Explain Life cycle in Class Component and functional component with Hooks

➔ In React, components are the building blocks of user interfaces. There are two main types of components: class components and functional components with Hooks. Each type has its own way of managing lifecycle events.

Class Components:

1. Mounting Phase:

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

- render(): This method is mandatory and is responsible for rendering JSX elements to the DOM.

- componentDidMount(): This is invoked immediately after a component is mounted (inserted into the tree). It's often used for fetching data from APIs or setting up subscriptions.

2. Updating Phase:

- shouldComponentUpdate(nextProps, nextState): This method is invoked before rendering when new props or state are being received. It's used to optimize performance by determining whether the component needs to re-render.

- render(): Re-renders the component with updated state or props.

- componentDidUpdate(prevProps, prevState): This is called immediately after updating occurs. It's useful for interacting with the DOM or performing side effects.

3. Unmounting Phase:

- **componentWillUnmount():** This is invoked immediately before a component is unmounted and destroyed. It's used for cleanup tasks like unsubscribing from event listeners or canceling network requests.

Functional Components with Hooks:

1. Mounting and Updating Phase:

- **useState():** Hook for adding state to functional components.

- **useEffect():** Hook that combines the functionality of componentDidMount, componentDidUpdate, and componentWillUnmount. It runs after every render and can perform side effects such as data fetching, DOM manipulation, or subscribing to events.

Example of functional component with Hooks:

```
import React, { useState, useEffect } from 'react';

function FunctionalComponent() {
  const [count, setCount] = useState(0);
  // Similar to componentDidMount and componentDidUpdate:
  useEffect(() => {
    // Update the document title using the browser API
    document.title = You clicked ${count} times;
  });
  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );}
```