گزارش درباره Redux Middleware، Redux Persist، Redux Logger، Redux Thunk و Recoil JS

Redux Middleware .

Middleware یا میان‌افزار در Redux به عنوان ابزاری برای گسترش قابلیت‌های این کتابخانه استفاده می‌شود. در حالت عادی، داده‌ها و state (حالت‌ها) در Redux به صورت همگام مدیریت می‌شوند. با این حال، اگر نیاز به اجرای عملیات‌هایی مانند درخواست‌های غیرهمگام (مثل درخواست‌های شبکه) داشته باشیم، استفاده از میان‌افزار ضروری است.

Middleware این امکان را فراهم می‌کند که قبل از رسیدن یک Action به Reducer، پردازشی انجام شود. این پردازش می‌تواند شامل درخواست‌های API، ثبت فعالیت‌ها، یا حتی تغییر Action قبل از رسیدن به Reducer باشد.

نمونه کد Middleware ساده:

```
const loggerMiddleware = store => next => action => {
    console.log('Dispatching:', action);
    const result = next(action);
    console.log('Next State:', store.getState());
    return result;
};

import { createStore, applyMiddleware } from 'redux';
import rootReducer from './reducers';

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

Redux Persist .

Redux Persist یک کتابخانه برای **نگهداری** (Persist) **داده‌ها** در Redux است. به طور معمول، زمانی که اپلیکیشن بسته یا رفرش می‌شود، اطلاعات موجود در state اصلی Redux از بین می‌رود. با استفاده از Redux Persist، می‌توان اطلاعات را در حافظه مرورگر (مثل LocalStorage یا SessionStorage) نگهداری کرد تا پس از باز کردن مجدد برنامه، این اطلاعات بازیابی شوند.

نصب و راه‌اندازی Redux Persist:

ابتدا پکیج را نصب کنید:

```
npm install redux-persist
```

نمونه کد استفاده از Redux Persist:

```
import { createStore } from 'redux';
import { persistStore, persistReducer } from 'redux-persist';
import storage from 'redux-persist/lib/storage';
import rootReducer from './reducers';

const persistConfig = {
    key: 'root',
    storage,
};

const persistedReducer = persistReducer(persistConfig,
rootReducer);
const store = createStore(persistedReducer);
const persistor = persistStore(store);

export { store, persistor };
```

1. *Redux Logger*

Redux Logger یک Middleware مفید برای توسعه‌دهندگان است که وضعیت **state** را در کنسول نمایش می‌دهد و تغییرات **state** را به‌طور دقیق نشان می‌دهد. این قابلیت به توسعه‌دهندگان کمک می‌کند تا مشکلات احتمالی را بهتر شناسایی کنند.

نصب Redux Logger:

برای استفاده از Redux Logger باید آن را نصب کنید:

```
npm install redux-logger
```

نمونه کد استفاده از Redux Logger:

```
import { createStore, applyMiddleware } from 'redux';
import { createLogger } from 'redux-logger';
import rootReducer from './reducers';

const logger = createLogger();
const store = createStore(rootReducer, applyMiddleware(logger));
```

2. *Redux Thunk*

Redux Thunk یکی دیگر از Middlewareهای Redux است که برای مدیریت درخواست‌های غیرهمگام (مانند درخواست‌های شبکه) استفاده می‌شود. به‌طور پیش‌فرض، Redux فقط **Action**هایی که به صورت شیء هستند را قبول می‌کند. اما با Redux Thunk، می‌توان تابعی را به عنوان Action ارسال کرد و درخواست‌های غیرهمگام را درون آن تابع اجرا کرد.

نصب Redux Thunk:

برای استفاده از Redux Thunk، پکیج را نصب کنید:

```
npm install redux-thunk
```

نمونه کد استفاده از Redux Thunk:

```
import { createStore, applyMiddleware } from 'redux';
import thunk from 'redux-thunk';
import rootReducer from './reducers';

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

const fetchData = () => {
    return async dispatch => {
        dispatch({ type: 'FETCH_DATA_REQUEST' });
        try {
            const response = await
fetch('https://jsonplaceholder.typicode.com/todos');
            const data = await response.json();
            dispatch({ type: 'FETCH_DATA_SUCCESS', payload: data
});
        } catch (error) {
            dispatch({ type: 'FETCH_DATA_FAILURE', error });
        }
    };
};

store.dispatch(fetchData());
```

Recoil JS .

Recoil یک کتابخانه جدید برای مدیریت state است که توسط تیم فیس‌بوک توسعه داده شده و با React یکپارچه شده است. Recoil از مفاهیم Atom و Selector برای مدیریت stateها استفاده می‌کند. یکی از مزایای اصلی Recoil این است که امکان استفاده آسان از stateهای اشتراکی بین کامپونت‌ها را بدون نیاز به ابزارهای پیچیده‌تر مانند Redux فراهم می‌کند.

نصب Recoil:

برای شروع کار با Recoil، ابتدا آن را نصب کنید:

```
npm install recoil
```

نمونه کد استفاده از Recoil:

```
import React from 'react';
import { RecoilRoot, atom, useRecoilState } from 'recoil';

const countState = atom({
    key: 'countState',
    default: 0,
});

function Counter() {
    const [count, setCount] = useRecoilState(countState);
    return (
        <div>
            <p>Count: {count}</p>
            <button onClick={() => setCount(count +
1)}>Increment</button>
            <button onClick={() => setCount(count -
1)}>Decrement</button>
        </div>
    );
}

function App() {
    return (
        <RecoilRoot>
            <Counter />
        </RecoilRoot>
    );
}

export default App;
```

نتیجه‌گیری

هر یک از این ابزارها ویژگی‌ها و قابلیت‌های مخصوص به خود را دارند. اگرچه Redux همچنان ابزار قوی و محبوبی برای مدیریت state است، اما با ظهور کتابخانه‌های جدید مانند Recoil، توسعه‌دهندگان انتخاب‌های بیشتری برای مدیریت state و بهبود عملکرد و تجربه کاربری برنامه‌ها دارند.