Definations:

**React Components Explained**

A **React component** is an independent, reusable piece of UI, written in JavaScript (or TypeScript), that controls a portion of the user interface and implements its own logic. Components make it easy to break a complex UI into smaller, manageable parts.

**Differences Between Components and JavaScript Functions**

While React function components are technically JavaScript functions, not all JavaScript functions are React components. Here are the key differences:

| **Aspect** | **React Component** | **JavaScript Function** |
| --- | --- | --- |
| Purpose | Describes UI and returns JSX | Executes any logic, returns any value |
| Naming | Starts with uppercase (e.g., MyComp) | Any naming convention |
| Lifecycle methods | (Class components) have lifecycle | No lifecycle methods |
| JSX Usage | Returns JSX/React elements | Returns any value/data |
| Used by React | Interpreted by React’s rendering | Not recognized by React as components |

**Types of Components in React**

1. **Function Components**: The most common and recommended type, declared as functions and return JSX.
2. **Class Components**: Use JavaScript classes, have lifecycle methods, and can use state through this.state and this.setState().
3. **Pure Components**: Special class components that implement shallow comparison in their shouldComponentUpdate method.

**Class Component Explained**

A **class component** is a React component defined as a JavaScript class extending React.Component. It supports state, lifecycle methods (such as componentDidMount), and must include a render() method that returns JSX. Example:

jsx

**import** React, { Component } **from** 'react';

**class** Greeting **extends** Component {

constructor(props) {

**super**(props); *// see next section for explanation*

**this**.state = { name: 'World' };

}

render() {

**return** <h1>Hello, {**this**.state.name}!</h1>;

}

}

**Function Component Explained**

A **function component** is a plain JavaScript function that returns JSX, does not have lifecycle methods (until hooks were introduced), and can use state and side effects via React Hooks (useState, useEffect).

Example:

jsx

**function** Greeting(props) {

**const** [name, setName] = React.useState('World');

**return** <h1>Hello, {name}!</h1>;

}

**Component Constructor**

The **constructor** is a special method in class components, used to:

* Initialize state (with this.state = ...)
* Bind event handlers
* Receive props

It’s the first method called when an instance of a class is created.

jsx

constructor(props) {

**super**(props); *// Sets up this.props*

**this**.state = { count: 0 };

}

**render() Function**

The render() method is required in a React class component. It returns the JSX (React elements) to be displayed on the UI. It is pure—which means it does not modify component state or interact with the browser directly.

jsx

render() {

**return** <div>Hello</div>;

}

* **In summary:** components are the core building blocks of React applications. They can be defined as either functions or classes, with function components being the modern standard. Class components feature constructors and a mandatory render() method, while function components use hooks for state and effects.