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

1. **What is React State?**

React State is a built-in JavaScript object used to store dynamic data within a component. It determines how that component behaves and what it renders.

Unlike props, which are passed from parent to child, state is managed within the component itself and can be changed over time, usually in response to user actions or other events.

* **Key Features of State:**
* Local to the component – not accessible outside unless lifted up
* Mutable – can be updated using special methods (e.g., setState() or useState())
* Triggers re-render – when state changes, the component re-renders with the new state
* Used for dynamic behavior – such as user input, toggles, counters, etc.
* **Class Component Example:**

class Counter extends React.Component {

constructor(props) {

super(props);

this.state = { count: 0 };

}

increment = () => {

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

};

render() {

return (

<div>

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

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

</div>

);

}

}

* **Function Component Example with Hooks:**

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 (e.g., show/hide password)
* To track counts, status, modes, selections, etc.
* To update the UI dynamically based on user interaction or API responses

**Conclusion**

React state allows components to be interactive and dynamic. While props configure a component, state controls its behavior over time. Whether using class-based components or modern functional components with hooks, managing state effectively is key to building responsive and engaging applications in React.