**Components in React JS**

* What are Components in React JS

Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML.

* Types of Components

Components come in two types, Class components and Function components.

* Functional component

A Function component also returns HTML, and behaves much the same way as a Class component, but Function components can be written using much less code, are easier to understand.

* Class Component

A class component must 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.

* Functional component vs Class Component

|  |  |
| --- | --- |
| Functional Components | Class Components |
| A functional component is just a plain JavaScript pure function that accepts props as an argument and returns a React element(JSX). | A class component requires you to extend from React. Component and create a render function which returns a React element. |
| There is no render method used in functional components. | It must have the render() method returning JSX (which is syntactically similar to HTML) |
| Functional component run from top to bottom and once the function is returned it cant be kept alive. | Class component is instantiated and different life cycle method is kept alive and being run and invoked depending on phase of class component. |
| Also known as Stateless components as they simply accept data and display them in some form, that they are mainly responsible for rendering UI. | Also known as Stateful components because they implement logic and state. |
| React lifecycle methods (for example, componentDidMount) cannot be used in functional components. | React lifecycle methods can be used inside class components (for example, componentDidMount). |
| Hooks can be easily used in functional components to make them Stateful.  example: const [name,SetName]= React.useState(‘ ‘) | It requires different syntax inside a class component to implement hooks.  example: constructor(props) {     super(props);     this.state = {name: ‘ ‘}  } |
| Constructors are not used. | Constructor are used as it needs to store state. |

* Creating Class Components

**Example:** Program to demonstrate the creation of class-based components. Create a React app and edit the App.js as:

**Filename:** App.js:

import React from "react";

class App extends React.Component {

render() {

return <h1>My First Class Component</h1>;

}

}

export default App;

**Output:**

My First Class Component

* Creating Functional Components

**Example:** Program to demonstrate the creation of functional-based components. Create a React app and edit the App.js as:

**Filename:** App.js:

import React from 'react';

import ReactDOM from 'react-dom';

const App=()=>

{

return <h1>My First Functional Component </h1>;

}

export default App;

**Output:**

My First Functional Component

* Converting Functional Components to Class Components

If we want to convert a function component to a class component then we need to make the following major changes.

* Change the function to a class
* Add the render method
* Convert all function to a method
* Add Constructor
* Replace hooks with lifecycle methods

**Creating React Application:**

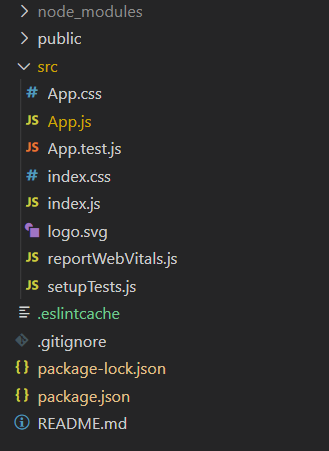
* **Step 1:**Create a React application using the following command:

npx create-react-app foldername

* **Step 2:**After creating your project folder i.e. foldername, move to it using the following command:

cd foldername

**Project Structure:** It will look like the following.



**Example: Functional Component**

Write down the following code in the **App.js** file. Here, App is our default component where we have written our code.

**App.js**

|  |
| --- |
| import React, { useState } from 'react';  function App() {    const [counter, setCounter] = useState(0);    const myFunction = (e) => {      alert("The value of counter is " + counter)      setCounter(counter + 1);    }    return (      <div>  <p>Hello Functional Component</p>        <button onClick={myFunction}>Click me!</button>      </div>    );  }  export default App; |

**Step to Run Application:**Run the application using the following command from the root directory of the project:

npm start

**Output:**Now open your browser and go to***http://localhost:3000/***, you will see the following output:

Hello Functional Component

**Click me!**

**Example: Class Component**

Using the above steps to convert a function component to a class component, we have written the following code in the **App.js** file.

**App.js**

|  |
| --- |
| import React, { useState } from 'react';  class App extends React.Component {    constructor(props) {      super(props)      this.state = {        counter: 0      }      this.myFunction = this.myFunction.bind(this);    }    myFunction(e) {      alert("The value of counter is " + this.state.counter)      this.setState({ counter: this.state.counter + 1 })    }    render() {      return (        <div >          <p>Hello Class Component</p>          <button onClick={this.myFunction}>           Click me!          </button>        </div>      );    }  }  export default App; |

**Step to Run Application:**Run the application using the following command from the root directory of the project:

npm start

**Output:**Now open your browser and go to***http://localhost:3000/***, you will see the following output:

Hello Class Component

**Click me!**