1. Explain the Need and Benefits of Component Life Cycle

Need:

- In modern UI frameworks like React or Angular, components go through various **stages** from creation to destruction.
- Managing behavior during these stages ensures **predictability, reusability, and performance optimization**.

Benefits:

- 📦 **Efficient Resource Management:** Initialize or clean up resources like event listeners, timers, or network requests.
- 🔄 **Dynamic UI Updates:** Perform actions (e.g., data fetching) at the right time—before or after rendering.
- 🧪 **Debugging Made Easier:** Know when and why a component is behaving in a certain way.
- ♻️ **Reusability and Maintainability:** Lifecycle methods make components modular and easier to update or reuse.
- 🚀 **Performance Optimization:** Avoid unnecessary renders or DOM manipulations by controlling component updates.

---

2. Identify Various Life Cycle Hook Methods

Here are common **lifecycle methods** for **React (Class Components)** and **React (Functional Components with Hooks)**:

🏗️ ***React Class Component Lifecycle Hooks:***

1. **Mounting (component is being created and inserted)**
    - constructor()
    - static getDerivedStateFromProps()
    - render()
    - componentDidMount()
2. **Updating (props or state change)**
    - static getDerivedStateFromProps()
    - shouldComponentUpdate()
    - render()
    - getSnapshotBeforeUpdate()
    - componentDidUpdate()
3. **Unmounting (component is removed)**

    - componentWillUnmount()

4. **Error Handling**
    - componentDidCatch()
    - getDerivedStateFromError()

🧠 ***React Functional Component Lifecycle Equivalents (with Hooks):***

- useEffect() – runs after render, can mimic componentDidMount, componentDidUpdate, and componentWillUnmount by using dependencies.
- useLayoutEffect() – similar to useEffect but fires **before** the browser paints.
- useRef() – for persisting values across renders (like instance variables).
- useState() – for managing state in functional components.

---

3. List the Sequence of Steps in Rendering a Component

Here's the typical rendering flow in **React Class Components**:

🧱 **Mounting Phase (Initial Render):**

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

🔁 **Updating Phase (Re-render):**

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

🧹 **Unmounting Phase:**

- componentWillUnmount()