1. What is useEffect? What are the different behaviors of useEffect? What is a dependency array?

Ans: useEffect is a tool in React that allows you to run some code (like fetching data or updating the page title) at specific times during the life of your component.

Different Behaviors of useEffect

No Dependency Array: When useEffect is called without a dependency array, it runs after every render of the component.

Empty Dependency Array: When useEffect is called with an empty dependency array, it runs only once, after the initial render (componentDidMount).

Dependencies Specified: When useEffect is called with a dependency array containing variables, it runs only when any of the dependencies change.

Dependency Array

The dependency array is a list of dependencies that useEffect watches for changes. If any of the dependencies in the array change between renders, the effect will be re-executed. The dependency array ensures that the effect is executed only when it is necessary, optimizing performance and preventing unnecessary executions.

2. What is useRef and when do you want to use it?

Ans: useRef is a hook in React that helps you keep track of values or access HTML elements directly without causing your component to re-render.

When to Use useRef

Accessing DOM Elements: If you need to directly interact with an HTML element (like an input field), you can use useRef to get a handle on that element.

Storing Mutable Values: If you want to keep track of a value that you can change without re-rendering the component, useRef is useful.

Remembering Previous Values: If you need to remember a value between renders without causing a re-render, you can use useRef.

3. How to reuse hook logic in React?

Ans: To reuse hook logic in React, you can create custom hooks, which are JavaScript functions that start with "use" and encapsulate logic involving React's built-in hooks like useState and useEffect. Custom hooks allow you to extract and share logic between components, enhancing code reusability, readability, and testability. For example, a custom hook named useFetch can handle data fetching and state management for loading and error states, which can then be easily used in multiple components to standardize and simplify data fetching logic across your application.