**React – Redux**

**UDAY RAJPUT**

**1) What is Redux?**

**Ans:-**

Redux is a state management library for JavaScript applications that centralizes the state in a single store, making it easier to manage and debug the state across the application.

**2) 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 useful for handling asynchronous operations, such as fetching data from an API, by enabling you to dispatch actions conditionally or after asynchronous operations are completed.

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

**Ans:-**

A Pure Component in React is a type of component that performs a shallow comparison of props and state to determine if the component should re-render. If the props and state haven't changed, the component will not re-render, improving performance.

When to Use Pure Component Over Component:

1. **Performance Optimization:** Use Pure Component when you have a component that renders the same output given the same props and state. This helps in avoiding unnecessary renders and boosts performance.

2. **Simple Comparison:** When the component's props and state are simple and don't require deep comparison.

3. **Stateless Functional Components:** Pure Components are more suitable for stateless functional components, though React.memo can also be used for function components to achieve similar optimizations.

**Example:**

jsx

import React, { PureComponent } from 'react';

class MyPureComponent extends PureComponent {

render() {

return <div>{this.props.value}</div>;

}

}

In contrast, a regular Component re-renders by default whenever its parent re-renders, without checking if the props or state have actually changed.

**Example:**

jsx

import React, { Component } from 'react';

class MyComponent extends Component {

shouldComponentUpdate(nextProps, nextState) {

// Custom logic to determine if a re-render is necessary

}

render() {

return <div>{this.props.value}</div>;

}

}

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

**Ans:-**

The second argument that can optionally be passed to setState is a callback function. This callback function is executed once the setState operation is complete and the component has re-rendered. It is useful for performing actions that depend on the updated state or DOM.