In this article, we will learn about the differences between functional and class components in React with the help of an example. We will create a counter and implement it using both class and functional components to understand the differences practically.

**Functional Components**

Functional components are some of the more common components that will come across while working in React. These are simply JavaScript functions. We can create a functional component to React by writing a JavaScript function.

**Syntax:**

const Car=()=> {  
 return <h2>Hi, I am also a Car!</h2>;  
}

**Counter using Functional Components**

**Example:**

Javascript

import React, { useState } from "react";

const FunctionalComponent = () => {

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

const increase = () => {

setCount(count + 1);

}

return (

<div style={{ margin: '50px' }}>

<h1>Welcome to Geeks for Geeks </h1>

<h3>Counter App using Functional Component : </h3>

<h2>{count}</h2>

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

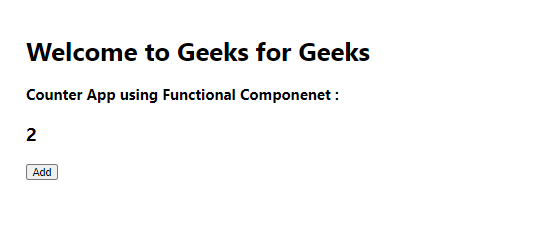
</div>

)

}

export default FunctionalComponent;

**Output:**

**Class Component**

This is the bread and butter of most modern web apps built in ReactJS. These components are simple classes (made up of multiple functions that add functionality to the application).

**Syntax:**

class Car extends React.Component {  
 render() {  
 return <h2>Hi, I am a Car!</h2>;  
 }  
}

**Counter using Class Components**

**Example:**

Javascript

import React, { Component } from "react";

class ClassComponent extends React.Component {

constructor() {

super();

this.state = {

count: 0

};

this.increase = this.increase.bind(this);

}

increase() {

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

}

render() {

return (

<div style={{ margin: '50px' }}>

<h1>Welcome to Geeks for Geeks </h1>

<h3>Counter App using Class Component : </h3>

<h2> {this.state.count}</h2>

<button onClick={this.increase}> Add</button>

</div>

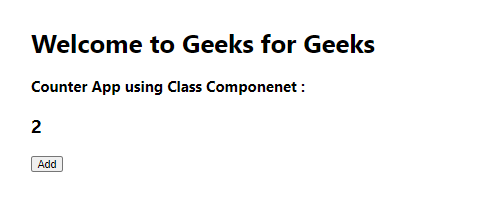
)

}

}

export default ClassComponent;

**Output:**



In the above example, for functional components, we use hooks (useState) to manage the state. If you write a function component and realize you need to add some state to it, previously you had to convert it to a class component. Now you can use a Hook inside the existing function component to manage the state and no need to convert it into the Class component. Hooks are a new addition to React 16.8. They let you use state and other React features without writing a class. Instead of Classes, one can use Hooks in the Functional component as this is a much easier way of managing the state. Hooks can only be used in functional components, not in-class components.

**Functional Components vs Class Components:**

|  |  |
| --- | --- |
| [**Functional Components**](https://www.geeksforgeeks.org/reactjs-functional-components/) | [**Class Components**](https://www.geeksforgeeks.org/reactjs-class-based-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 that 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 components run from top to bottom and once the function is returned it can’t be kept alive. | The class component is instantiated and different life cycle method is kept alive and is run and invoked depending on the phase of the class component. |
| Also known as Stateless components as they simply accept data and display them in some form, 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 is used as it needs to store state. |

Last Updated : 02 Aug, 2023

113

Similar Reads

[[](https://www.geeksforgeeks.org/differences-between-functional-and-non-functional-testing/)](https://www.geeksforgeeks.org/differences-between-functional-and-non-functional-testing/)

[Differences betw](https://www.geeksforgeeks.org/differences-between-functional-and-non-functional-testing/)