# **OBJECTIVES**

# **Explanation of React Components**

In React, a **component** is a reusable, independent block of code that defines a part of a user interface (UI). Components help in breaking down complex UIs into smaller, manageable pieces, making development more efficient.

### **Key Characteristics of React Components:**

* **Reusable:** Can be used multiple times across an application.
* **Independent:** Each component manages its own state and logic.
* **Composable:** Components can be nested inside other components.
* **Isolated:** Changes in one component do not directly affect others.

# **Differences Between Components and JavaScript Functions**

| **Feature** | **React Components** | **JavaScript Functions** |
| --- | --- | --- |
| **Purpose** | Render UI elements | Perform computations or logic |
| **Return Value** | Returns JSX (HTML-like syntax) | Returns any data type |
| **State Management** | Can maintain internal state (if needed) | Stateless (unless using closures) |
| **Lifecycle** | Has lifecycle methods (in class components) | No lifecycle methods |
| **Usage** | Used in React applications | Used in any JavaScript program |
| **Example** | function Greeting(){return <h1>Hello,World!</h1>;} | function add(a, b){return a + b;} |

# **3. Types of Components in React**

React supports two main types of components:

**Class Components**

* Defined using ES6 classes.
* Have lifecycle methods (componentDidMount, componentDidUpdate ).
* Can hold state (this.state).

**Functional Components**

* Defined as JavaScript functions.
* Initially stateless, but with **Hooks** (useState , useEffect), they can manage state.
* Simpler and more modern (recommended for new projects).

# **4. Explanation of Class Components**

Class components are traditional React components that use **ES6 classes** and extend React.Component

### **Features of Class Components:**

* Use this.state to manage internal state.
* Have lifecycle methods (componentDidMount , render, etc.).
* Require a render() method to return JSX.

**Example:**

jsx

import React from 'react';

class Counter extends React.Component {

constructor(props) {

super(props);

this.state = { count: 0 };

}

render() {

return <div>Count: {this.state.count}</div>;

}

}

# **5. Explanation of Function Components**

Functional components are simpler and use **JavaScript functions** instead of classes. With React Hooks, they can now manage state and side effects.

### **Features of Function Components:**

* Lightweight and easier to read.
* Use useState for state management.
* Use useEffect for side effects (like data fetching).

**Example:**

jsx

import { useState } from 'react';

function Counter() {

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

return <div>Count: {count}</div>;

}

# **6. Definition of Component Constructor**

In **class components**, the constructor is a special method used for:

* Initializing **local state** (this.state).
* Binding **event handlers** (this.handleClick = this.handleClick.bind(this)).

### **Key Points:**

* Must call super(props) before using this.
* Used only in class components (not needed in functional components).

**Example:**

jsx

class User extends React.Component {

constructor(props) {

super(props);

this.state = { name: "Gopika" };

this.handleClick = this.handleClick.bind(this);

}

render() { */\* ... \*/* }

}

# **7. Definition of render() Function**

The **render()**  method is a **required** function in **class components** that determines what gets displayed on the screen.

### **Key Features of render()** **:**

* Must return **JSX** (React elements).
* Should be **pure** (no side effects like API calls).
* Called automatically when state or props change.

**Example:**

jsx

class App extends React.Component {

render() {

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

}

}

* Functional components **do not need** a **render()** method—they directly return JSX.