**Hands-on: 2. ReactJS-HOL**

**Introduction**

In React, components are the foundational building blocks used to create dynamic and reusable user interfaces. Understanding how components work, their types, and how they differ from regular JavaScript functions is essential for developing scalable React applications. This guide covers the nature of React components, their types, and key lifecycle features such as constructors and the render() method.

1. **Explain 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.

* Components help in breaking down complex UIs into smaller, manageable pieces.

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

|  |  |  |
| --- | --- | --- |
| **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 |

1. **Identify the Types of Components**

React has two main types of components:

1. Class Components – ES6 classes that extend React.Component
2. Function Components – Functions that return JSX (can use hooks)
3. **Explain Class Component**

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

* **Example:**

class Welcome extends React.Component {

render() {

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

}

}

* **Features:**
* Has lifecycle methods like componentDidMount()
* Can manage state using this.state
* More verbose than function components

1. **Explain Function Component**

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

* **Example:**

function Welcome(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>;

}

* **Features:**
* Less boilerplate
* Supports hooks like useState, useEffect
* Preferred for modern React development

1. **Define Component Constructor**

The constructor() method is a special function used inside class components. It initializes state and binds methods.

* **Syntax:**

constructor(props) {

super(props);

this.state = { count: 0 };

}

* **Purpose:**
* Set up initial state
* Bind class methods to this
* Access props before render() runs

1. **render() Function**

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

* **Example:**

render() {

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

}

* **Role:**
* Describes what to display
* Called automatically by React when state or props change

**Conclusion**

React components—both class-based and functional—are the heart of any React application. Understanding their differences, structure, and key methods like constructor and render() helps developers build scalable, 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.