***List and Hooks***

***Q. Explain Life Cycle in Class Component and***

***functional component with Hooks.***

*🡪 In React.js, component lifecycle refers to the series of events that occur during the lifespan of a component, from its creation to its destruction. These events allow developers to hook into specific moments in a component's lifecycle and perform actions such as initialization, updating, and cleanup.*

*There are different lifecycle phases in both class components and functional components with Hooks. Let's explore each:*

*Class Components*

*1. Mounting Phase*

* *constructor() : This is called when a component is first initialized. It's typically used for initializing state and binding event handlers*.
* *render() : This method renders the component’s UI.*
* *componentDidMount() : This is invoked immediately after a component is mounted (inserted into the DOM). It's commonly used to perform tasks such as fetching data from APIs or setting up event listeners.*

*2. Updating Phase*

* *shouldComponentUpdate() : This is invoked before rendering when new props or state are received. It allows developers to control if the component should re-render based on the changes.*
* *render(): The component re-renders based on new props or state.*
* *componentDidUpdate(): This is* *called immediately after the component updates. It's often used for performing side effects after a component re-renders, such as fetching new data.*

*3. Unmouting Phase*

* *componentWillunmount() : This is called just before the component is unmounted and destroyed. It's used for cleanup tasks such as removing event listeners or canceling subscriptions*.

*Functional Components With Hooks*

*1. Mounting Phase*

* *useState() : This Hook allows functional components to manage local state.*
* *useEffect() : This Hook is similar to ‘componentDidMount’ and ‘componentDidUpdate’. It is called after every render and can perform side effects such as data fetching , DOM manipulation, or subscription setup.*

*2. Updating Phase*

* *useState() : The component re-renders based on state updates.*
* *useEffect() : The ‘useEffect’ Hook is called after every render, allowing for side effects to be performed based on state or prop changes.*
* *Optionally, you can use the ‘useMemo()’ and ‘useCallback()’ Hooks to optimize performance by memoizing values or callbacks that don't need to be recomputed on every render*.

*3. Unmounting Phase*

* *useEffect()cleanup function : If ‘useEffect()’ returns a*

*function, React will call this function when the component is unmounted. This is where you can perform cleanup tasks like removing event listeners or subscriptions*.

*In summary, both class components and functional components with Hooks provide lifecycle methods or Hook equivalents to perform actions at different stages of a component's lifecycle. Functional components with Hooks offer a more concise and expressive way to manage component lifecycle and state, while class components provide a more traditional approach.*