Redux with Redux Thunk

Last updated by | Subramanya Dixit | May 5, 2025 at 8:13 PM GMT+5:30

📘 Why Use Redux with Thunk?

Redux is a predictable state container for JavaScript apps, allowing you to manage state globally. It helps in managing state across large applications, ensuring that state changes are predictable and centralized. Redux Thunk middleware allows you to write action creators that return functions (thunks) instead of action objects, enabling asynchronous operations like fetching data from APIs.

🧠 Key Concepts

✅ What is Redux?

Redux is a state management library that provides a centralized store for your app's state, ensuring all components have access to the same state. Redux works through the following core concepts:

Store: The global state of the application.
Actions: Payloads of information that send data to the store.
Reducers: Functions that specify how the application's state changes in response to an action.

```
// Action example
const increment = () => ({
  type: 'INCREMENT'
});

// Reducer example
const counterReducer = (state = 0, action) => {
  switch (action.type) {
    case 'INCREMENT':
      return state + 1;
    default:
      return state;
  }
};
```

✅ What is Redux Thunk?

Redux Thunk is a middleware for Redux that allows action creators to return a function (a thunk) instead of an action object. This function can then dispatch actions asynchronously or perform side effects, such as API calls.
Example action creator with Redux Thunk:

```
// Async action creator
const fetchData = () => {
  return async (dispatch) => {
    const response = await fetch('https://api.example.com/data');
    const data = await response.json();
    dispatch({ type: 'SET_DATA', payload: data });
  };
};
```

✅Setting up Redux and Redux Thunk

1. **Install Redux and Redux Thunk:**

npm install redux react-redux redux-thunk

2. **Create the Redux Store and Apply Thunk Middleware:**

```
import { createStore, applyMiddleware } from 'redux';
import thunk from 'redux-thunk';

const rootReducer = (state = {}, action) => {
  switch (action.type) {
    case 'SET_DATA':
      return { ...state, data: action.payload };
    default:
      return state;
  }
};

const store = createStore(rootReducer, applyMiddleware(thunk));
```

3. **Provide the Redux Store to Your Application:**

```
import { Provider } from 'react-redux';
import App from './App';
import { store } from './store';

function Root() {
  return (
    <Provider store={store}>
      <App />
    </Provider>
  );
}
```

💡 Best Practices

- **Keep Reducers Pure:** Reducers should be pure functions, meaning they should not modify the state directly or have side effects.
- **Separate Action Creators:** Keep your action creators separate from the component logic. This improves code reusability and testability.
- **Normalize Data:** When dealing with large datasets, normalize your data to prevent redundancy and make it easier to update individual items.

📊 Visual Overview

```
graph TD
  A[User Action] --> B[Action Creator (Thunk)]
  B --> C[Dispatch Action]
  C --> D[Redux Store]
  D --> E[React Components]
```

🛠️ Practice Exercises

1. Set up Redux in a React application and create a simple counter app with increment and decrement actions.

2. Use Redux Thunk to fetch data from an API and update the state with the fetched data.

3. Refactor the counter app to handle async actions (like incrementing after a delay).

4. Implement normalization in the Redux state for handling complex data like lists of users.

❓ Quiz

1. What is Redux Thunk used for?
a) To make actions synchronous
b) To allow asynchronous actions in Redux
c) To simplify Redux reducers
d) To avoid the need for actions
✅ **Answer:** b) To allow asynchronous actions in Redux

---

2. What type of argument does Redux Thunk action creators return?
a) Action objects
b) Functions (thunks)
c) Promises
d) Plain objects
✅ **Answer:** b) Functions (thunks)

---

3. How do you apply Redux Thunk middleware in the store?
a) By passing it as the second argument to `createStore`
b) By wrapping the store with `applyMiddleware`
c) By using `applyMiddleware(thunk)` with `createStore`
d) By passing it directly to the component
✅ **Answer:** c) By using `applyMiddleware(thunk)` with `createStore`

---

4. What should a reducer function always return?
a) A new object or array directly mutated from the state
b) A new state object, unchanged from the original
c) A promise
d) A side effect like an API call
✅ **Answer:** b) A new state object, unchanged from the original

---