

Реализация адаптивного hero-слайдера на основе Swiper.js в React-приложении образовательного учреждения

Андриенко Иван Сергеевич

Приамурский государственный университет имени Шолом-Алейхема

Студент

Пащенко Екатерина Александровна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье рассматривается реализация главного слайдера (hero-блока) на главной странице веб-сайта, разработанного с использованием библиотеки React и компонента Swiper.js. Описывается архитектура компонента, включающая загрузку конфигурации слайдов с серверного API, обработку отсутствующих данных с использованием заглушек, настройку анимации переходов слайдов с эффектом «затухания», а также организацию пользовательского взаимодействия через кликабельные слайды и адаптивный текстовый блок. Приводятся особенности работы с изображениями, включая обработку ошибок загрузки и автоматическую подмену на плейсхолдер, а также интеграция кастомной верстки и иконок для формирования единого визуального стиля.

Ключевые слова: слайдер, Swiper.js, React, главная страница, hero-блок, веб-интерфейс, адаптивный дизайн, API.

Implementation of an adaptive hero slider based on Swiper.js in the React application of an educational institution

Andrienko Ivan Sergeevich

Sholom-Aleichem Priamursky State University

Student

Paschenko Ekaterina Alexandrovna

Sholom-Aleichem Priamursky State University

Student

Abstract

The article discusses the implementation of the main slider (hero block) on the main page of a website developed using the React library and the Swiper component.js. The architecture of the component is described, which includes loading the slide configuration from the server API, processing missing data using stubs, configuring animation of slide transitions with a "fade-in" effect, as well as organizing user

interaction through clickable slides and an adaptive text block. The article describes the features of working with images, including handling loading errors and automatic substitution for a placeholder, as well as the integration of custom layout and icons to form a unified visual style.

Keywords: slider, Swiper.js, React, home page, hero block, web interface, responsive design, API.

1 Введение

1.1 Актуальность

Главный слайдер на главной странице сайта образовательного учреждения выполняет одновременно информационную и имиджевую функции, являясь первым визуальным элементом, с которым взаимодействует пользователь. Через него транслируются ключевые сообщения: сведения об образовательных программах, преимуществах учебного заведения, студенческой жизни и актуальных событиях. В условиях высокой конкуренции между образовательными ресурсами важной задачей становится создание динамичного и адаптивного hero-блока, который не только визуально привлекателен, но и тесно интегрирован с системой управления контентом. При этом реализация должна учитывать особенности современного фронтенд-стека, возможность работы с API, обработку ошибок и поддержку различных размеров экранов. Использование библиотеки Swiper.js в связке с React обеспечивает гибкий и хорошо управляемый инструмент для построения подобного интерфейсного компонента, что делает исследуемое решение актуальным для веб-разработчиков в сфере образования.

1.2 Обзор исследований

Перед анализом собственной реализации важно рассмотреть научные работы, посвящённые использованию слайдеров и компонентных библиотек в React-приложениях. В работе М. С. Берьянова рассматриваются подходы к интеграции слайдера в интерфейс React-приложения, включая организацию структуры компонентов и обработку пользовательских событий [1]. Исследование В. В. Бронской и А. Р. Ибатуллиной демонстрирует практическое применение React при создании пользовательских интерфейсов, где используются динамические элементы, включая визуальные карусели и интерактивные блоки [2]. В статье Д.А. Хайруллина и М.Р. Богданова рассматриваются особенности применения статической типизации в React-приложениях, включая методики типизации компонентов, хуков и состояния, а также влияние строгой типизации на надёжность пользовательских интерфейсов [3]. К.Р. Удальцов исследует вопросы обеспечения качества кода в командной разработке, предлагая формализацию шаблонов безопасных pull request-ов для проектов на React и TypeScript в рамках Git Flow [4]. Особое внимание созданию библиотек компонентов на основе TypeScript уделяется в работе Т.М. Воронко, где описываются принципы модульности, повторного

использования кода и проектирования архитектуры компонентов для масштабируемых React-решений [5].

1.3 Цель исследования

Целью данной работы является разработка и описание реализации главного слайдера на основе Swiper.js в React-приложении, предназначенном для сайта образовательного учреждения. В рамках исследования рассматриваются методика загрузки данных о слайдах с серверного API, организация fallback-механизмов при отсутствии данных, настройка параметров слайдера (автопрокрутка, эффект затухания, пагинация и навигация), а также реализация пользовательского взаимодействия при переходе по ссылкам, встроенным в слайды.

2 Материалы и методы

В качестве основы был использован фронтенд-приложение на базе библиотеки React, где главная страница реализована в виде компонента Home. Для создания слайдера применялся компонент Swiper из пакета Swiper.js с использованием модулей Navigation, Pagination, Autoplay и EffectFade, обеспечивающих отображение стрелок навигации, точек пагинации, автоматическую смену слайдов и плавный переход с эффектом «fade».

3 Результаты и обсуждения

Создание главного hero-слайдера на основе библиотеки Swiper.js позволила создать динамичный и адаптивный визуальный блок, занимающий центральное место на главной странице веб-приложения. Первым этапом разработки стала настройка компонентной структуры и подключение необходимых модулей Swiper.js, включая Navigation, Pagination, Autoplay и EffectFade. Эти модули обеспечивают переходы между слайдами, отображение элементов навигации, автоматическую прокрутку и плавную анимацию затухания. Фрагмент подключаемых зависимостей представлен на рисунке (рис. 1).

```
frontend > src > pages > Home.tsx > ...  
1  import { useState, useEffect } from 'react';  
2  import { Swiper, SwiperSlide } from 'swiper/react';  
3  import { Navigation, Pagination, Autoplay, EffectFade } from 'swiper/modules';  
4  import { FaChevronRight } from 'react-icons/fa';  
5  |  
6
```

Рис. 1. Импорт компонентов Swiper.js и необходимых модулей для работы слайдера

Для типизации данных слайда был определён интерфейс SliderItem, включающий ключевые параметры отображения: заголовок, подзаголовок, изображение и опциональную ссылку для перехода на внешний ресурс или внутренний раздел сайта. Такая структура обеспечивает единый формат

данных для всех слайдов и упрощает последующую обработку в компоненте. Код интерфейса приведён на рисунке (рис. 2).

```
7   interface SliderItem {  
8       id: string;  
9       title: string;  
10      subtitle?: string;  
11      image: string;  
12      link?: string;  
13  }  
14
```

Рис. 2. Интерфейс SliderItem, описывающий структуру данных слайда

Загрузка содержимого осуществлялась в асинхронной функции `loadHomeData`, которая обращается к серверному эндпоинту `/api/page-data?page=home`. Полученные данные извлекаются из блока `homeSlider`, где могут присутствовать пользовательские конфигурации, загруженные через административную панель. При наличии слайдов в базе данные преобразуются в массив объектов `SliderItem`. Если же сервер возвращает пустую коллекцию или недоступен, система автоматически использует резервные наборы слайдов, сформированные заранее. Таким образом обеспечивается бесперебойная работа интерфейса. Фрагмент обработки данных представлен на рисунке (рис. 3).

```
104 // загружаем слайды из api  
105 const slidesResponse = await fetch('/api/home-slider');  
106 const slidesData = await slidesResponse.json();  
107  
108 if (slidesData && slidesData.length > 0) {  
109     setSliderItems(slidesData.map((slide: any) => ({  
110         id: slide.id,  
111         title: slide.title,  
112         subtitle: slide.subtitle,  
113         image: slide.imageUrl,  
114         link: slide.link  
115     })));  
116 }  
117
```

Рис. 3. Загрузка данных слайдера из API и использование fallback-заглушек при отсутствии данных

Сама визуальная часть реализована внутри JSX-структуры компонента `Home`. Здесь создаётся `Swiper`, которому передаются параметры отображения: эффект затухания, кросс-фейд анимация, задержка автопрокрутки, бесконечный цикл, кликабельная пагинация и стрелки навигации. Каждый объект массива `sliderItems` отображается как отдельный `SwiperSlide`, внутри которого располагается изображение с обработчиками загрузки и ошибок, а

также текстовая область, появляющаяся поверх картинки при наличии заголовка. Полный фрагмент JSX-разметки показан на рисунке (рис. 4).

```

161 <div>
162   { /* слайдер */ }
163   <section className="relative w-full">
164     {sliderItems.length > 0 ? (
165       <Swiper
166         modules={[Navigation, Pagination, Autoplay, EffectFade]}
167         effect="fade"
168         fadeEffect={{ crossFade: true }}
169         spaceBetween={0}
170         slidesPerView={1}
171         navigation
172         pagination={{ clickable: true }}
173         autoplay={{
174           delay: 5000,
175           disableOnInteraction: false,
176         }}
177         loop={true}
178         className="w-full"
179       >
180         {sliderItems.map((item) => [
181           <SwiperSlide key={item.id} >
182             <div
183               className={`relative w-full ${item.link ? 'cursor-pointer' : ''}`}
184               onClick={() => item.link && window.open(item.link, '_blank')}
185             >
186               <img
187                 src={item.image}
188                 alt={item.title}
189                 className="w-full h-auto transition-all duration-500"
190                 style={{
191                   objectFit: 'contain',
192                   maxHeight: '80vh',
193                 }}
194                 onError={(e) => {
195                   console.error('Image failed to load:', item.image);
196                   e.currentTarget.src = '/placeholder-image.png';
197                 }}
198                 onLoad={() => {}}
199             />

```

Рис. 4. JSX-структура слайдера с настройками Swiper и визуальным оформлением контента

Отдельное внимание уделено обработке изображений. Параметры `objectFit: 'contain'` и ограничение `maxHeight` до 80 % высоты окна гарантируют корректное отображение изображений различных пропорций. При возникновении ошибки загрузки выполняется автоматическая замена изображения на плейсхолдер по умолчанию, что предотвращает визуальные артефакты и обеспечивает стабильную работу интерфейса.

Система взаимодействия пользователя со слайдером реализована через обработчик клика по изображению: при наличии ссылки в объекте слайда выполняется открытие страницы в новой вкладке. Таким образом, слайдер выполняет не только декоративную, но и навигационную функцию. Текстовые элементы — заголовок, подзаголовок и кнопка — адаптируются под размеры экрана с помощью заранее настроенных CSS-классов и позиционируются в нижней центральной части слайда, что повышает читаемость. Итоговый визуальный вид слайдера на странице представлен на рисунке (рис. 5).

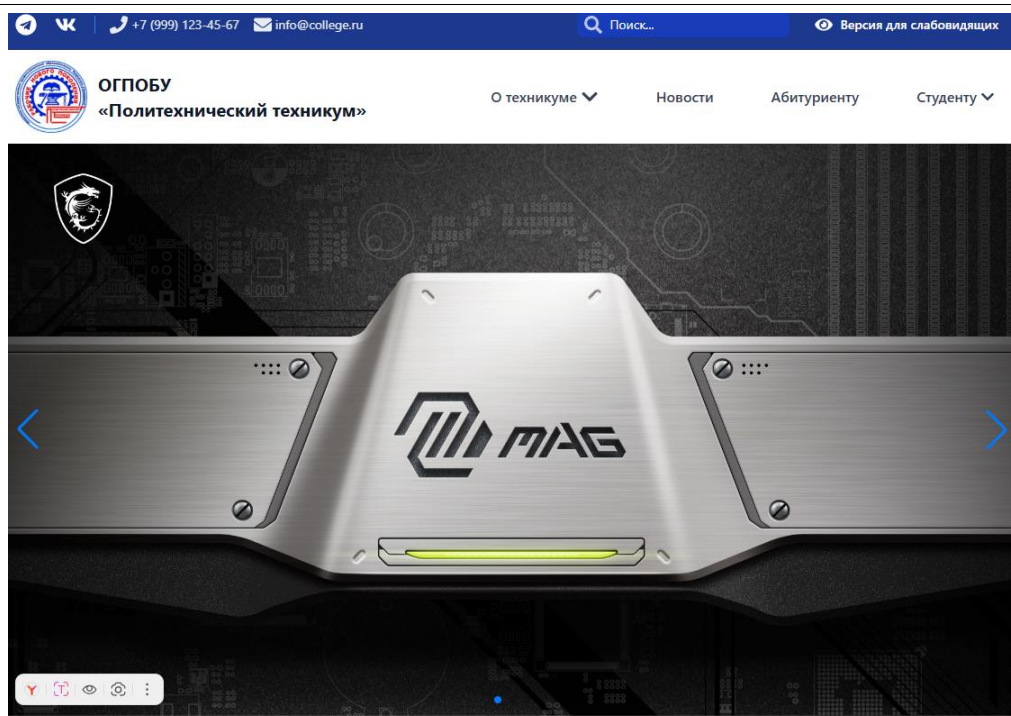


Рис. 5. Отображение сайта при выключенном режиме слабовидящих

4 Выводы

Реализация hero-слайдера на основе Swiper.js в React-приложении показала, что использование модульной структуры библиотеки и динамической загрузки данных позволяет создать функциональный, визуально выразительный и полностью адаптивный элемент главной страницы. Проведённое тестирование подтвердило стабильную работу автопрокрутки, корректное отображение заголовков и изображений, а также надёжность fallback-механизмов при отсутствии данных или ошибках загрузки. Итоговая архитектура обеспечила удобство обновления контента и расширяемость компонента, что делает предложенный подход эффективным решением для современных образовательных веб-ресурсов.

Библиографический список

1. Берьянов М. С. Внедрение компонента-слайдера (карусели) в React-приложение // Студенческий. 2023. № 1-1 (213). С. 32–34.
2. Бронская В. В., Ибатуллина А. Р. Применение JavaScript-библиотеки React для создания приложения интернет-магазина одежды // Известия высших учебных заведений. Технология текстильной промышленности. 2023. № 4 (406). С. 169–175.
3. Хайруллин Д.А., Богданов М.Р. Изучение использования TypeScript в React-приложениях, преимущества статической типизации, подходы к типизации компонентов, хуков и состояния // Международный журнал гуманитарных и естественных наук. 2024. № 12-3 (99). С. 233–238.
4. Удалцов К.Р. Формализация шаблонов безопасных pull request-ов для React-приложений с TypeScript в командах с Git Flow // Актуальные

-
- проблемы социально-гуманитарного и научно-технического знания. 2025. № 4 (44). С. 35–36.
5. Воронко Т.М. Создание React-библиотеки с использованием TypeScript // Электронные системы и технологии. Материалы 58-й научной конференции аспирантов, магистрантов и студентов БГУИР. Минск, 2022. С. 200–202.