ReactJS about component life cycle

1)Explain the need and Benefits of component life cycle

In React, every component undergoes a lifecycle – a series of phases from when it is created and added to the DOM to when it is updated or removed. React provides special lifecycle methods (class-based) or hooks (function-based) that allow developers to run custom logic at specific points of this lifecycle.

* To control behavior of a component at different stages (creation, update, deletion).
* To fetch data (e.g., from APIs) only after the component mounts.
* To improve performance by controlling updates.
* To clean up resources (e.g., timers, event listeners) when a component is removed.
* To synchronize state with props or handle side effects properly.

Key Benefits:

* Organized Code: Lifecycle methods separate logic for mount, update, and unmount phases.
* Performance Optimization: Methods like shouldComponentUpdate can prevent unnecessary renders.
* Controlled Updates: Lets us decide how the UI should react to prop or state changes.
* Cleanup Support: componentWillUnmount helps avoid memory leaks (clearing intervals, listeners, etc.)
* Data Handling: Fetch data only when necessary using componentDidMount.

2) Identify various life cycle hook methods

The lifecycle is mainly divided into three phases:

1. Mounting Phase (Component is being created and inserted into the DOM)

These methods are called in this order:

constructor(props)

* Called when the component is initialized.
* Used to set the initial state and bind methods.
* Should always call super(props) before using this.

getDerivedStateFromProps(props, state)

* Called right before render().
* Used to update the state based on changes in props.
* It is a static method, so it doesn’t have access to this.

render()

* Required method in every class component.
* Returns JSX that describes what to render on the screen.
* Should be pure and free of side effects.

componentDidMount()

* Called after the component is rendered to the DOM.
* Best place to:
  + Fetch data from APIs
  + Add event listeners
  + Start timers
* Only called once in the component’s lifetime.

2. Updating Phase (Component is re-rendered due to changes in props or state)

These methods are called in this order when the component updates:

getDerivedStateFromProps(props, state)

* Called again just before re-rendering.
* Can be used to update state in response to prop changes.

shouldComponentUpdate(nextProps, nextState)

* Called before rendering.
* Allows you to optimize performance by preventing unnecessary re-renders.
* Return false to skip render() and subsequent lifecycle methods.

render()

* Re-renders the component’s UI based on updated state or props.

getSnapshotBeforeUpdate(prevProps, prevState)

* Called right before the changes are applied to the DOM.
* Useful to capture some DOM information (like scroll position) before update.
* The value returned here is passed to componentDidUpdate.

componentDidUpdate(prevProps, prevState, snapshot)

* Called after the component updates and re-renders in the DOM.
* Best place to:
  + Perform side effects based on prop/state changes
  + Handle external libraries
  + Work with updated DOM

3. Unmounting Phase (Component is removed from the DOM)

componentWillUnmount()

* Called just before the component is destroyed and removed from the DOM.
* Use this to:
  + Clear timers
  + Remove event listeners
  + Cancel network requests
  + Clean up subscriptions
* Prevents memory leaks in your app.

4. Error Handling Phase (If something goes wrong)

These are special lifecycle methods for catching errors in rendering or lifecycle methods of child components:

componentDidCatch(error, info)

* Catches errors thrown in child components.
* Can be used to log error messages or show fallback UI.

getDerivedStateFromError(error)

* Used to update state to display fallback UI in case of an error.

3) List the sequence of steps in rendering a component

A. Mounting (Initial Render)

1. constructor()  
   → Initializes the component’s state and binds methods.
2. getDerivedStateFromProps()  
   → Static method to update state from props (if needed).
3. render()  
   → Returns JSX that React uses to render UI.
4. DOM is updated
5. componentDidMount()  
   → Runs after DOM is mounted. Best for API calls or subscriptions.

B. Updating (on state or props change)

1. getDerivedStateFromProps()
2. shouldComponentUpdate()  
   → Can return false to stop re-rendering.
3. render()
4. getSnapshotBeforeUpdate()  
   → Optional. Used to capture info from the DOM.
5. componentDidUpdate()  
   → Use for side-effects (like updating external data).

C. Unmounting

1. componentWillUnmount()  
   → Cleanup logic goes here (clearing timers, listeners, etc.)