React JS

**Module : 11 React – Applying Redux**

1. **What is Redux ?**

* Redux is a State Management Library.

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.

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

* Redux does not have Dispatcher concept.
* Redux has an only Store whereas Flux has many stores.
* The Action objects will be received and handled directly by Store.

1. **What is Redux Thunk used for ?**

* Redux Thunk is a middleware for the Redux JavaScript Library. It allows you to return functions instead of actions, which allows for delayed actions. This is useful for handling actions that might not be synchronous.

**Here are some use cases for Redux Thunk :**

* Delayed actions : You can delay the dispatch of an action, or only dispatch if a certain condition is met.
* Working with promises : You can use Redux Thunk to make asynchronous API calls.
* Abstracting storage logic : Redux Thunk uses a pattern that abstracts storage logic from component to services, action builders and actions.

1. **What is Pure Component ? When to use Pure Component over Component ?**

* Pure Component is the type of component which re-renders only when the props passed to it changes and not even if its parent component re-renders of if the shouldComponentUpdate() method is called. It is greatly used to enhance the performance of a web application.

Pure Components are also called “stateless components” or “dumb components”. They optimize performance by reducing unnecessary re-renders.

**We can use Pure Components in the following scenarios :**

* When a component’s output depends only on its state and props.
* When there are no side effects, such as network requests or interactions with the DOM.
* When children components are classes and are being passed only some of the props of their parents.

1. **What is the second argument that can optionally be passed to setState and what is its purpose ?**

* The Second argument that can optionally be passed to setState is a callback function. This function is called after the state has been updated and the component has been re-rendered. This can be useful for performing any actions that need to be done after the state has been updated, such as updating the DOM or making an API call.

Generally we recommended using componentDidUpdate() for such logic instead.

**Example of how to use a callback function with setState :**

setState((prevState, props) => {

// Updated the State

return {

count: prevState.count + 1,

};

}, () => {

// Do something after the State has been updated.

Console.log(‘The state has been updated.’);

});