**List and Hooks**

1. Explain Life cycle in Class Component and functional component with Hooks?

* In React, components are the building blocks of user interfaces. React components have a life cycle, which refers to the different phases a component goes through from its creation to its removal from the DOM. In class components, the life cycle is managed using methods defined in the component class, while in functional components, the life cycle is handled using hooks.
* Let's start with the life cycle of a class component:
* 1. \*\*Mounting Phase\*\*:
* - `constructor()`: This is the first method called when an instance of the component is created. It is used to initialize the component's state and bind event handlers.
* - `render()`: This method is responsible for rendering the component's JSX markup.
* - `componentDidMount()`: This method is called immediately after the component is mounted to the DOM. It is commonly used for side effects like data fetching, subscriptions, or manually interacting with the DOM.
* 2. \*\*Updating Phase\*\*:
* - `render()`: This method is called whenever the component's state or props change, and it is responsible for updating the component's JSX markup.
* - `componentDidUpdate(prevProps, prevState)`: This method is invoked after an update occurs. It allows you to perform side effects when the component updates. You can compare the previous props and state with the current ones to determine the necessary actions.
* 3. \*\*Unmounting Phase\*\*:
* - `componentWillUnmount()`: This method is called just before the component is removed from the DOM. It is used to perform cleanup tasks like canceling network requests or removing event listeners.

Now, let's discuss the life cycle of a functional component using hooks:

* 1. \*\*Mounting and Updating Phases\*\*:
* - `useState()`: This hook allows functional components to have state. It returns a state variable and a function to update that variable. It is typically used multiple times within a component to manage different pieces of state.
* - `useEffect()`: This hook is used to perform side effects in functional components. It combines the functionality of `componentDidMount`, `componentDidUpdate`, and `componentWillUnmount` from class components. You can use `useEffect` to handle side effects like data fetching, subscriptions, or interacting with the DOM. It accepts a callback function and an optional array of dependencies that determine when the effect should be re-run.
* The `useEffect()` hook can be used in different ways to replicate the behavior of different class component life cycle methods:
* - To replicate `componentDidMount`, pass an empty array as the second argument to `useEffect()`. This ensures that the effect runs only once after the initial render.
* - To replicate `componentDidUpdate`, pass the necessary dependencies as an array to the second argument of `useEffect()`. The effect will re-run whenever any of the dependencies change.
* - To replicate `componentWillUnmount`, return a cleanup function from the effect. This cleanup function will be called before the component unmounts or before the effect is re-run.
* These are the basics of the life cycle in both class components and functional components with hooks. React hooks provide a simpler and more flexible way to manage component life cycle and state in functional components.