**Question 1: What are components in React? Explain the difference between functional components and class components.**

**What Are Components in React?**

In React, components are the building blocks of the user interface. They allow developers to split the UI into reusable, independent pieces. Each component is like a JavaScript function that returns JSX to describe what should appear on the screen.

There are two main types of components in React:

1. Functional Components
2. Class Components
3. **Functional Components (Modern Approach):**

Functional components are **JavaScript functions** that return JSX. They are simpler, easier to read, and the preferred way to create components in modern React.

**Example of a Functional Component:**

import React from "react";

const Greeting = () => {

return <h1>Hello, welcome to React!</h1>;

};

export default Greeting;

**2. Class Components (Older Approach)**

Class components are ES6 classes that extend React.Component and must include a render() method to return JSX.

**Example of a Class Component:**

import React, { Component } from "react";

class Greeting extends Component {

render() {

return <h1>Hello, welcome to React!</h1>;

}

}

export default Greeting**;**

**Question 2: How do you pass data to a component using props?**

**How to Pass Data to a Component Using Props in React?**

In React, props (short for "properties") allow us to pass data from a parent component to a child component. Props make components reusable and dynamic by allowing them to receive different values.

**1. Passing Props from Parent to Child Component**

To pass data, we add attributes to the child component in the parent, just like HTML attributes.

**Example: Parent Component Passing Props**

import React from "react";

import Greeting from "./Greeting";

const App = () => {

return <Greeting name="Alice" age={25} />;

};

export default App;

**2.Receiving and Using Props in the Child Component**

The child component receives props as an argument.

**Example: Child Component Receiving Props**

import React from "react";

const Greeting = (props) => {

return <h1>Hello, {props.name}! You are {props.age} years old.</h1>;

};

export default Greeting;

**3. Using Object Destructuring for Cleaner Code**

Instead of using props.name, we can **destructure** props.

const Greeting = ({ name, age }) => {

return <h1>Hello, {name}! You are {age} years old.</h1>;

};

**4. Passing Functions as Props (Prop Drilling Example)**

Props can also pass functions to child components.

**Example: Passing a Function as a Prop**

const App = () => {

const handleClick = () => {

alert ("Button Clicked!");

};

return <Button handleClick={handleClick} />;

};

**Child Component Using Function Prop**

const Button = ({handleClick}) => {

return <button onClick={handleClick}>Click Me</button>;

};

**Question 3: What is the role of render() in class components?**

Role of render() in Class Components in React

In React class components, the render() method is required and is responsible for returning JSX (UI elements) that should be displayed on the screen.

1. **Basic Usage of render()**

Every class component **must** include a render() method, which:  
✔ Returns **JSX** that represents the component’s UI.  
✔ Runs **whenever the component re-renders** due to state or props changes.

**Example of render() in a Class Component:**

import React, { Component } from "react";

class Greeting extends Component {

render() {return <h1>Hello, {this.props.name}!</h1>}

}

export default Greeting;

2. **render() Works with State & Props**

The render() method automatically updates the UI when **state** or **props** change.

**Example with State:**

import React, { Component } from "react";

class Counter extends Component {

constructor() {

super();

this.state = { count: 0 };

}

increment = () => {

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

};

render() {

return (

<div>

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

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

</div>

);

}

}

export default Counter;

**3.When is render() Called?**

The render() method runs:  
✔ **When the component mounts** (first time it appears).  
✔ **When props change** (parent component sends new data).  
✔ **When state updates** (this.setState() is called).

**4️. Limitations of render()**

🚫 **Should not modify state directly** – It should only return JSX.  
🚫 **Should not contain side effects** (e.g., API calls, timers) – Use componentDidMount() instead.