**MODULE: 13 React – Applying Redux**

**Question – 1 : What is Redux?**

**Ans.**

Redux is an open-source JavaScript library used to manage application state. React uses Redux for building the user interface. It was first introduced by **Dan Abramov** and **Andrew Clark** in **2015**.

React Redux is the official React binding for Redux. It allows React components to read data from a Redux Store, and dispatch **Actions** to the **Store** to update data. Redux helps apps to scale by providing a sensible way to manage state through a unidirectional data flow model. React Redux is conceptually simple. It subscribes to the Redux store, checks to see if the data which your component wants have changed, and re-renders your component.

Redux was inspired by Flux. Redux studied the Flux architecture and omitted unnecessary complexity.

**Question – 2 :** What is Redux Thunk used for?

**Ans.**

Redux Thunk middleware allows you to write action creators that return a function instead of an action. The thunk can be used to delay the dispatch of an action, or to dispatch only if a certain condition is met. The inner function receives the store methods dispatch and getState as parameters.

**Question - 3 : What is Pure Component? When to use Pure Component over Component?**

**Ans.**

**Pure Component:**

Pure components are  specifically designed for performance optimization by minimizing unnecessary renders through automatic prop comparison. The second argument that can optionally be passed to setState is a callback function which gets called immediately after the setState is completed and the components get re-rendered.

If you want your program to update the value of a state using setState and then perform certain actions on the updated value of the state then you must specify those actions in a function which should be the second argument of the setState. If we do not do so then those actions will be performed on the previous value of state because of the asynchronous nature of setState. Specifically, a Pure Component implements a **“shouldComponentUpdate”** method with a shallow prop and state comparison.

The **“shouldComponentUpdate”** method in a Pure Component automatically checks if there are any changes in props or state by performing a shallow comparison of the new and old values. If there are no changes, React prevents the component from re-rendering, thus optimizing performance by avoiding unnecessary re-renders.

**To use Pure Component over Component:**

1. **Performance Optimization :**

If your component's render method is a pure function of props and state and doesn't rely on any other external factors, using Pure Component can improve performance by reducing unnecessary re-renders.

1. **Complex Components:**

When dealing with complex components or components with many child components, using Pure Components can help reduce the number of re-renders and improve the overall performance of your application.

1. **Avoiding Manual Optimization:**

Instead of manually implementing “shouldComponentUpdate” method in every component to optimize performance, you can use Pure Components which handle this optimization automatically.

However, it's important to note that using Pure Components comes with a few caveats:

* **Shallow Comparison Limitation:**

Pure Components only perform a shallow comparison of props and state. If your props or state contain complex data structures like nested objects or arrays, you might need to implement custom logic for deep comparison.

* **Potential Performance Impact:**

While Pure Components can improve performance in many cases, they might not always be the best choice, especially if your component relies on external factors or has side effects in its render method.

**Question – 4 :** What is the second argument that can optionally be passed tosetState and what is its purpose?

**Ans.**

The second argument that can optionally be passed to setState is a**callback function**which gets called immediately after the setState is completed and the components get re-rendered.

If you want your program to update the value of a state using setState and then perform certain actions on the updated value of the state then you must specify those actions in a function which should be the second argument of the setState. If we do not do so then those actions will be performed on the previous value of state because of**the asynchronous**nature of setState.