent;

class component with props

ClasscompoenentDemo.jsx

import React,{Component} from "react";

class MyClassComponent extends Component{

    render(){

        var mystyle={

            color:'brown', background:'silver'

        };

        var heading ={

            background:'limegreen',color:'red'

        }

        const  {name}=this.props;

        return<div> <h2 style={heading}> example for Class Component</h2>

        <h3 style={mystyle}> Hello World from MyClass Component</h3>

        <p> Welcome {name}!</p>

        </div>

    }

}

export default MyClassComponent;

app.js

to pass the value to the props

<br/>

      <MyClassComponent name='Peter' />

    <br/>



Class Component with state

ClassComponentDemo.js

import React from "react";

class MyClassComponent extends React.Component{

    constructor(props)

    {

        super(props);

        this.state ={

            count:0

        }

    }

    render(){

       return(

           <div>

               <h2> Name :{this.props.name}</h2>

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

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

           </div>

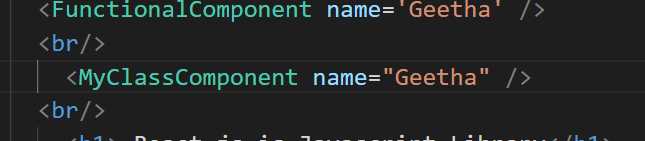
       )

    }

}

export default MyClassComponent;

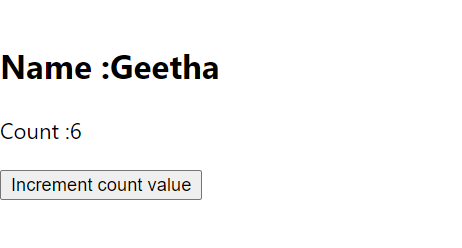
app.js



 <br/>

      <MyClassComponent name="Geetha" />

    <br/>



Another Example for Class components with Properties

import React from "react";

class StudentDetails extends React.Component

{

    constructor()

    {

        super();

            this.state={

                data:[

                    {name:'Divyasree'},

                    {name:'Poonam Samal'},

                    {name:'Neelima'}

                ]

            }

    }

    render()

    {

        return(

            <div>

                <StudentName />

                <ul>

                    {this.state.data.map((item) => <List data = {item} />)}

                </ul>

            </div>

        )

    }

}

class StudentName extends React.Component

{

    render()

    {

        return(

            <div> <h1>Student Details</h1> </div>);

    }

}

class List extends React.Component

{

    render() {

        return (

             <ul>

                 <li> {this.props.data.name}</li>

             </ul>

        );

    }

}

export default StudentDetails;

App.js

  return (<div>

      <StudentDetails />

Another example toggle

import { toHaveDisplayValue } from "@testing-library/jest-dom/dist/matchers";

import React,{Component} from "react";

class ToggleDemo extends React.Component

{

    constructor()

    {

        super();

        this.state={displayDetails:false};

        console.log( 'Component this',this);

        this.toggleDisplayDetails= this.toggleDisplayDetails.bind(this);

    }

     toggleDisplayDetails(){

         this.setState({displayDetails:!this.state.displayDetails});

     }

        render()

        {

            return (

                <div>

                    <h1>Welcome to React!!</h1>

                    {

                        this.state.displayDetails ? (

                            <div>

                                <p><h4>React is one of the best front end UI.

                                The bind() method creates a new function that, when called, has its this keyword set to the provided value, with a given sequence of arguments preceding any provided when the new function is called.</h4></p>

                                <button onClick={this.toggleDisplayDetails}> Show Less </button>

                          </div>

                            ) : (

                                <div>

                                    <button onClick={this.toggleDisplayDetails}> Read More </button>

                                </div>

                            )

                    }

               </div>

            )

        }

    }

    export default ToggleDemo;