УДК 004.9

Студ. Е. Р. Смирнов
Науч. рук. А. Н. Щербакова
(кафедра информатики и веб-дизайна, БГТУ)

РАЗРАБОТКА HAREACT/REDUX С ИСПОЛЬЗОВАНИЕМ RAMDA

React – это инструмент для создания пользовательских интерфейсов. Его главная задача – обеспечение вывода на экран того, что можно видеть на веб-страницах. React значительно облегчает создание интерфейсов благодаря разбиению каждой страницы на небольшие фрагменты, называемые компонентами.

Компонент React – это участок кода, который представляет часть веб-страницы. Каждый компонент – это JavaScript-функция, которая возвращает кусок кода, представляющего фрагмент страницы [1].

Компоненты делятся на:

– умные;

– глупые.

Глупые служат для отображения статического контента страницы (рисунок 1).Умные служат для хранения и работы с состояниями (рисунок 2).Redux – это инструмент управления как состоянием данных, таки состоянием интерфейса в JavaScript-приложениях. Он подходит для одностраничных приложений, в которых управление состоянием может со временем становиться сложным. Redux не связан с каким-то определенным фреймворком, и, хотя разрабатывался для React, также может использоваться с Angular.

```
const Layout = ({ children }) => {
    return (
        <div className='view-container'>
            <Header />
            <div className='container'>
                <div className='row'>
                    <div className='col-lg-3 col-xl-3 col-sm-12 col-md-3'>
                        <Sidebar />
                    </div>
                    <div className='col-lg-9 col-xl-9 col-md-9 col-sm-12'>
                        {children}
                    </div>
                </div>
            </div>
            <Footer />
        </div>
    )
}
```

Рисунок 1 – Глупый компонент

```
class Search extends Component {
  constructor(props) {
    super(props)
    this.state = {
      value: ''
    }

    this.handleChange = this.handleChange.bind(this)
    this.handleSubmit = this.handleSubmit.bind(this)
  }

  handleChange(event) {
    this.setState({
      value: event.target.value
    })
  }

  handleSubmit(event) {
    event.preventDefault();
    this.props.searchWatch(this.state.value)
  }

  render() {
    return (
      <div className='search'>
        <form onSubmit={this.handleSubmit}>
          <input
            type="text"
            onChange={this.handleChange}
            className='searchInput'
            placeholder='Поиск товара'
          />
          <button className='searchButton'><img src="/images/icons8-search-filled-50.png" alt="Search" /></button>
        </form>
      </div>
    )
  }
```

Рисунок 2 – Умный компонент

Redux предлагает хранить все состояние приложения в одном месте, называемом «store» («хранилище»). Компоненты «отправляют» изменение состояния в хранилище, а не напрямую другим компонентам. Компоненты, которые должны быть в курсе этих изменений, «подписываются» на хранилище.

С Redux все компоненты получают свое состояние из хранилища. Также ясно, куда компонент должен отправить информацию об изменении состояния – опять же в хранилище. Компонент только инициирует изменение и не заботится об остальных компонентах, которые должны получить это изменение. Таким образом, Redux делает поток данных более понятным.

Ключевые моменты Redux:

– хранилище (store): хранит состояние приложения.

– действия (actions): некоторый набор информации, который исходит от приложения к хранилищу и который указывает, что именно нужно сделать. Для передачи этой информации у хранилища вызывается метод dispatch().

Reducer: функция (или несколько функций), которая получает действие и в соответствии с этим действием изменяет состояние хранилища [2].

Общая схема взаимодействия элементов архитектуры Redux представлена на рисунке 3.

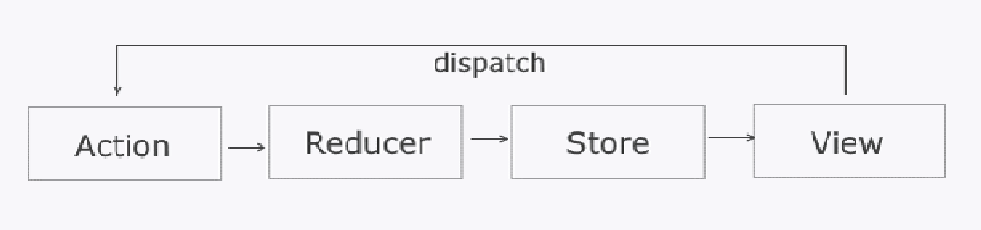

Рисунок 3–Схема взаимодействия элементов архитектуры Redux

Из View (то есть из компонентов React) мы посылаем действие, это действие получает функция reducer, которая в соответствии с действием обновляет состояние хранилища. Затем компоненты React применяют обновленное состояние из хранилища.

Ramda предлагает другой стиль написания кода, стиль, который был позаимствован из чисто функциональных языков программирования. Здесь механизм создание сложной функциональной логики реализуется с помощью композиции. Однако, любая библиотека предоставляет возможность реализовать функциональную композицию. Но в отличие от других, тут это «делается с легкостью» [3].

Пример использования данных технологий на разработанном проекте интернет-магазина часов.

Структура подключаемых пакетов представлена на рисунке 4.

```
import React from 'react'
import ReactDOM from 'react-dom'
import { createStore, applyMiddleware } from 'redux'
import { composeWithDevTools } from 'redux-devtools-extension'
import thunk from 'redux-thunk'
import { syncHistoryWithStore } from 'react-router-redux'
import { Router, Route, browserHistory } from 'react-router'
import { Provider } from 'react-redux'
```

Рисунок 4–Структура подключаемых пакетов

Главная страница сайта представлена на рисунке 5.

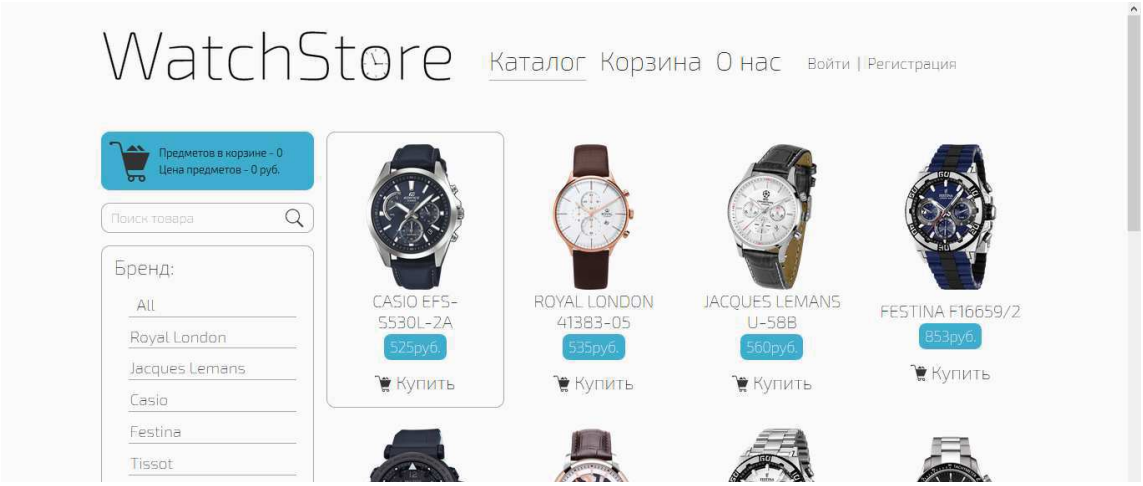

Рисунок 5–Главная страница веб-сайта

Фрагмент реализации поиска на веб-сайте представлен на рисунке 6.

```
const applySearch = item => R.contains(
  state.watchesPage.search.toUpperCase(),
  R.prop('name', item)
)
```

Рисунок 6–Фрагмент реализации поиска на веб-сайте

R.containsполучает значение bool, если есть вхождение подстроки в строку, R.propполучает значение по ключу.

Таким образом, можно сделать вывод, что, используя react/redux, можно очень просто обрабатывать данные внутри компонента независимо от других компонентов и передавать данные между ними. Также использование ramda за счет набора своих функций весьма упрощает работу с обработкой данных.

ЛИТЕРАТУРА

1. Основы React– [Электронный ресурс]. – 2006-2019. – Режим доступа: https://habr.com/ru/company/ruvds/blog/343022/. – Дата доступа: 20.04.2018.

2. Redux– [Электронный ресурс]. – 2014-2018. – Режим доступа: https://getinstance.info/articles/react/learning-react-redux/. – Дата доступа: 20.04.2018.

3. Почему Ramda? – [Электронный ресурс]. – 2014-2018. – Режим доступа: https://getinstance.info/articles/react/learning-react-redux/. – Дата доступа: 20.04.2018.