MODULE: 13 React – Applying Redux

. What is Redux?

Ans. Redux is a JS library for predictable and maintainable global state management.Applying Redux in a React application involves several steps. Below is a comprehensive guide to get you started with Redux in a React project.

.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.

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

Ans.The setState method accepts a second argument: the callback function executed after the state update has been applied and the component has re-rendered. In the above example, the setState callback function logs the message after the state has been updated with the data fetched from the server.

.What is Pure Component? When to use Pure

Component over Component?

Ans.Pure components, whether they're classes or functions, are special ones that aim to make your app run faster by avoiding unnecessary updates/re-renders. These components automatically figure out if they need to update/re-render or not by checking if anything important has changed in their data.

Step 1: Install Dependencies

```
npm install redux react-redux
```

Step 2: Create a Redux Store

Create a **store.js** file to configure and create the Redux store.

```
// store.js
import { createStore } from 'redux';
import rootReducer from './reducers'; // Assume you have a root reducer

const store = createStore(
  rootReducer,
  window.__REDUX_DEVTOOLS_EXTENSION__ && window.__REDUX_DEVTOOLS_EXTENSION__()
);

export default store;
```

Step 3: Create a Root Reducer

Create a **reducers** directory with an **index.js** file that combines your reducers.

```
// reducers/index.js
import { combineReducers } from 'redux';
import someReducer from './someReducer';

const rootReducer = combineReducers({
  someState: someReducer,
  // Add other reducers here
});

export default rootReducer;
```

Create a sample reducer:

```
// reducers/someReducer.js
const initialState = {
  // Initial state
};

const someReducer = (state = initialState, action) => {
  switch (action.type) {
    case 'SOME_ACTION':
      return {
        ...state,
        // Update state
      };
    default:
      return state;
```

Step 4: Provide the Redux Store to React

```
// index.js
import React from 'react';
import ReactDOM from 'react-dom';
import { Provider } from 'react-redux';
import App from './App';
import store from './store';

ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById('root')
);
```