

## Реализация адаптивной динамической карусели изображений с использованием React и Swiper

*Вихляев Дмитрий Романович*

*Приамурский государственный университет имени Шолом-Алейхема*

*Студент*

### Аннотация

В статье рассматривается технология создания адаптивной карусели изображений, автоматически адаптирующейся к различным размерностям медиаконтента (портретной, альбомной и квадратной ориентации) с сохранением визуальной целостности и пропорций. Результатом исследования станет готовый React-компонент с автоматическим переключением слайдов.

**Ключевые слова:** React, CSS-модули, адаптивная верстка.

### Implementing an adaptive dynamic image carousel using React and Swiper

*Vikhlyayev Dmitry Romanovich*

*Sholom-Aleichem Priamursky State University*

*Student*

### Abstract

The article discusses the technology of creating an adaptive image carousel that automatically adapts to various dimensions of media content (portrait, landscape and square orientation) while maintaining visual integrity and proportions. The result of the research will be a ready-made React component with automatic slide switching.

**Keywords:** React, CSS modules, adaptive layout.

## 1 Введение

### 1.1 Актуальность

Swiper представляет собой современную библиотеку JavaScript, специализированную для создания адаптивных, сенсорно-ориентированных слайдеров и каруселей. Архитектурно библиотека построена на модульной системе, где базовый функционал расширяется плагинами для навигации, пагинации, автопрокрутки и визуальных эффектов. Ключевым технологическим преимуществом Swiper является нативная поддержка сенсорных жестов (swipe, drag), аппаратное ускорение анимаций через CSS transforms и глубокая интеграция с фреймворками React, Vue и Angular. Эффект Coverflow, реализованный в отдельном модуле, использует трехмерные трансформации CSS (transform: rotate3d, translate3d) для создания перспективного отображения слайдов, имитирующего перелистывание элементов в пространстве. Библиотека автоматически рассчитывает

геометрические параметры каждого слайда в зависимости от его позиции в виртуальном стеке, обеспечивая плавные переходы между состояниями.

### **1.2 Обзор исследований**

В работе М.С.Берьянова рассматриваются подходы к интеграции слайдера в интерфейс React-приложения, включая организацию структуры компонентов и обработку пользовательских событий [1]. В.В.Бронская, А.Р.Ибатуллина продемонстрировали создание пользовательских интерфейсов, включающих визуальные карусели и интерактивные блоки [2]. В статье Д.А.Хайруллина, М.Р.Богданова рассматриваются особенности применения статической типизации в React приложениях, включая влияние строгой типизации на надёжность пользовательских интерфейсов [3]. К.Р.Удальцов исследует вопросы обеспечения качества кода в командной разработке для проектов на React и TypeScript в рамках Git Flow [4]. Т.М.Воронко описал принципы модульности, повторного использования кода и проектирования архитектуры компонентов для масштабируемых React-решений [5]. Исследование И.А.Шадрова и Л.В.Гаева содержит современные подходы в создании пользовательских интерфейсов на React.js [6]. Влияние передовых технологий фронтенд-разработки на улучшение пользовательского интерфейса и повышение уровня взаимодействия с веб-приложениями приведено в работе О.С.Благодельского [7].

### **1.3 Цель исследования**

Цель исследования – разработать адаптивную динамическую карусель изображений с использованием библиотеки React и модуля Swiper, которая гарантирует сохранение пропорций исходных изображений независимо от их ориентации.

## **2 Материалы и методы**

Для реализации используется вариация модуля Swiper для платформы ReactJS. Эффект Coverflow, параметризованный через геометрические настройки в сочетании с продуманной системой визуального акцентирования.

## **3 Результаты и обсуждения**

Для начала необходимо импортировать модули (Navigation, Pagination, Autoplay, EffectCoverflow). Их активация в компоненте «Swiper» создает комплексную систему управления каруселью. Используемый эффект «Cover Flow» является визуальным шаблоном проектирования пользовательского интерфейса. В его основе элементы отображаются горизонтально, а центральный элемент выражен в переднем плане. При этом происходит имитация 3D-эффекта, когда соседние элементы располагаются под углом, также могут добавляться тени и другие визуальные эффекты. В модуле «Swiper» параметры «coverflowEffect» (rotate, stretch, depth, modifier) определяют геометрию в трехмерном пространстве, такие как угол поворота,

коэффициент растяжения по осям, глубину перспективы и множитель преобразований соответственно.

Установка параметра «slidesPerView» равным 3 и определения «centeredSlides» в значения (true), формирует стабильную визуальную композицию с тремя одновременно видимыми элементами, где центральный выставлен в переднюю позицию. Метод для автопрокрутки (autoplay: delay 3000) реализован через интервальные таймеры, которые автоматически приостанавливаются при пользовательском взаимодействии. Навигационные элементы (стрелки) и пагинация (индикаторы) подключаются декларативно, их поведение и стилизация полностью контролируются библиотекой с возможностью кастомизации через CSS-переменные и хуки жизненного цикла (рис.1).

```
import { Swiper, SwiperSlide } from 'swiper/react';
import { Navigation, Pagination, Autoplay, EffectCoverflow } from 'swiper/modules';
import 'swiper/css';
import 'swiper/css/navigation';
import 'swiper/css/pagination';
import 'swiper/css/effect-coverflow';
import { Navigation, Pagination, Autoplay, EffectCoverflow } from 'swiper/modules';

<Swiper
  modules={[Navigation, Pagination, Autoplay, EffectCoverflow]}
  effect="coverflow"
  coverflowEffect={{
    rotate: 10,
    stretch: 40,
    depth: 60,
    modifier: 1.5,
    slideShadows: false,
  }}
  spaceBetween={20}
  slidesPerView={3}
  centeredSlides={true}
  navigation
  pagination={{ clickable: true }}
  autoplay={{ delay: 3000 }}
  loop={true}
  grabCursor={true}
  initialSlide={1}
/>
```

Рис. 1. Конфигурация Swiper с эффектом Coverflow

Всё содержимое, включая изображения, содержится в структурном элементе массива slides. Который представляет декларативное описание контента. Каждый элемент содержит внутри себя идентификатор, путь к изображению и метаданные.

В процессе рендеринга метод массива «map» преобразует данные в компоненты объекта «SwiperSlide», которые инкапсулируют индивидуальную разметку для каждого элемента.

Архитектура компонента разделяет ответственность между внешним контейнером «slide-card», определяющим общие стилевые рамки, внутренним блоком «image-frame», отвечающим за позиционирование медиаконтента, и нижней панели «slide-label» выводящей метаинформацию. Такое разделение позволяет независимо модифицировать отдельные слои визуальных композиций (рис.2).

```

const slides = [
  { id: 1, image: mountain, title: 'Горы' },
  { id: 2, image: beach, title: 'Пляж' },
  { id: 3, image: f1, title: 'Лес' },
  { id: 4, image: forest, title: 'Море' },
  { id: 5, image: city, title: 'Амазонка' },
];

{slides.map((slide) => (
  <SwiperSlide key={slide.id} className="coverflow-slide">
    <div className="slide-card">
      <div className="image-frame">
        <img
          src={slide.image}
          alt={slide.title}
          className={`slide-image ${slide.orientation}`}
        />
        <div className="frame-border"></div>
      </div>
      <div className="slide-label">
        <span className="slide-number">#{slide.id}</span>
        <span className="slide-title">{slide.title}</span>
      </div>
    </SwiperSlide>
  )]}

```

Рис. 2. Структура данных слайдов и рендеринг с динамическими классами

Стили компонента используют принцип контекстно-зависимого масштабирования. Сначала применяются ограничивающие параметры через свойства «max-width», «max-height» к элементам изображений. Ключевое свойство «object-fit» со значением contain гарантирует сохранение исходных пропорций контента при его вписывании в заданные границы, предотвращая геометрические искажения. Благодаря этому изображения не растягиваются до размера родительского контейнера и не обрезаются края. Контейнер со свойством «display» равным значению flex и параметрами центрирования для «align-items» и «justify-content», создает адаптивную область, которая автоматически выравнивает контент по осям независимо от его размеров. Установка overflow: visible критически важна для работы эффекта Coverflow, так как разрешает элементам выходить за пределы родительского блока, формируя иллюзию глубины и перекрытия слайдов. Дополнительный декоративный элемент .frame-border, позиционированный абсолютно, визуально обрамляет область контента без влияния на layout (рис.3).

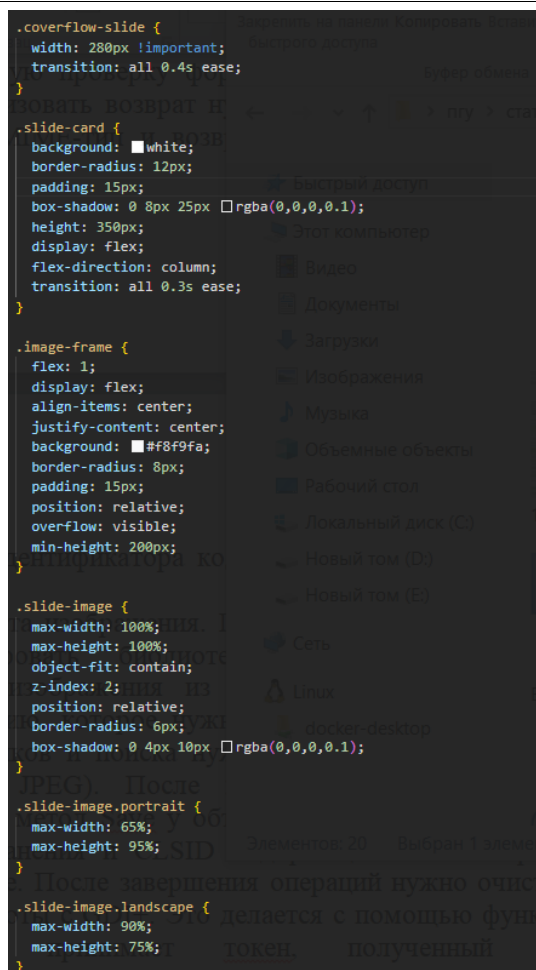


Рис. 3. Стили для адаптации изображений разных ориентаций и контейнера

Используется простой визуальный приём, который помогает сразу понять, какой слайд сейчас главный. Параметр свойство «transform» имеет функцию «scale», в которую передаётся значение 1.15 для центрального слайда, ближайшие боковые уменьшаются до 0.9, а остальные получают минимальный масштаб (0.8). Библиотека «Swiper» сама определяет, какой слайд находится в центре, а какие рядом, и назначает им специальные классы. На основе этих классов задаются размеры, порядок наложения и внешний вид элементов. Чтобы слайды не перекрывали друг друга некорректно, используется разный приоритет Z-индекса. Центральный слайд находится поверх остальных. Дополнительные эффекты, такие как тень и цветная рамка, ещё больше подчёркивают активный слайд. Вся логика работает автоматически под руководством «Swiper», которая отслеживает положение слайдов на экране и в реальном времени, применяя нужные стили и обеспечивая плавное и понятное визуальное поведение карусели (рис. 4).

```

:global(.swiper-slide-active) {
  transform: scale(1.15) !important;
  z-index: 10 !important;
}
4: Иерархия визуального акцента через масштабирование и состояние слайдов

:global(.swiper-slide-active .slide-card) {
  box-shadow: 0 15px 35px rgba(0,0,0,0.15);
  border: 2px solid #3498db;
}
:global(.swiper-slide-prev),
:global(.swiper-slide-next) {
  transform: scale(0.9) !important;
  opacity: 0.95 !important;
  z-index: 5 !important;
}
:global(.swiper-slide-active .slide-card) {
  box-shadow: 0 15px 35px rgba(0,0,0,0.15);
}
:global(.swiper-slide:not(.swiper-slide-active):not(.swiper-slide-prev):not(.swiper-slide-next)) {
  transform: scale(0.8) !important;
  opacity: 0.8 !important;
  z-index: 1 !important;
}

```

Рис. 4. Иерархия визуального акцента через масштабирование и состояние слайдов

Чтобы сайт выглядел корректно на экранах разного размера, используется медиа-запросы. При уменьшении ширины экрана (1024, 768 и 480 пикселей) сайт постепенно упрощается, меняя ширину основного блока, размеры слайдов и карточек, а также масштаб элементов. Процесс происходит не резко, а поэтапно, чтобы интерфейс оставался удобным.

Размер слайдов в карусели задаётся принудительно, чтобы библиотека «Swiper» не изменяла их автоматически. Это сохраняет одинаковый внешний вид элементов независимо от устройства. На маленьких экранах изображения становятся немного уже с 65% до 60% от ширины контейнера. Это компенсирует уменьшение общего пространства и помогает сохранить читаемость и аккуратный внешний вид.

В итоге сайт выглядит одинаково аккуратно и предсказуемо на всех устройствах, без необходимости делать отдельные версии интерфейса. Изображения и контейнеры в карусели сохраняют свои размеры и форму (рис. 5, 6).

```

@media (max-width: 1024px) {
  .visible-coverflow {
    max-width: 900px;
    padding: 30px 15px;
  }
  .coverflow-slide {
    width: 240px !important;
  }
  .slide-card {
    height: 320px;
    padding: 20px 10px;
  }
}

@media (max-width: 768px) {
  .visible-coverflow {
    max-width: 700px;
    padding: 20px 10px;
  }
  .coverflow-slide {
    width: 200px !important;
  }
  .slide-card {
    height: 280px;
    padding: 12px;
  }
  :global(.swiper-slide-active) {
    transform: scale(1.1) !important;
  }
  .slide-image.portrait {
    max-width: 60%;
  }
}

@media (max-width: 480px) {
  .visible-coverflow {
    max-width: 100%;
    padding: 15px 5px;
  }
  .coverflow-slide {
    width: 180px !important;
  }
  .slide-card {
    height: 250px;
    padding: 10px;
  }
  :global(.swiper-button-next),
  :global(.swiper-button-prev) {
    display: none;
  }
}

```

Рис. 5. Адаптивные медиа запросы для различных устройств

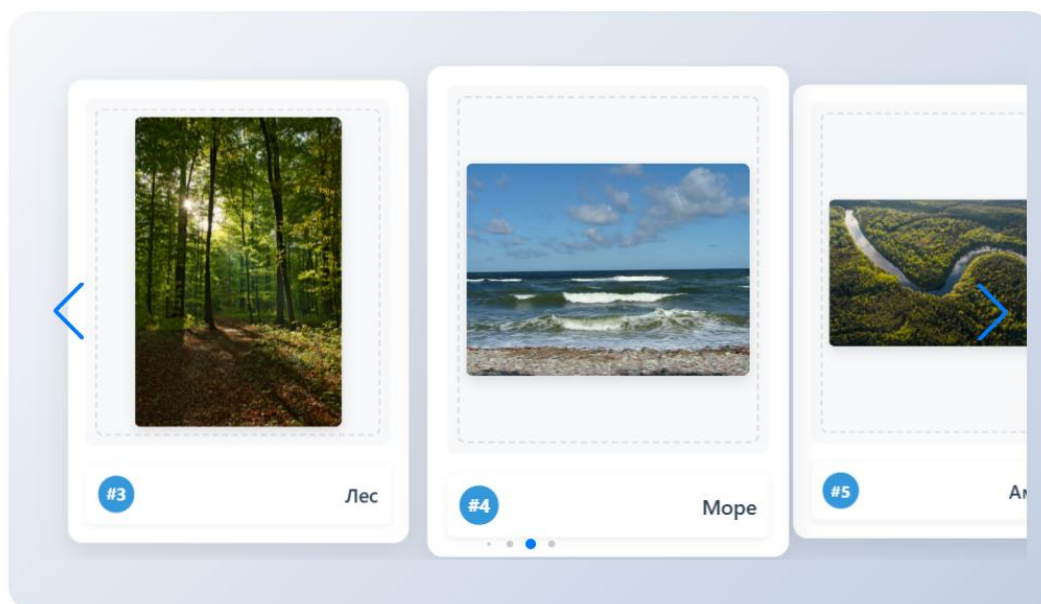


Рис. 6. Адаптивная карусель изображений

Разработанный дизайн является адаптивным, он позволяет сайту одинаково отображаться на устройствах с разным размером экрана. По мере уменьшения экрана интерфейс становится проще. Элементы уменьшаются в размере, а менее важные детали скрываются. При этом сохраняются основные компоненты сайта, а текст и изображения становятся удобными для восприятия.

#### 4 Вывод

В результате работы описано решение построения современной карусели изображений. Представлена программная реализация исходного кода. Приведено обоснование логики и требования к адаптивности за счет использования библиотек и CSS-стилей. Полученные результаты могут служить основой для разработки аналогичных компонентов, требующих одновременного выполнения семантической ясности и технической надежности.

#### 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. Удальцов К.Р. Формализация шаблонов безопасных pull request-ов для React-приложений с TypeScript в командах с Git Flow // Актуальные Постулат. 2025. №12 ISSN 2414-4487 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.
6. Шадров И.А., Гаев Л.В. React в современной веб-разработке: ключевые задачи и решения // Инновационная наука. 2025. № 5-2. С. 88-89.
7. Благодельский О.С. Влияние передовых технологий фронтенд-разработки на улучшение пользовательского интерфейса и повышение уровня взаимодействия с веб-приложениями // Актуальные исследования. 2024. № 20-1 (202). С. 73-79.