**Hands-on: 3. ReactJS-HOL**

**Introduction**

React is a powerful JavaScript library used to build modern, interactive user interfaces. The core concept in React is the component, which lets developers break down complex UIs into smaller, reusable parts. Understanding components and their structure is essential for building clean and maintainable React applications.

1. **Explain React Components**

A React component is a self-contained module that represents a part of the user interface (UI). It receives input (called props) and returns JSX (JavaScript XML) to define what appears on the screen.

* **Example**: A button, a form, or a header can each be a separate component.
* Components allow for code reuse, better organization, and easier maintenance.

1. **Identify the Differences Between Components and JavaScript Functions**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **React Component** | **JavaScript Function** |
| **Purpose** | Used to render UI elements in a React app | Used for logical operations or calculations |
| **Return** | Returns JSX (UI structure) | Returns values (strings, numbers, etc.) |
| **Integration** | Works within React's UI lifecycle | Independent of any UI framework |
| **State Handling** | Can use state and hooks | No state or React-specific features |
| **UI Binding** | Designed to output HTML via JSX | Does not produce UI |

1. **Identify the Types of Components**

There are two main types of React components:

1. Class Components – Use ES6 classes and can manage their own state and lifecycle methods.
2. Function Components – Use plain JavaScript functions and can use hooks to manage state and lifecycle.

* Function components are now preferred in modern React due to their simplicity and hook support.

**4. Explain Class Component**

A Class Component is a component defined using an ES6 class that extends React.Component.

* It must include a render() method, which returns the JSX to display.
* **Example:**

class Greeting extends React.Component {

render() {

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

}

}

* Can use state, props, and lifecycle methods like componentDidMount().

1. **Explain Function Component**

A Function Component is a simple JavaScript function that returns JSX. It can now use hooks to manage state and side effects.

* **Example:**

function Greeting(props) {

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

}

**With hooks:**

import { useState } from 'react';

function Counter() {

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

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

}

* Function components are cleaner, shorter, and now just as powerful as class components.

1. **Define Component Constructor**

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

* **Syntax:**

constructor(props) {

super(props);

this.state = { count: 0 };

}

* It’s called once when the component is created.
* super(props) is required to access this.props in the constructor.

1. **Define render() Function**

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

* **Example:**

render() {

return <div>Hello World</div>;

}

* React automatically calls render() when the component needs to be displayed or re-rendered after updates.

**Conclusion**

React components, whether class-based or function-based, are essential for building modern web interfaces. Understanding their structure, differences, and core methods like constructor and render() helps developers write clean, modular, and maintainable code. As React continues to evolve, function components with hooks are becoming the new standard, offering both simplicity and power in UI development.