### 🔄 Need and Benefits of Component Lifecycle

React lifecycle methods help manage a component’s behavior across its existence:

* **Structured Control**: Enables code execution at specific stages—creation, update, and removal.
* **Data Fetching**: Use componentDidMount() or componentDidUpdate() to fetch data at the right time.
* **Performance Optimization**: Prevent unnecessary re-renders using shouldComponentUpdate().
* **Resource Management**: Clean up timers, listeners, or subscriptions with componentWillUnmount().
* **Dynamic UI Updates**: Respond to prop or state changes in a controlled way.

### 🧪 Various Lifecycle Hook Methods

Lifecycle hooks are grouped into three phases:

#### 1. ****Mounting (Component is created and inserted into DOM)****

* constructor() – Initializes state and binds methods.
* getDerivedStateFromProps() – Syncs state with props.
* render() – Returns JSX to be displayed.
* componentDidMount() – Executes after first render (ideal for API calls).

#### 2. ****Updating (Component re-renders due to state/prop changes)****

* getDerivedStateFromProps() – Called again before re-render.
* shouldComponentUpdate() – Decides if re-render is needed.
* render() – Re-renders JSX.
* getSnapshotBeforeUpdate() – Captures info before DOM updates.
* componentDidUpdate() – Executes after DOM updates.

#### 3. ****Unmounting (Component is removed from DOM)****

* componentWillUnmount() – Cleanup tasks before removal.

### 🧭 Sequence of Steps in Rendering a Component

Here’s the typical order during mounting and updating:

#### 🏗️ Mounting Phase

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

#### 🔁 Updating Phase

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

#### 🧹 Unmounting Phase

1. componentWillUnmount()