# **MODULE: 13 React – Applying Redux**

Q1) What is Redux?

Ans: Redux is a predictable state container for JavaScript applications, most commonly used with libraries like React or Angular for managing application state in a consistent way. It helps developers write applications that behave consistently, run in different environments (client, server, native), and are easy to test.

Some benefits

-> The state of the entire application is stored in a single object tree within a single store. This makes it easier to manage and debug the state of your application.

-> The only way to change the state is to emit an action, an object describing what happened. This ensures that neither the view nor the network callbacks will ever write directly to the state.

-> To specify how the state tree is transformed by actions, you write pure reducers. Reducers are just pure functions that take the previous state and an action, and return the next state.

Q2) What is Redux Thunk used for?

Ans: Redux Thunk is a middleware for Redux that allows you to write action creators that return a function instead of an action. This is particularly useful for handling asynchronous operations in Redux, such as fetching data from an API or performing side effects.

Some benefits

-> Redux Thunk allows action creators to return a function that can dispatch actions asynchronously. This is essential for operations that involve API calls, timers, or other asynchronous tasks.

-> You can dispatch actions after a certain delay or condition is met. This is useful for scenarios like debouncing user input or waiting for an animation to complete before dispatching an action.

-> Thunks can dispatch actions conditionally based on the current state or other factors, allowing for more complex state logic.

Q3) What is Pure Component? When to use Pure Component over Component?

Ans: A Pure Component in React is a component that implements a shallow comparison on the component’s props and state in the “shouldComponentUpdate” lifecycle method. This means that a Pure Component will only re-render if it detects changes in the props or state that are passed to it, making it a performance optimization technique for React applications.

-> Pure Component performs a shallow comparison of props and state. This means it only checks if the first-level properties of the objects have changed, rather than performing a deep comparison of nested objects.

-> Pure Component automatically implements “shouldComponentUpdate” with a shallow comparison, so you don't have to write it manually.

When to Use Pure Component Over Component

Use Pure Component when you want to optimize the performance of your component by avoiding unnecessary re-renders. This is particularly useful in large applications where re-rendering can be costly.

Pure Component is most effective when your props and state are simple and flat. Shallow comparison works well in these cases, but may not be as effective with deeply nested objects or arrays.

If your component's props or state rarely change, using Pure Component can help reduce the number of renders, leading to better performance.

Q4) What is the second argument that can optionally be passed to setState and what is its purpose?

Ans: In React, the setState method can accept a second optional argument, which is a callback function. This callback is executed once the state has been updated and the component has re-rendered.

The callback function can be used to execute any code that depends on the new state. Since setState is asynchronous, using the callback ensures that the state update has been applied before running the code.

If you need to perform side effects (e.g., API calls, updating local storage, interacting with third-party libraries) based on the updated state, the callback provides a safe place to do so.

The callback ensures that you are working with the latest state, avoiding potential issues with stale state.