React

1. Explain the Need and Benefits of Component Lifecycle

In React, components go through a series of stages from their creation to destruction. This progression is known as the **component lifecycle**. Understanding the lifecycle is essential because it allows developers to control what happens at each stage of a component's existence — such as initialization, updates, and cleanup.

Need for Component Lifecycle:

- To execute specific logic during certain phases (e.g., fetching data after the component mounts).
- To optimize performance by avoiding unnecessary re-renders.
- To handle side effects like timers, API calls, and event listeners appropriately.

Benefits:

- **Efficient resource management:** Cleanup operations (like clearing timers or unsubscribing from listeners) prevent memory leaks.
- **Fine-grained control:** Developers can hook into each phase of the component's life to perform required tasks.
- **Improved debugging and maintenance:** Knowing the flow helps in understanding unexpected behavior and fixing bugs efficiently.

---

2. Identify Various Lifecycle Hook Methods

In class-based components, React provides built-in lifecycle methods categorized as:

1. **Mounting (When the component is being inserted into the DOM):**
    - constructor()
    - static getDerivedStateFromProps()
    - render()
    - componentDidMount()

2. **Updating (When the component is being re-rendered due to state or props change):**
    - static getDerivedStateFromProps()
    - shouldComponentUpdate()
    - render()
    - getSnapshotBeforeUpdate()
    - componentDidUpdate()
3. **Unmounting (When the component is being removed from the DOM):**
    - componentWillUnmount()
4. **Error Handling:**
    - componentDidCatch()
    - getDerivedStateFromError()

In functional components, these lifecycle phases are handled using **Hooks**, primarily:

- useEffect()
- useLayoutEffect()
- useState()
- useRef()
- useMemo() (for optimization)

---

3. List the Sequence of Steps in Rendering a Component

For **class components**, the typical sequence when a component is rendered initially (mounting phase) is:

1. **constructor()** – Initializes state and binds methods.
2. **getDerivedStateFromProps()** – Syncs state with props, if needed.

3. **render()** – Returns the JSX to be rendered in the DOM.

4. **componentDidMount()** – Invoked after the component is mounted. Ideal for data fetching, subscriptions, etc.

During **updates**, when props or state change, the order is:

1. **getDerivedStateFromProps()**

2. **shouldComponentUpdate()** – Decides whether a re-render is needed.

3. **render()**

4. **getSnapshotBeforeUpdate()**

5. **componentDidUpdate()**

When a component is removed:

- **componentWillUnmount()** – Used to clean up resources like event listeners or timers.