**Hands-on: 8. ReactJS-HOL**

**React State: Managing Dynamic Data in Components**

React State is a built-in JavaScript object used to store dynamic data within a component. It dictates how that component behaves and what it renders. Unlike props, which are passed from a parent to a child, state is managed internally by the component itself and can be modified over time, typically in response to user actions or other events.

**Key Features of State:**

* **Local to the component:** State is not directly accessible outside the component unless it is "lifted up".
* **Mutable:** State can be updated using specific methods like setState() in class components or useState() in functional components.
* **Triggers re-render:** Whenever the state changes, the component automatically re-renders to reflect the new state.
* **Used for dynamic behavior:** State is employed for managing dynamic aspects such as user input, toggles, counters, and more.

**Class Component Example:**

JavaScript

class Counter extends React.Component {

constructor(props) {

super(props);

this.state = { count: 0 }; // Initialize state

}

increment = () => {

this.setState({ count: this.state.count + 1 }); // Update state

};

render() {

return (

<div>

<p>Count: {this.state.count}</p>

<button onClick={this.increment}>Add</button>

</div>

);

}

}

**Function Component Example with Hooks:**

JavaScript

import { useState } from 'react';

function Counter() {

const [count, setCount] = useState(0); // 'count' is the state variable

return (

<div>

<p>Count: {count}</p>

<button onClick={() => setCount(count + 1)}>Add</button>

</div>

);

}

**When to Use State:**

* To manage user input values in forms.
* To toggle elements, such as showing or hiding a password.
* To track counts, status, modes, or selections.
* To dynamically update the UI based on user interaction or API responses.

In conclusion, React state allows components to be interactive and dynamic. While props are used to configure a component, state controls its behavior over time. Effectively managing state, whether with class-based components or modern functional components using hooks, is crucial for building responsive and engaging applications in React.