Урок 6.2. Redux Thunk

Материалы

Документация по Redux Thunk

Writing Logic with Thunks | Redux

The word "thunk" is a programming term that means "a piece of code that does some delayed work". Rather than execute some logic now, we can write a function body or code that can be used to perform the work later.

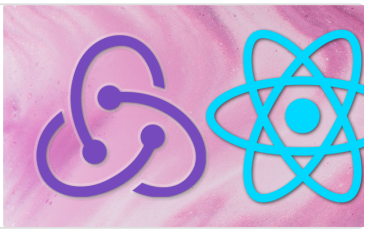
https://redux.js.org/usage/writing-logic-thunks

Сохранение состояния Redux в localStorage (redux-persist)

Persist state with Redux Persist using Redux Toolkit in React - LogRocket Blog

With the Redux Persist library, developers can save the Redux store in persistent storage, for example, the local storage. Therefore, even after refreshing the browser, the site state will still be preserved. Redux Persist also includes methods that allow us to

https://blog.logrocket.com/persist-state-redux-persist-redux-toolkit-react/



Summary

Redux Thunk необходим для выполнения асинхронных операций в Redux хранилище.

Например, если необходимо получить данные с сервера и записать в хранилище, можно использовать Thunk.

Дополнительно устанавливать ничего не нужно: весь необходимый функционал содержит в себе `@reduxjs/toolkit`.

Создадим отдельный slice (модуль) для хранения списка товаров: `store/productsSlice.js`.

```
// Импортируем дополнительно функцию createAsyncThunk из toolkit
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";

// Используем функцию createAsyncThunk
// Функция принимает два параметра: строку (название) и асинхронную функцию.
// Название должно состоять из двух частей, разделённых слэшем: название модуля и название самой функц
ии.
// В асинхронной функции можно получить данные и вернуть результат получения данных.
const getProducts = createAsyncThunk(
    'products/getProducts',
    async (thunkAPI) => {
        console.log('getProducts');
        const response = await fetch('https://fakestoreapi.com/products');
```

```
        const result = await response.json();
        console.log('result', result);

        return result;
    }
)


// В самом слайсе тоже есть изменения.
// Вместо объекта reducers будем использовать extraReducers.
export const productsSlice = createSlice({
    name: 'products',
    initialState: {
        // В entities будет храниться список товаров
        entities: [],
        // loading – индикатор/флаг загрузки
        loading: false
    },
    extraReducers: {
        // У созданного Thunk есть встроенные свойства: pending, fulfilled и rejected, каждое из котор
ых будет являться здесь функцией-редьюсером, изменяющим состояние
        [getProducts.pending]: (state) => {
            console.log('getProducts pending');
            // pending – состояние ожидания товаров, поэтому присваиваем loading значение true
            state.loading = true;
        },
        [getProducts.fulfilled]: (state, { payload }) => {
            console.log('getProducts done');
            // сбрасываем значени флага, поскольку в fulfilled завершилось получение данных, а также з
аписываем payload (результат) в entities
            state.loading = false;
            state.entities = payload;
        },
        [getProducts.rejected]: (state) => {
            console.log('getProducts rejected');
            // Если запрос не удался, просто сбрасываем флаг (запрос завершился, но завершился неудачн
о, поэтому данных нет, но загрузка уже завершена
            state.loading = false;
        }
    }
});

export default productsSlice.reducer;

// Экспортируем созданный thunk
export { getProducts };
```

Чтобы данные загрузились, необходимо вызвать созданный thunk. Делать это будем с помощью функции `dispatch`, так же, как мы это делали с обычным изменением состояния.

```
// Импортируем созданный Thunk
import { getProducts } from '../../store/productsSlice';

function IndexPage() {
    // Используем состояние из Redux
    const [ products, isLoading ] = useSelector((state) => [ state.products.entities, state.products.l
oading ]);
    // Создаём функцию dispatch
```

```
    const dispatch = useDispatch();

    useEffect(() => {
        // Используем dispatch
        dispatch(getProducts());
    }, []);

...
}

export default IndexPage;
```