### ****React Components****

React components are **reusable building blocks** of a user interface. Each component defines a part of the UI and can be composed together to build complex applications.

* They accept **props** (inputs) and return **JSX** (UI structure).
* Components can be **nested**, **reused**, and **managed independently**.
* They help maintain a **modular architecture** in React apps.

### 🔍 ****Difference Between Components and JavaScript Functions****

| **Aspect** | **React Component** | **JavaScript Function** |
| --- | --- | --- |
| Purpose | Defines UI and behavior in React | Performs logic or computation |
| Return Value | Returns JSX (React elements) | Returns any JavaScript value |
| Lifecycle | Has lifecycle methods (class) or hooks | No lifecycle awareness |
| React Integration | Used with <ComponentName /> syntax | Called like functionName() |
| State Management | Can manage state (via hooks or class) | No built-in state management |

### ****Types of React Components****

React offers two primary types:

1. **Class Components**
2. **Function Components**

Other patterns include:

* Higher-Order Components (HOCs)
* Render Props
* Custom Hooks (used within function components)

### 🏛️ ****Class Compnent****

A class component is an ES6 class that extends React.Component. It can manage state and lifecycle methods.

class Welcome extends React.Component {

render() {

return <h1>Hello, {this.props.name}</h1>;

}

}

* Uses this.state for internal data
* Lifecycle methods like componentDidMount, componentDidUpdate
* Requires render() method to return JSX

### 🧠 ****Function Component****

A function component is a plain JavaScript function that returns JSX. With hooks, it can manage state and side effects.

function Welcome(props) {

return <h1>Hello, {props.name}</h1>;

}

Or with hooks:

function Counter() {

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

return <button onClick={() => setCount(count + 1)}>Count: {count}</button>;

}

* Simpler syntax
* Preferred in modern React
* Uses hooks like useState, useEffect

### 🏗️ ****Component Constructor****

The constructor is a special method in class components used to initialize state and bind methods.

class Car extends React.Component {

constructor(props) {

super(props);

this.state = { color: 'red' };

}

}

* Called before the component mounts
* Must call super(props) to access this.props
* Used for initial setup

### 🖼️ ****render() Function****

The render() method is **mandatory** in class components. It returns the JSX that defines the UI.

class Car extends React.Component {

render() {

return <h2>I am a {this.state.color} car!</h2>;

}

}

* Called whenever state or props change
* Should return a single parent JSX element
* React uses it to update the virtual DOM