Redux

Состояние в redux

Redux - отдельная библиотека. Может использоваться и без react. react-redux - связывает react с redux.

Установим redux:
npm i redux react-redux

Состояние опишем в **index.js**. Это может быть объект или массив.

```
// 1. Дефолтное состояние
const defaultState = {
  counter: 0,
};
```

Определим функцию reducer, которая будет менять объект с состоянием с двумя аргументами - состояние и действие.

```
// 2. Функция reducer
// state - состояние
// action - { type: "", payLoad: "?"}
const reducer = (state = defaultState, action) => {
  switch (action.type) {
    case "INCREASE":
      return { ...state, counter: state.counter + action.payLoad };  // возвращаем состояние. В данном случае ...state можно опустить
    case "DECREASE":
      return { ...state, counter: state.counter - action.payLoad };

    default:
      return state;  // возвращаем начальное состояние
  }
};
```

Для работы с reducer создадим store, предварительно ее импортировать:

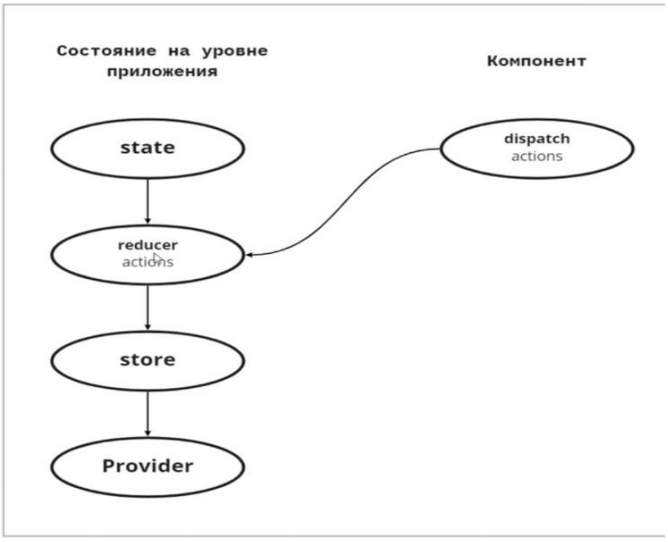
```
import { createStore } from "redux";

// 3. Создаем store
const store = createStore(reducer);
```

Чтобы передать store в приложение, нужно приложение (App) **обернуть в провайдер,** предварительно его импортировать:

```
import { Provider } from "react-redux";

root.render(
  <Provider store={store}>
    <App />
  </Provider>
);
```



Перепишем код в **App.js**:

```
import { useDispatch, useSelector } from "react-redux";

function App() {
 // состояние state.counter
 const counter = useSelector((state) => state.counter);  // достаем состояние - функция с аргументом состоянием state

 // dispatch для изменения состояния
 const dispatch = useDispatch();  // достаем функцию dispatch


 const increase = (value) => {
  dispatch({ type: "INCREASE", payLoad: value });  // передаем объект action
 };

 const decrease = (value) => {
  dispatch({ type: "DECREASE", payLoad: value });  // передаем объект action
 };

 return (
  <div className="App">
   <h1>{counter}</h1>
   <button onClick={() => decrease(Number(prompt()))}>Уменьшить</button>
   <button onClick={() => increase(Number(prompt()))}>Увеличить</button>
  </div>
 );
```

Объединение reducer-ов

Допустим, нам нужно несколько состояний - reducer-ов - список пользователей и счетчиков. Создадим папку **store**. Внутри нее файл **src/storecounterReducer.js**, где будет стартовое значение счетчика и reduce:

```
const defaultState = {
 counter: 10,
};

export const reducer = (state = defaultState, action) => {
 switch (action.type) {
  case "INCREASE":
   return { ...state, counter: state.counter + action.payLoad };
  case "DECREASE":
   return { ...state, counter: state.counter - action.payLoad };

  default:
   return state;
 }
};
```

Аналогично для пользователей в **src/store/usersReducers.js**:

```
const defaultState = {
 users: [],
};

export const userReducer = (state = defaultState, action) => {
 switch (action.type) {
  default:
   return state;  // пока нет никаких actions
 }
};
```

Объединим в общий reducer и общее store. Создадим файл **src/store/index.js**:

```
import { combineReducers, createStore } from "redux";
import { counterReducer } from "./counterReducer";
import { usersReducer } from "./usersReducer";
```

```
const rootReducer = combineReducers({
 counter: counterReducer,
 users: usersReducer
}); // объединим reducer-ы

export const store = createStore(rootReducer);  // создадим store
```

Перепишем **src/index.js:**

```
import React from "react";
import ReactDOM from "react-dom/client";
import { createStore } from "redux";
import { Provider } from "react-redux";
import App from "./App";
import { store } from "./store";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
 <Provider store={store}>
  <App />
 </Provider>
);
```

В **App.js** поправим строку:

```
 const counter = useSelector((state) => state.counter.counter);
```

Redux devtools

Установим redux-devtools/extension:
npm i @redux-devtools/extension

Далее, при создании store передадим второй аргумент в файле **store/index.js:**

```
import { composeWithDevTools } from "@redux-devtools/extension";
...
export const store = createStore(rootReducer, composeWithDevTools());
```

В браузере **Chrome** установим расширение Redux DevTools
После этого в инструментах разработчика появится пункт Redux

Отображение, удаление, добавление пользователей

В файле **store/usersReducers.js** добавим пользователей:

```
const defaultState = {
 users: [
  { name: "John", id: 1 },
  { name: "Bob", id: 2 },
  { name: "Peter", id: 3 },
 ],
};
```

В файле **App.js** добавим:

```
function App() {
...
 /* Users */
 const deleteUser = (id) => {
  dispatch({ type: "DELETE_USER", payload: id });
 };

 const addUser = (name) => {
  const user = {
   name: name,
   id: Date.now(),
  };
  dispatch({ type: "ADD_USER", payload: user });
 };
```

```
 const showUsers = () => {
  return users.map((user) => (
   <div onClick={() => { deleteUser(user.id); }}  className="user"  key={user.id}>{user.name} </div>
  ));
 };

 return (
  <div className="App">
   <h1>{counter}</h1>
   <button onClick={() => decrease(Number(prompt()))}>Уменьшить</button>
   <button onClick={() => increase(Number(prompt()))}>Увеличить</button>
   <hr />
   <button onClick={() => addUser(prompt())}>Добавить пользователя</button>
   {users.length > 0 ? showUsers() : <h3>Нет пользователей</h3>}
  </div>
 );
}
```

Файл **store/usersReducer.js**:

```
const defaultState = {
 users: [
  { name: "John", id: 1 },
  { name: "Bob", id: 2 },
  { name: "Peter", id: 3 },
 ],
};

export const usersReducer = (state = defaultState, action) => {
 switch (action.type) {
  case "ADD_USER":
   return { ...state, users: [...state.users, action.payload] };
  case "DELETE_USER":
   return { ...state, users: [...state.users.filter((user) => user.id !== action.payload)] };
  default:
   return state;
 }
};
```

Рефакторинг

store/counterReducer.js:

```
const INCREASE = "INCREASE";
const DECREASE = "DECREASE";
...
export const increaseCounterAction = (payload) => {
 return {
  type: INCREASE,
  payload,
 };
};

export const decreaseCounterAction = (payload) => {
 return {
  type: DECREASE,
  payload,
 };
};
```

App.js:

```
import { increaseCounterAction, decreaseCounterAction } from "./store/counterReducer";

 const increase = (value) => { dispatch(increaseCounterAction(value)); };
 const decrease = (value) => {  dispatch(decreaseCounterAction(value)); };
```

Аналогично с **store/usersReducer.js**

Асинхронность. Redux-thunk, applyMiddleware

Будем запрашивать данные о пользователях по кнопке с запросом к серверу jsonplaceholder.typicode.com

Установим redux-thunk:
npm i redux-thunk

App.js:

```
import { addUserAction, deleteUserAction, addManyUsersAction } from "./store/usersReducer";
...
 const fetchusers = () => {    //добавили функцию для обработки сетевого запроса
   return function (dispatch) {
     fetch("https://jsonplaceholder.typicode.com/users")
       .then((response) => response.json())
       .then((data) => dispatch(addManyUsersAction(data)));
   };
 };

 return (
   <div className="App">
     <h1>{counter}</h1>
     <button onClick={() => decrease(Number(prompt()))}>Уменьшить</button>
     <button onClick={() => increase(Number(prompt()))}>Увеличить</button>
     <hr />
     <button onClick={() => addUser(prompt())}>Добавить пользователя</button>
     <button onClick={() => dispatch(fetchusers())}>Добавить пользователей из базы</button> //добавили кнопку и обработчик
     {users.length > 0 ? showUsers() : <h3>Нет пользователей</h3>}
   </div>
 );
```

store/usersReducer.js:

```
...
const ADD_MANY_USERS = "ADD_MANY_USERS";

export const usersReducer = (state = defaultState, action) => {
  switch (action.type) {
    case ADD_USER:
      return { ...state, users: [...state.users, action.payload] };
    case DELETE_USER:
      return { ...state, users: [...state.users.filter((user) => user.id !== action.payload)] };
    case ADD_MANY_USERS:  //добавляем действие - добавление пользователей
      return { ...state, users: [...state.users, ...action.payload] };
    default:
      return state;
  }
};
...
export const addManyUsersAction = (payload) => {
  return { type: ADD_MANY_USERS,  payload,  };
};
```

store/index.js:

```
import { applyMiddleware, combineReducers, createStore } from "redux";
import thunk from "redux-thunk";
...

const rootReducer = combineReducers({
  counter: counterReducer,
  users: usersReducer,
});

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

В последней строке можно было сделать и так, но тогда не будет devTools:

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

Redux toolkit

Перепишем формирование **store** через **toolkit**

Установим @reduxjs/toolkit:
npm i @reduxjs/toolkit

Создадим папку **src/toolkitRedux**. В ней создадим **src/toolkitRedux/counterReducer.js:**

```
import { createReducer, createAction } from "@reduxjs/toolkit";

const defaultState = {
  counter: 100,
};

// создаем функции action
export const increaseAction = createAction("INCREASE");
export const decreaseAction = createAction("DECREASE");

// описываем reducer
export default createReducer(defaultState, {  // объект
  [increaseAction]: function (state, action) {
    state.counter = state.counter + action.payload;  // деструктуризация уже не нужна
  },
  [decreaseAction]: function (state, action) {
    state.counter = state.counter - action.payload; // деструктуризация уже не нужна
  },
});
```

Все аналогично для **src/toolkitRedux/usersReducer.js**

Создадим общее store в **src/toolkitRedux/index.js:**

```
import { combineReducers, configureStore } from "@reduxjs/toolkit";
import counterReducer from "./counterReducer";
import usersReducer from "./usersReducer";

const rootReducer = combineReducers({  // объединяем reducer-ы
  counter: counterReducer,
  users: usersReducer,
});

export const store = configureStore({  // конфигурируем store
  reducer: rootReducer,
});
```

В **src/index.js** переподключим store:

```
import { store } from "./toolkitRedux";
```

В **src/App.js** подправим импорт и названия функций:

```
import { addUserAction, deleteUserAction, addManyUsersAction } from "./toolkitRedux/usersReducer";
import { increaseCounterAction, decreaseCounterAction } from "./toolkitRedux/counterReducer";
```

createSlice

Рассмотрим еще один способ описания reducer-ов в redux с помощью Slice. Заново создадим store и reducer-ы и сделаем это в отдельной директории **src/toolkitSliceRedux**. С помощью createSlice создадим и состояние и reducer

В **src/toolkitSliceRedux/counterSliceReducer.js:**

```
import { createSlice } from "@reduxjs/toolkit";

const counterSlice = createSlice({
  name: "counter",  // имя
  initialState: {    // начальное состояние
    counter: 200,
  },
  reducers: {  // reducer-ы
```

```
    increase(state, action) {
      state.counter = state.counter + action.payload;
    },
    decrease(state, action) {
      state.counter = state.counter - action.payload;
    },
  },
});

// делаем экспорты
export default counterSlice.reducer;  // ! reducer, а не reducers
export const { increase, decrease } = counterSlice.actions;
```

Аналогично **src/toolkitSliceRedux/usersSliceReducer.js:**

```
import { createSlice } from "@reduxjs/toolkit";

const counterSlice = createSlice({
  name: "users",
  initialState: {
    users: [
      { name: "John", id: 100 },
      { name: "Bob", id: 101 },
      { name: "Peter", id: 102 },
    ],
  },
  reducers: {
    addUser(state, action) {
      state.users.push(action.payload);
    },
    deleteUser(state, action) {
      const index = state.users.findIndex((user) => user.id === action.payload);
      state.users.splice(index, 1);
    },
    addManyUsers(state, action) {
      state.users = [...state.users, ...action.payload];
    },
  },
});

export default counterSlice.reducer;
export const { addUser, deleteUser, addManyUsers } = counterSlice.actions;
```

Создадим общее store в **src/toolkitSliceRedux/index.js** аналогично, как в toolkit:

```
import { combineReducers, configureStore } from "@reduxjs/toolkit";
import counterSliceReducer from "./counterSliceReducer";
import usersSliceReducer from "./usersSliceReducer";

// объединяем reducer-ы
const rootReducer = combineReducers({
  counter: counterSliceReducer,
  users: usersSliceReducer,
});

// конфигурируем store
export const store = configureStore({
  reducer: rootReducer,
});
```

В **src/index.js** переподключим store:

```
import { store } from "./toolkitSliceRedux";
```

В **src/App.js** подправим импорт и названия функций:

```
import {  addUser as addUserAction, deleteUser as deleteUserAction, addManyUsers as addManyUsersAction } from
./toolkitSliceRedux/usersSliceReducer";
import {  increase as increaseAction,  decrease as decreaseAction} from "./toolkitSliceRedux/counterSliceReducer";
```