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

**React Components: Structure and Functionality**

React components are fundamental building blocks for constructing modern, interactive user interfaces. Understanding how they work, their types, and their differences from regular JavaScript functions is essential for developing scalable React applications.

**What are React Components?**

A React component is a reusable, self-contained block of code that represents a part of the user interface. Components accept input, called props, and return React elements that describe what should appear on the screen. They help in breaking down complex UIs into smaller, manageable pieces.

**Components vs. JavaScript Functions: Key Differences**

| Aspect | React Component | JavaScript Function |
| --- | --- | --- |
| Purpose | Builds UI elements in React | Performs logic or calculations |
| Return Value | Returns JSX (React elements) | Returns any value (number, string, object, etc.) |
| Reusability | Can be reused in React UI | Reused for logic, not UI |
| State Handling | Can manage state (especially in class/function with hooks) | Does not manage UI state |
| React Features | Uses props, state, lifecycle (render, useEffect, etc.) | Purely logic-based without React context |

**Types of React Components**

React has two main types of components:

* **Class Components:** These are ES6 classes that extend React.Component.
* **Function Components:** These are functions that return JSX and can use hooks.

**Class Components Explained**

A Class Component is a JavaScript class that extends React.Component and must define a render() method to return JSX.

**Example:**

JavaScript

class Welcome extends React.Component {

render() {

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

}

}

Class components have lifecycle methods like componentDidMount() , can manage state using

this.state , and are more verbose than function components.

**Function Components Explained**

A Function Component is a simpler way to write components using plain JavaScript functions.

**Example:**

JavaScript

function Welcome(props) {

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

}

**With Hooks:**

JavaScript

import { useState } from 'react';

function Counter() {

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

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

}

Function components have less boilerplate , support hooks like useState and useEffect , and are preferred for modern React development.

**Component Constructor Definition**

The constructor() method is a special function used inside class components to initialize state and bind methods.

**Syntax:**

JavaScript

constructor(props) {

super(props);

this.state = { count: 0 };

}

Its purpose is to set up initial state , bind class methods to this , and access props before render() runs.

**Function Definition**

The render() function is required in all class components. It returns the JSX that defines the UI for that component.

**Example:**

JavaScript

render() {

return <div>Hello, World!</div>;

}

Its role is to describe what to display and it is called automatically by React when state or props change.

In conclusion, React components, whether class-based or functional, are at the heart of any React application. Understanding their differences, structure, and key methods like

constructor() and render() helps developers build scalable and efficient UIs. With the rise of hooks, functional components have become the standard, offering simplicity without sacrificing power. Mastery of components is the first step to becoming proficient in React development.