MODULE:10 List and Hooks

- Explain Life cycle in Class Component and functional component with Hooks

Ans:-In React, components are the building blocks of a user interface. There are two main types of components: class components and functional components. Both types of components have lifecycle methods or hooks that allow you to perform actions at different stages of the component's existence. In this response, I'll provide an overview of the lifecycle methods for both class components and hooks for functional components.

Class Components:

Class components in React have a set of lifecycle methods that are invoked at different points during the component's life cycle. The lifecycle can be divided into three main phases:

Mounting:

constructor(): This is called when an instance of the component is being created. It is often used for initializing state and binding event handlers.

render(): This method is responsible for rendering the component. It must be implemented in every class component.

componentDidMount(): Invoked immediately after a component is inserted into the DOM. It's a good place to perform side effects like network requests.

Updating:

shouldComponentUpdate(nextProps, nextState): This method is called before rendering when new props or state are received. It allows you to control whether the component should update or not.

render(): The component is re-rendered.

componentDidUpdate(prevProps, prevState): Invoked immediately after updating occurs. It's a good place to perform side effects based on the changed props or state.

Unmounting:

componentWillUnmount(): Invoked immediately before a component is unmounted and destroyed. It's a good place to clean up resources or subscriptions.

Functional Components with Hooks:

Functional components can now use hooks to manage state and side effects, allowing them to have similar lifecycle functionality. The key hooks are:

Mounting:

useState: Hook for adding state to functional components.

useEffect (() { , []): Hook for performing side effects in function components. The empty dependency array [] ensures that the effect runs only once after the initial render, simulating componentDidMount.

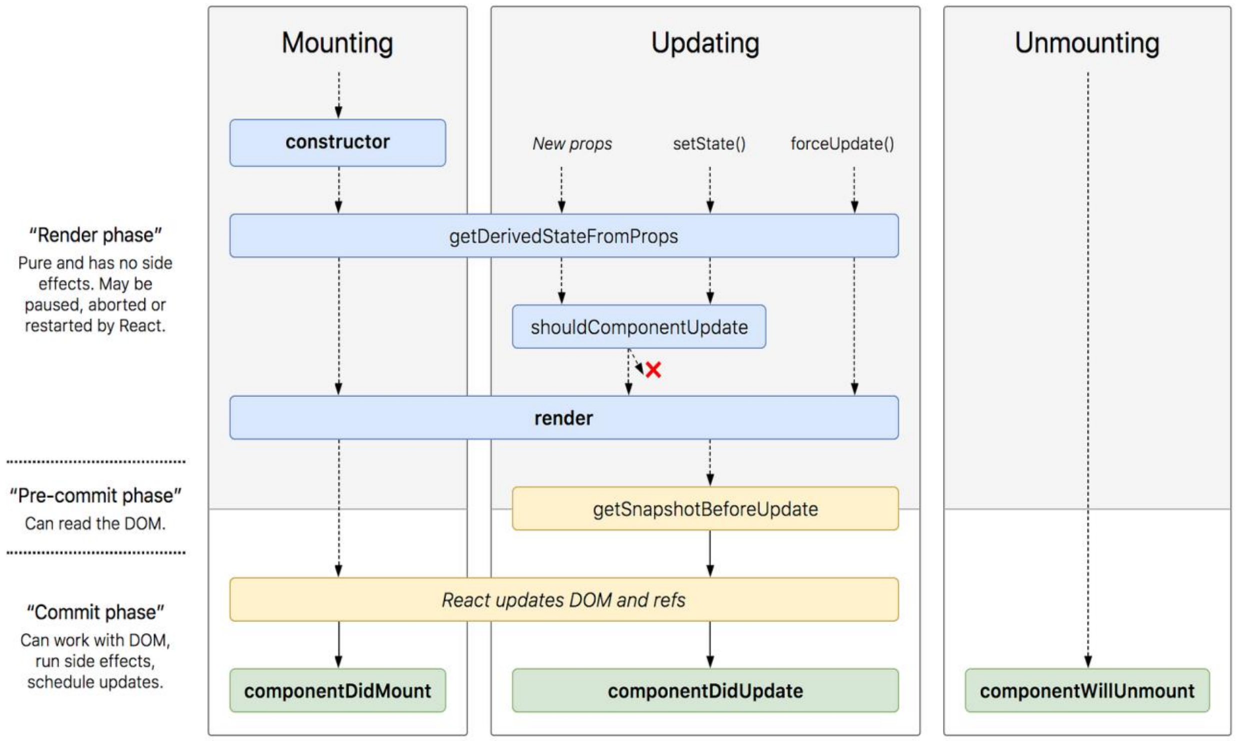
Updating:

useState: Continues to be used for managing state.

useEffect(() { , [dependencies]): The effect runs when any of the dependencies change, simulating the behavior of componentDidUpdate.

Unmounting:

; }, []): The cleanup function inside useEffect will be called when the component is unmounted, simulating the behavior of componentWillUnmount.


Mounting
Updating
Unmounting
constructor
New props
setState()
forceUpdate()
"Render phase"
Pure and has no side effects. May be paused, aborted or restarted by React.
getDerivedStateFromProps
shouldComponentUpdate
render
"Pre-commit phase"
Can read the DOM.
getSnapshotBeforeUpdate
"Commit phase"
Can work with DOM, run side effects, schedule updates.
React updates DOM and refs
componentDidMount
componentDidUpdate
componentWillUnmount