* What is the difference between Named Export, Default export and \* as export?

Named imports in JavaScript make specific functions, variables, or objects from a module using their names available for any other modules while Default imports let you import a single entity, often the main export of a module which we could also give it a different name. \* as export will make the whole module functionality available.

* Props in react

Props which stand short for properties are a way to pass data from a parent component to a child component. They are not being able to be changed but they make components reusable and connective by sharing their data and functionality to be passed in as arguments.

* Map, Filter Function JavaScript

In JavaScript, map transforms each element of an array using a provided function and returns a new array of the transformed elements. filter creates a new array containing elements that pass a test specified by a provided function, excluding those that don't meet the criteria.

* Destructuring in ES6

Destructuring allowing us to choose values from properties or lists/arrays from objects into distinct variables using syntax like array destructuring or object destructuring. It grants straight forward way to extract and assign values, which makes developing simple and clear.

* What are React Hooks?

React Hooks are functions that allow functional components in React to use state and other React features without writing a class. They enable developers to add state, side-effects, and other React features to functional components, improving code organization and reusability.

* Explain useState Hook?

The useState hook in React allows functional components to manage state. It returns a stateful value and a function to update it. By invoking this function, components can re-render with updated state, simplifying state management in functional components compared to class components.

* Why do we need a useEffect Hook?

The useEffect hook in React enables functional components to perform side effects, such as data fetching, DOM manipulation, or subscriptions, after rendering. It replaces lifecycle methods in class components, ensuring that side effects are executed consistently and efficiently in response to component updates.

* When is the useEffect hook called?

The useEffect hook in React is called after the initial render and after every update to the component. It allows functional components to execute side effects, such as data fetching or DOM manipulation, in response to component mounts, updates, and unmounts, ensuring consistent behavior across renders.

* Why do we write our API in useEffect?

We write API calls in useEffect to execute them after the component renders, ensuring data fetching occurs at the appropriate lifecycle stage. This prevents unnecessary re-renders and optimizes performance by fetching data only when needed, improving user experience and reducing potential API request errors.

* What is the difference between async and sync functions in JavaScript?

Async functions in JavaScript allow non-blocking, asynchronous operations, returning Promises. They use async and await to handle asynchronous operations elegantly. Sync functions execute code sequentially, blocking further execution until the current operation completes, without returning Promises or using async/await.

* Why do we use async functions for fetching data using an API?

We use async functions for fetching data using an API to perform non-blocking, asynchronous operations. This allows other code to run while waiting for the API response, enhancing performance and responsiveness. async/await simplifies handling Promises, making asynchronous data fetching more readable and maintainable.

* Why and How does react component update?

React components update when their state or props change. Changes trigger a re-render of the component, updating the UI to reflect the new data or props. React efficiently compares the virtual DOM with the previous state, rendering only the changed elements, optimizing performance and user experience.