**MODULE: 4 (List and Hooks)**

**• Explain Life cycle in Class Component and functional component with Hooks**

**Ans:-**

A lifecycle that encompasses their birth, growth, and eventual removal. Let’s explore the lifecycle in both class components and functional components with Hooks.

Lifecycle of ComponentsEach component in React has a lifecycle which you can monitor and manipulate during its three main phases.

**Component LifeCycle In React in Class Component:-**

Constructor: The constructor is called when an instance of the component is created. It initializes the component’s state and binds event handlers.

**render():** The render() method generates the component’s UI based on its current state and props.

* **Didmout Phase:**

**componentDidMount():-**This method is invoked after the component is added to the DOM. It’s a good place to set up timers, network requests, or subscriptions.

* **Updating Phase:**

**componentDidUpdate():-****(**prevProps, prevState**):** Called after the component updates (when new props or state are received). Useful for handling side effects or re-rendering based on changes.

shouldComponentUpdate(nextProps, nextState): Determines whether the component should re-render. Optimize performance by preventing unnecessary updates.

* **Unmounting Phase:**

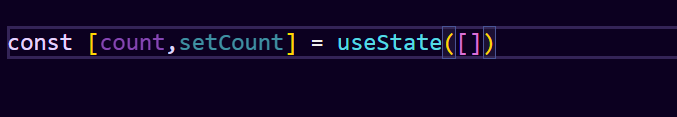
**componentWillUnmount():-**Invoked just before the component is removed from the DOM. Clean up any resources (e.g., cancel timers, unsubscribe from subscriptions).

* **Functional Components with Hooks:**

**useState Hook:**

Allows functional components to manage local state.

Example:

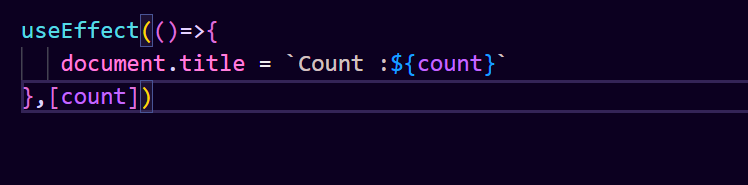


**useEffect Hook:**

Replaces lifecycle methods like componentDidMount, componentDidUpdate, and componentWillUnmount.

Handles side effects (e.g., data fetching, subscriptions).

Example:

****

**useContext**: Access context values.

**useReducer**: Manage complex state logic.

**useMemo**: Optimize expensive computations.

**useCallback**: Memoize functions to prevent unnecessary re-renders.

Remember that Hooks allow functional components to handle state and side effects more elegantly, making them a powerful alternative to class components.