### **React Components**

**React components** are the basic building blocks of a React app. They’re like small, reusable pieces of UI. Think of them like **LEGO blocks** — you create small parts and combine them to build full pages.

Each component can have its own **logic**, **design**, and even **data**.

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

While both components and JavaScript functions can take input and return output, here’s how they differ:

| **Aspect** | **JS Function** | **React Component** |
| --- | --- | --- |
| Purpose | Executes logic | Renders UI |
| Return | Any value | JSX (UI) |
| React-specific features | None | Can use state, lifecycle, etc. |

So, components are like **upgraded functions** built specifically to handle UI in React.

### **Types of Components**

There are mainly two types:

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

### **Class Component**

Class components are the older way of writing components. They are ES6 classes that extend React.Component. They have more features like **state** and **lifecycle methods** built-in.

Example:

class Welcome extends React.Component {

render() {

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

}

}

### **Function Component**

Function components are simpler and preferred now. Earlier, they couldn't use state or lifecycle, but with **Hooks** (like useState), they can.

Example:

function Welcome(props) {

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

}

Or using **arrow functions**:

const Welcome = (props) => <h1>Hello, {props.name}</h1>;

### **Component Constructor**

In class components, the constructor is used to initialize **state** and bind methods. It’s the first method that runs when a component is created.

Example:

class Welcome extends React.Component {

constructor(props) {

super(props);