**Hands-on: 4. ReactJS-HOL**

**Introduction**

In React, components go through a lifecycle from the moment they are created to when they are removed from the UI. React provides lifecycle methods (also called hooks) that allow developers to execute code at specific stages of this lifecycle. Understanding these methods helps in managing data fetching, DOM updates, cleanup, and performance optimization.

1. **Explain the Need and Benefits of Component Lifecycle**

* **Need:**

React components need a way to handle tasks at different stages—like when they’re mounted, updated, or unmounted. Lifecycle methods offer this control.

* **Benefits:**
* Control over behavior at different phases (mount, update, unmount)
* Efficient resource management (e.g., start/stop timers)
* Data fetching when a component loads
* Conditional rendering during updates
* Cleanup of memory, listeners, or intervals on component removal

1. **Identify Various Lifecycle Hook Methods**

React (Class Component) lifecycle is divided into **3 main phases**, each with specific methods:

* **Mounting (Component is being created and inserted into DOM):**
* constructor()
* static getDerivedStateFromProps()
* render()
* componentDidMount()
* **Updating (Component is re-rendered due to changes in props or state):**
* static getDerivedStateFromProps()
* shouldComponentUpdate()
* render()
* getSnapshotBeforeUpdate()
* componentDidUpdate()
* **Unmounting (Component is removed from DOM):**
* componentWillUnmount()
* **Error Handling (if component throws an error):**
* componentDidCatch()
* static getDerivedStateFromError()

**Note**: In Function Components, similar behavior is achieved using React Hooks like useEffect().

1. **List the Sequence of Steps in Rendering a Component**

Here’s the typical sequence React follows in Class Components from creation to removal:

* **Mounting Phase (initial render):**

1. constructor()
2. getDerivedStateFromProps()
3. render()
4. componentDidMount()

* **Updating Phase (on props/state change):**

1. getDerivedStateFromProps()
2. shouldComponentUpdate()
3. render()
4. getSnapshotBeforeUpdate()
5. componentDidUpdate()

* **Unmounting Phase:**

1. componentWillUnmount()

**Conclusion**

Understanding the React component lifecycle is crucial for writing robust, responsive, and efficient applications. Lifecycle methods allow developers to manage component behavior during creation, updates, and removal. With the right use of these hooks, especially in class components or with useEffect() in function components, developers can optimize performance, manage side effects, and keep code clean and predictable.