1) What is Redux?

Ans:

=> Redux is a predictable state container for JavaScript applications, primarily used with libraries like React or Angular for building user interfaces.

It helps manage the state of an application in a predictable and efficient manner.

Redux follows the principles of a unidirectional data flow architecture, which means that data flows in a single direction through the application.

At its core, Redux stores the entire state of your application in a single JavaScript object, often referred to as the "store".

One of the key benefits of Redux is its ability to manage complex state in large-scale applications, making it easier to debug and trace how data changes over time.

Overall, Redux simplifies state management by providing a clear and centralized approach, which helps in writing scalable and maintainable JavaScript applications.

2) What is Redux Thunk used for?

Ans:

=> Redux Thunk is a middleware for Redux that allows you to write action creators that return functions instead of plain action objects.

These functions can perform asynchronous operations, such as fetching data from an API, and dispatch regular actions with the fetched data once the asynchronous operation is complete.

Without Redux Thunk, Redux action creators are typically synchronous functions that return plain action objects.

While Redux itself is synchronous, many real-world applications require handling asynchronous operations, such as making HTTP requests or interacting with a database.

Redux Thunk provides a convenient way to handle asynchronous logic in Redux applications while keeping actions and reducers pure and synchronous.

3) What is Pure Component? When to use Pure Component over Component?

Ans:

=> A Pure Component is a concept in React that represents a component class that implements the shouldComponentUpdate() lifecycle method with a shallow prop and state comparison.

This means that a Pure Component will only re-render if its props or state have changed compared to the previous render.

Pure Components provide a performance optimization by preventing unnecessary re-renders when the component's input data hasn't changed.

Pure Components indiscriminately in all components may not always be beneficial, especially for components that frequently receive new props or have complex rendering logic.

4) What is the second argument that can optionally be passed tosetState and what is its purpose?

Ans:

=> In React, the setState() function is used to update the state of a component.

It can optionally take a second argument, which is a callback function that gets executed after the state has been updated and the component has been re-rendered.

The purpose of this callback function is to perform some action after the state update is complete and the component has been re-rendered.

This can be useful for scenarios where you need to execute code that depends on the updated state or the DOM.