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

- useEffect is a React hook that allows you to perform side effects in functional components. Side effects are operations that interact with external resources that are beyond the local scope of a components' rendering logic, such as fetching data from an external API, or cleaning up resources when a component is unmounted.
- The behavior of useEffect depends on the presence and content of the dependency array. If the dependency array is not present, the useEffect runs in the initial rendering and each time the component re-render. If the empty dependency array is present, the useEffect runs only once in the initial rendering state. If a dependency array with value is present, the useEffect runs in the initial rendering and whenever any of the values of the dependency array change.
- The dependency array is the optional second argument of the useEffect which tells React when to re-render the effect. It is an array of values that the effect depends on.

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

- useRef is a react hook that allows you to reference a value that persists across renders without causing a re-render when it changes. We use it when we want to directly access the DOM element such as focusing an input field. We also use it when we want to store mutable values, such as timers or interval IDs that persist across render but don't trigger a re-render.

3. **How to reuse hook logic in React?**

- We can create custom hooks to reuse hook logic. Custom hooks allow you to extract and share logic across multiple components. To create and use custom hook, we need to first create a new function that use React hooks like useState, useEffect, etc and encapsulates the logic that you want to reuse. Then we can use the custom hook by calling the custom hook inside the functional components to reuse the logic

useR