Урок 6.1. React Redux

Для того, чтобы хранить общее состояние приложения, которое доступно из любого компонента, можно использовать React Redux.

Состояние Redux глобальное и хранится на протяжении работы приложения (до обновления страницы). То есть при переходе между страницами без перезагрузки состояние сохраняется.

Примером данных, которые обычно хранятся в общем хранилище, могут служить данные, необходимые в нескольких местах приложения (компонентах и страницах), например, товары в корзине, отображать которые необходимо и на странице товаров, и на странице корзины, и в шапке.

react-redux – библиотека, которую необходимо установить с помощью npm. Для работы нам также потребуется библиотека redux toolkit, которая содержит функции, упрощающие работу с хранилищем.

Устанавливаем необходимые библиотеки:

`npm install @reduxjs/toolkit react-redux`

Создание хранилища

Хранилище будет состоять из нескольких элементов: сам файл хранилища (мы назовём его `store.js`, “слайсы” – модули, в каждом из которых будут храниться необходимые данные. Разделять на слайсы можно по смыслу: разные данные лучше хранить отдельно в разных модулях.

Управлять состоянием приложения будут редьюсеры (reducers) – специальные функции, изменяющие состояние.

В `src` необходимо создать папку `store` и в ней новый файл `store.js`.

```
import { configureStore } from '@reduxjs/toolkit'

export default configureStore({
  reducer: {}, // пока оставим этот объект пустым
})
```

Подключение хранилища

Подключение хранилища происходит в главном файле: `index.js` – там же, где мы подключали роутер.

Для подключения необходимо импортировать компонент `Provider` из `react-redux` и использовать его. У данного компонента есть пропс `store`, в который необходимо положить объект хранилища, который мы сделали на предыдущем шаге.

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import './index.css'
import App from './App'
import store from './store/store' // импорт состояния
import { Provider } from 'react-redux' // импорт компонента для подключения хранилища

// As of React 18
const root = ReactDOM.createRoot(document.getElementById('root'))

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

Создание модуля (слайса)

Далее создадим так называемый слайс – модуль, который будет содержать какие-то данные для использования по всему приложению.

Создадим файл `counterSlice.js` в `src/store`.

```
import { createSlice } from '@reduxjs/toolkit'

export const counterSlice = createSlice({
// имя модуля
  name: 'counter',
// изначальное состояние (в данном случае – объект с полем value)
  initialState: {
    value: 0,
  },
// редьюсеры – функции, которые будут менять состояние
  reducers: {
    increment: (state) => {
      state.value += 1
```

```
    },
    decrement: (state) => {
      state.value -= 1
    },
    incrementByAmount: (state, action) => {
      state.value += action.payload
    },
  },
})

// Экспортируем три экшна (из образовавшегося объекта counterSlice).
// Экшны имеют такое же название, как и редьюсеры, что немного путает,
// однако нам нужно будет использовать именно их для изменения состояния
export const { increment, decrement, incrementByAmount } = counterSlice.actions

// Экспортируем главный редьюсер модуля
export default counterSlice.reducer
```

Теперь необходимо добавить модуль в главное хранилище, для этого нужно его импортировать и добавить в объект reducer в `store/store.js`:

```
import { configureStore } from '@reduxjs/toolkit'
import counterReducer from './counterSlice'

export default configureStore({
  reducer: {
    counter: counterReducer,
  },
})
```

Использование состояния

Для получения и изменения состояния необходимо использовать хуки, входящие в `react-redux`.

Для получения будем использовать хук `useSelector`:

```
import React from 'react'
import { useSelector, useDispatch } from 'react-redux'
import { decrement, increment } from './counterSlice'

export function Counter() {
// Получаем текущее значение counter (обратите внимание на state.counter.value:
// state – это общее состояние приложения, counter – модуль,
// который мы создали и value – его значение внутри
```

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

// Хук useDispatch добавляет к нашему функционалу функцию, которая позволит
// нам изменять состояние
  const dispatch = useDispatch()

  return (
    <div>
      <div>
        <button
          aria-label="Increment value"
{/* На клик используем функцию dispatch, а внутри передаём экшн increment, который мы так
 же экспортировали ранее из файла модуля хранилища */}
          onClick={() => dispatch(increment())}
        >
          Increment
        </button>
        <span>{count}</span>
        <button
          aria-label="Decrement value"
          onClick={() => dispatch(decrement())}
        >
          Decrement
        </button>
      </div>
    </div>
  )
}
```