1. What are some differences between class and functional components?

Ans:

Class Components:

Definition: Defined using ES6 class syntax, extending React.Component.

State Management: Use this.state and this.setState to manage and update state.

Lifecycle Methods: Use methods like componentDidMount, componentDidUpdate, and componentWillUnmount.

Bindings: Often require binding of event handlers to this.

Readability: Can be less concise and harder to read, especially with complex logic.

Performance: May have more overhead compared to functional components due to the class structure.

Functional Components:

Definition: Defined as plain JavaScript functions returning JSX.

State Management: Use hooks like useState to manage state.

Lifecycle Methods: Use hooks like useEffect to handle side effects and mimic lifecycle methods.

Bindings: No need for binding as functions do not have this.

Readability: Typically simpler, more concise, and easier to read.

Performance: Generally lighter and potentially more performant due to the absence of class overhead.

2. Explain what lifecycle is in a simple way. How do you manage it in class and functional components?

Ans:

Lifecycle in Simple way:

The lifecycle of a component in React refers to the different stages a component goes through from its creation to its removal from the DOM. These stages include mounting (being added to the DOM), updating (re-rendering due to changes in props or state), and unmounting (being removed from the DOM).

Managing Lifecycle in Class Components

In class components, you manage the lifecycle using specific methods:

Mounting: Handled by methods like constructor and componentDidMount, which are used to set up the initial state and perform actions after the component is inserted into the DOM.

Updating: Managed with componentDidUpdate, which allows you to respond to changes in props or state after the component has re-rendered.

Unmounting: Managed with componentWillUnmount, used for cleanup tasks like clearing timers or cancelling network requests before the component is removed from the DOM.

Managing Lifecycle in Functional Components

In functional components, you manage the lifecycle using hooks:

Mounting and Updating: Managed with useEffect. By using different dependency arrays, you can control when the effect runs, similar to componentDidMount and componentDidUpdate.

Unmounting: Also managed within useEffect. By returning a cleanup function from useEffect, you can perform cleanup tasks, similar to componentWillUnmount.

3. Explain immutability in one sentence.

Ans: Immutability means that once an object or data structure is created, it cannot be changed, and any modifications result in the creation of a new object.