УДК 004

Использование библиотек React и Redux языка JavaScript для создания интерфейса на языке JSX

Халиманенков Андрей Сергеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статьей рассматривается создание интерфейса веб-приложения с возможностью удалять и добавлять позиции, а также менять их названия и цену. Для этой цели используются библиотеки React и Redux и язык JSX, который затем компилируется в JavaScript код для работы в браузере. Итогом исследования является часть веб-приложения, отвечающая за клиентскую сторону пользовательского интерфейса, разработанная с использованием вышеперечисленных технологий.
Ключевые слова: создание веб-сайтов, Redux, React, JavaScript, JSX, интерфейс, фронтэнд.

Using the React and Redux libraries of the JavaScript language to create an interface with help of JSX

Khalimanenkov Andrey Sergeevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses the creation of a web application interface with the ability to delete and add items, as well as change their names and price. For this purpose, the React and Redux libraries and the JSX language are used, which is then compiled into JavaScript code for working in the browser. The result of the study is the part of the web application responsible for the client side of the user interface, developed using the above technologies.
Keywords: websites creating, Redux, React, JavaScript, JSX, interface, frontend.

React — это удобный инструмент, который разработан для создания пользовательских интерфейсов. Главной особенностью является собственный язык JSX, который является смесью из JavaScript и HTML синтаксиса. JSX файлы компилируются в JavaScript с помощью библиотеки Babel. Разработчики могут добиваться высокой производительности приложений с помощью Virtual DOM. React позволяет разрабатывать автономные приложения, которые хранят данные на стороне клиента и помогают избавиться от длительных загрузок. Созданные компоненты разрабатываются так, чтобы была возможность их использовать в другом

проекте без изменений в коде. Высокий процент переиспользования кода повышает покрываемость тестами, что, в свою очередь, приводит к более высокому уровню контроля качества.

Цель исследования – описать возможности React и языка JSX на примере создания интерфейса.

Вопрос разработки интерфейсов при создании сайтов на React, и не только, волнует некоторых исследователей и специалистов: А. Ю. Винокуров и А. А. Леванов [1] посвятили статью разработке универсальной библиотеки многократно используемых компонентов для построения интерактивных мультимедийных электронных курсов на платформе HTML5+Javascript с использованием архитектуры SPA. О. В. Меркулова [2] привела сравнительный анализ трех самых популярных по состоянию на 2019 год JavaScript- фреймворков, выделенных путем анализа запросов Google и количества размещенных вакансий на различных ресурсах. А. С. Крюков [3] выявил рациональность использования библиотеки React.js в современной разработке. Е. В. Пантелеева [4] отразила сущность разработки сайта с использованием языка разметки HTML, особенности таблицы стилей CSS и языка программирования JavaScript. Н. Д. Лушников и А. Д. Альтерман [5] рассмотрели основы (технические возможности) каскадных таблиц стилей CSS. Кроме того, освещены главные преимущества и принцип работы каскадных таблиц. Л. А. Надеинский [6] разработал архетип модуля каскадных таблиц стилей с LESSCSS компилятором для быстрого создания модулей для разработки каскадных таблиц стилей с возможностью подключения к WAR модулям Java. Н. О. Айдарбаев [7] раскрыл понятие адаптивного дизайна как одного из процессов веб разработки. Дал определения разновидностей фронтэнд фреймворков, используемых в веб разработке, и подробный анализ компонентов фреймворка Bootstrap. Т. В. Макарова [8] в своём учебном пособии рассмотрела теоретические вопросы: гипертекстовый способ представления информации в компьютерной сети; основные принципы визуального дизайна веб-страниц; разработка концепции, структуры, макета, сайта; анализ юзабилити.

Большое количество важных на рынке IT компаний используют React [9]. Язык JSX, используемый в React, позволяет включать в себя HTML код, что сильно упрощает написание и понимание кода по сравнению с чистым JavaScript. Поэтому его использование является весомым аргументом в пользу React. Пример на рисунке 1. Слева чистый JS, а справа JSX.

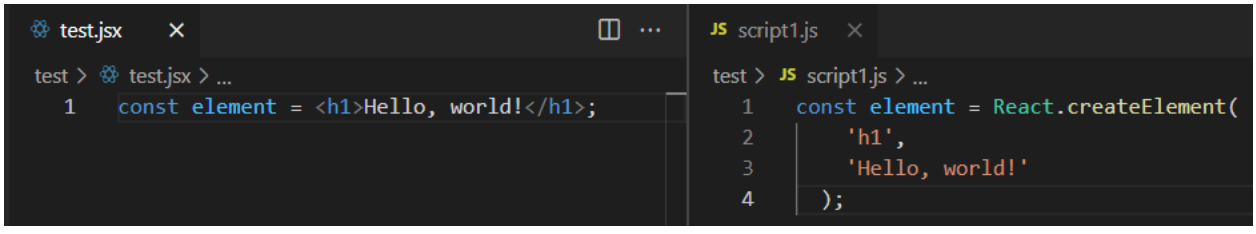

Рисунок 1 – Демонстрация разницы между синтаксисом JS и JSX

Для демонстрации возможностей и синтаксиса нужно всего два файла. Код будет работать без сборщиков задач, таких как Gulp или WebPack, т.к. они используются для действительно больших проектов. В данном случае их использование будем излишним. Для работы всех компонентов, таких как React, языка JSX и библиотеки Redux, их нужно подключить в главном HTML файле. Подключение этих модулей показано на рисунке 2.

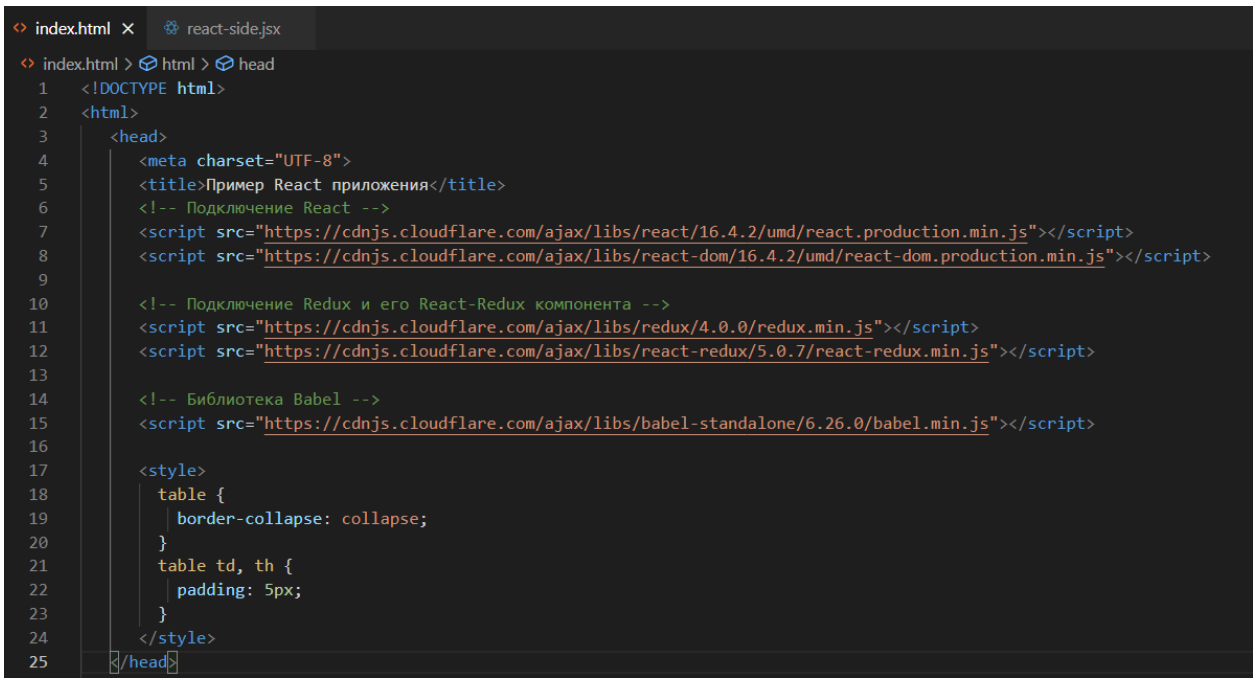

Рисунок 2 – Подключение библиотек

Также следует подключить JSX-файл в теге body главного html документа, как показанно на рисунке 3.

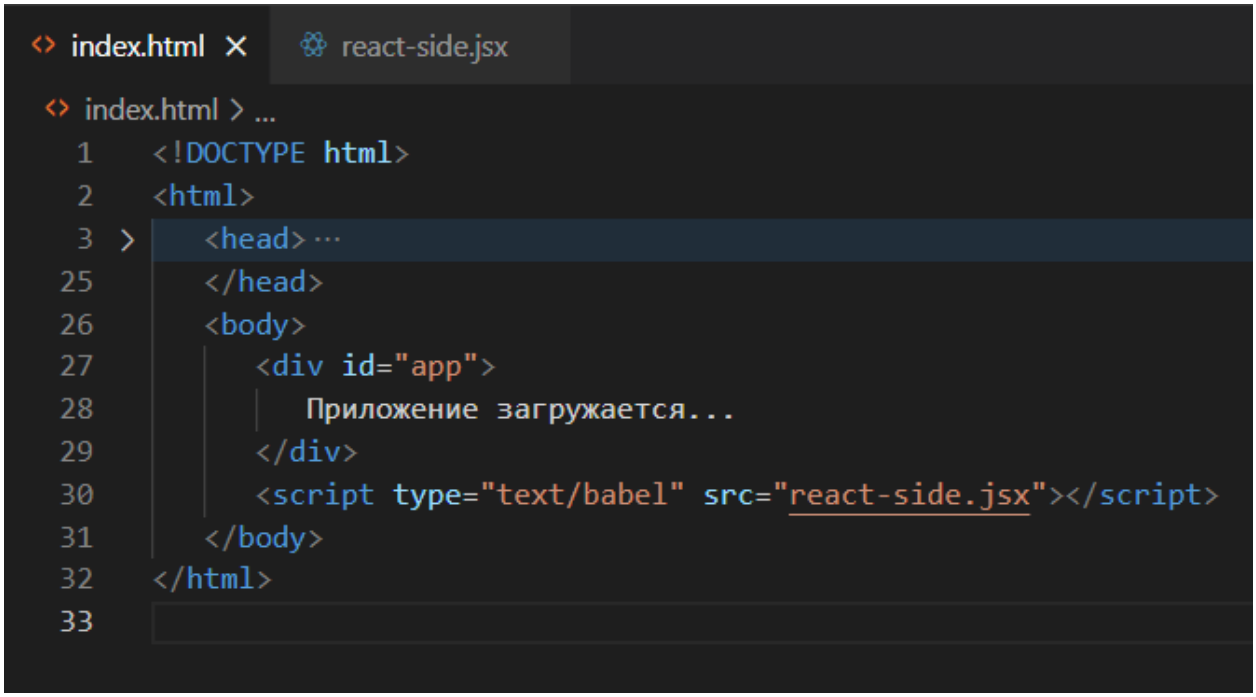

Рисунок 3 – Подключение JSX файла

Сам файл react-side.jsx содержит следующий код:

```
class ProductsPane extends React.Component {
   constructor (props, context) {
    super(props, context);
   }
   //эти методы подготавливают и отправляют события в Redux
   //Метод добавляет новый элемент в таблицу
   itemAdd = () => {
    var action = {
       type: 'ITEM_ADD'
    };
    this.props.dispatch(action);
   }
   //Метод удаляет элемент из таблицы
   itemDelete  = (itemId) =>  {
    var action = {
     type: 'ITEM_DELETE',
     itemId: itemId
    };
    this.props.dispatch(action);
   }
   //Метод редактирует элемент в таблице
   itemEdit = (itemId, event) =>   {
    var newName =  event.target.value;
    var newPrice =  event.target.value;
    var action = {
     type: 'ITEM_EDIT',
     data: {
      itemId: itemId,
      itemName: newName,
      itemPrice: newPrice
     }
    };
    this.props.dispatch(action);
   }
   //рендерит список элементов в виде html тегов
   render () {
    const items = this.props.items;

    var trList = items.map( (item, index) => {
     return (<tr key={item.itemId}>
      <td>{item.itemId}</td>
      <td><input type="text" onChange={this.itemEdit.bind(null, item.itemId)}
value={item.itemName} /></td>
      <td><input type="number" onChange={this.itemEdit.bind(null, item.itemName)}
value={item.itemPrice} /></td>
      <td>
       <button onClick={this.itemDelete.bind(null, item.itemId)}>
```

```
          Удалить
        </button>
      </td>
    </tr>);
  });

  return (<div>
    <table border="1">
      <thead>
        <th>Номер</th>
        <th>Наименование</th>
        <th>Цена</th>
        <th>Удалить</th>
      </thead>
      <tbody>
        {trList}
      </tbody>
    </table>
    <br/>
    <button onClick={this.itemAdd}>
      Добавить
    </button>
  </div>);
 }
}

var nextItemId = 1;

//интегрирует компоненты Redux в React
const stateToStorage = (state) => {
 return {
  items: state.itemList
 }
}

const getIndexByItemId = (items, itemId)  => {
  for(var i = 0; i < items.length; i++) {
    var item = items[i];
    if(item.itemId === itemId) {
      return i;
    }
  }
  return -1;
};

//Следующие методы берут текущие состояния Redux и создают новые на основе
действий пользователя. Это необходимо, т.к. состояния Redux не поддаются изменению
const appReducer = (state = {itemList: []}, action) => {
```

```
  // Копируем старый массив данных
  let items = state.itemList.slice();
  console.log('Actions', action);

  switch (action.type) {
   case 'ITEM_ADD':
    nextItemId++;
    var item = {itemId : nextItemId, itemName: "" };
    items.push(item);
    break;
   case 'ITEM_DELETE':
    var idx = getIndexByItemId(items, action.itemId);
    if(idx != -1)  {
     items.splice(idx, 1); // Removes element at `idx`
    }
    break;
   case 'ITEM_EDIT':
    var idx = getIndexByItemId(items, action.data.itemId);
    if(idx != -1)  {
       items[idx].itemName = action.data.itemName;
    }
    break;
  }
  //создаём новое изменённое состояние
  const newState = {
   itemList: items
  }
  console.log('Current State', newState);
  return newState;
 }

 //createStore создаёт страницу со складом магазина, appReducer обновляет состояния,
itemList[] это список товаров
 let store = Redux.createStore(appReducer, {
  itemList: [
   { itemId : 1 , itemName :'Ручка', itemPrice : '50' }
  ]
 });

 // Присоединяем Redux библиотеку для React к состоянию товаров в Redux
 const MyApp = ReactRedux.connect (
  stateToStorage
 )(ProductsPane);

 // Рендер DOM элементов (из чего состоит разметка страницы)
 ReactDOM.render(
  <ReactRedux.Provider store={store}>
   <MyApp />
  </ReactRedux.Provider>,
```

```
document.getElementById('app')
    );
```

В итоге получаем интерфейс с возможностью создания и удаления полей, который показан на рисунке 4.

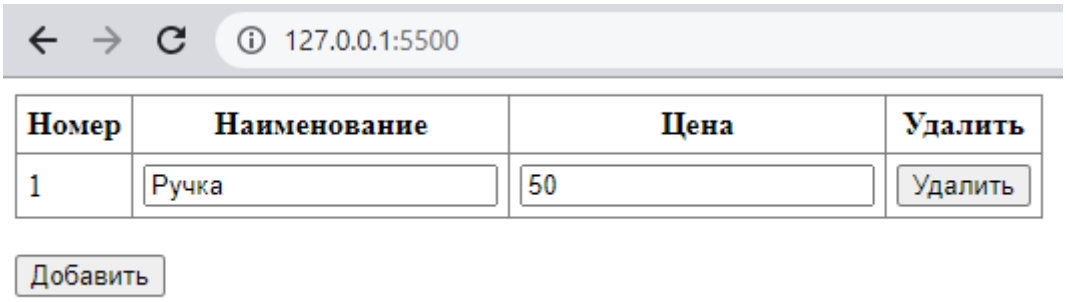

Рисунок 4 – Интерфейс приложения

С помощью кнопки «Добавить» можно создать новое поле для заполнения и добавления товара. Ему будет присвоен новый ID, что видно на рисунке 5.

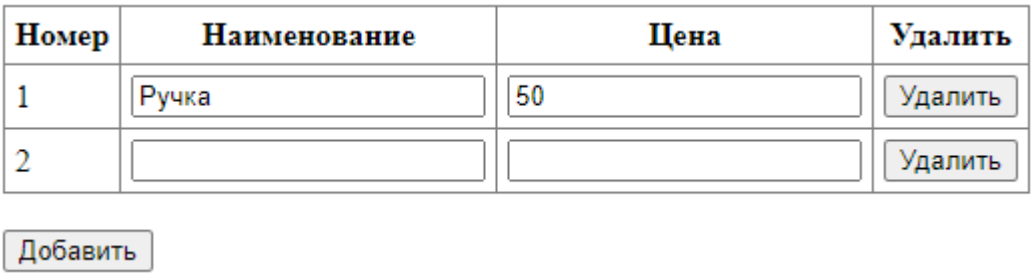

Рисунок 5 – Добавленное поле (результат нажатия на кнопку «Добавить»)

В итоге получаем интерфейс с возможностью создания и удаления полей, который показан на рисунке 4. Поле «Наименование» имеет тип text, а поле «Цена» имеет тип поля number. Введённые данные видно на рисунке 6.

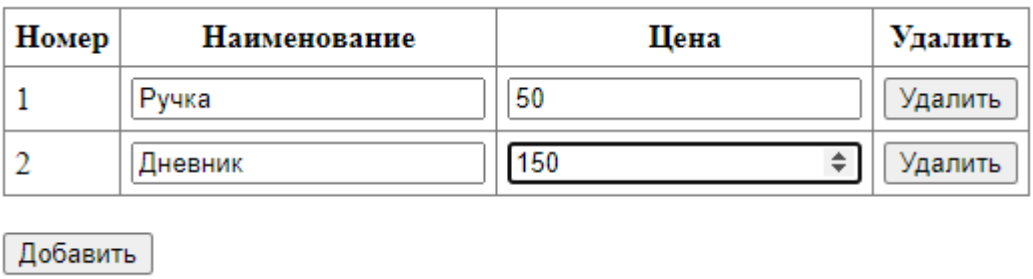

Рисунок 6 – Введённые данные для новых полей

Также присутствует возможность удаление, но при этом ID товаров не перезаписываются, что создаёт возможность написания серверной части приложения без исправлений в клиентской, т.к. при администрировании баз

данных нельзя допускать полного удаление ID или их изменений и перезаписи. Это видно на рисунке 7.

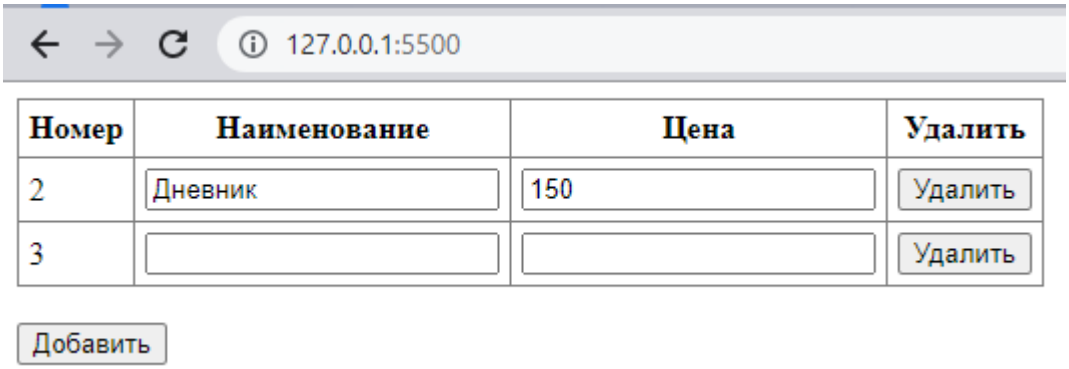

Рисунок 7 – Удаление первого товара и добавление строчки для нового с сохранением ID

Таким образом, создан интерфейс веб приложения с помощью библиотеки React. Этот интерфейс может быть использован в других проектах, т.к. использует независимость и модульность, что и является одним из главных преимуществ React, помимо его удобного языка JSX, сочетающего в себе синтаксис JavaScript и HTML, а также множества команд и функций, встроенных в библиотеку и заменяющих сотни строк чистого JS кода на одну функцию в React. Эта библиотека создана Facebook и используется множеством именитых IT-гигантов.

Библиографический список

1. Винокуров А. Ю., Леванов А. А. REACT. JS: компонентный подход к разработке электронных курсов // Электронное обучение в непрерывном образовании. 2017. №. 1. С. 60-64.
2. Меркулова О. В. Сравнительный анализ JavaScript-фреймворков // Альманах научных работ молодых ученых университета ИТМО. 2019. С. 194-197.
3. Крюков А. С. Исследование особенностей библиотеки React.Js // Вопросы технических и физико-математических наук в свете современных исследований. 2020. С. 9-14.
4. Пантелеева Е. В. Разработка сайта с использованием языка разметки HTML, таблицы стилей CSS и языка программирования JavaScript. // Информационные системы и технологии в образовании, науке и бизнесе. 2020. С. 94-96.
5. Лушников Н. Д., Альтерман А. Д. Основы каскадных таблиц стилей (CSS). // Наука и образование: новое время. 2019. №. 1. С. 69-72.
6. Надеинский Л. А. Архетип модуля каскадных таблиц стилей с LESSCSS компилятором. М., 2018.
7. Айдарбаев Н. О. Адаптивный дизайн веб-сайта с использованием фронтэнд-фреймворка Bootstrap // Молодой ученый. 2018. №. 21. С. 115-

119.
8. Макарова Т. В. Веб-дизайн. М., 2015.
9. Список компаний, которые используют React для написания своих приложений // URL: https://brainhub.eu/library/famous-apps-using-reactjs/
10. Babel // URL: https://babeljs.io/ (дата обращения: 21.08.2021)
11. React // URL: https://ru.reactjs.org/ (дата обращения: 25.08.2021)
12. Redux // URL: https://redux.js.org/ (дата обращения: 26.08.2021)