**Need and Benefits of Component Life Cycle**

Component lifecycle is essential for managing the **creation**, **update**, and **destruction** of components in a predictable and controlled manner. It helps developers hook into key phases of a component’s existence to perform tasks like:

* Initializing data
* Making API calls
* Optimizing performance
* Cleaning up resources

**Benefits:**

* **Efficient Resource Management:** Perform actions at specific points (e.g., setup on mount, cleanup on unmount).
* **Better Debugging and Maintenance:** Clearly defined stages help isolate and fix issues.
* **Optimized Performance:** Update only when necessary, improving rendering efficiency.
* **Improved Code Organization:** Separation of logic into appropriate lifecycle stages.

**Various Life Cycle Hook Methods**

Different frameworks have different hooks, but here's a general overview using **React (class-based)** and **Angular** for context:

|  |  |  |
| --- | --- | --- |
| Phase | Hook Method | Purpose |
| Mounting | constructor() | Initialize state and bind methods |
|  | componentDidMount() | Run after component is added to DOM (e.g., API calls) |
| Updating | shouldComponentUpdate() | Control re-rendering |
|  | componentDidUpdate() | Run after updates (e.g., DOM manipulations or new props) |
| Unmounting | componentWillUnmount() | Cleanup (e.g., remove event listeners, clear timers) |
| Error Handling | componentDidCatch() | Catch and handle errors in child components |

**Sequence of Steps in Rendering a Component**

For **React Class Components**, the typical rendering sequence is:

1. **constructor()**
   * Initializes state and props.
2. **static getDerivedStateFromProps()** *(optional)*
   * Sync state with props.
3. **render()**
   * Returns the JSX (UI).
4. **componentDidMount()**
   * Invoked after the component is inserted into the DOM.