

## **Реализация веб-версии для слабовидящих на основе React и CSS-глобальных стилей**

*Андриенко Иван Сергеевич*

*Приамурский государственный университет имени Шолом-Алейхема*

*Студент*

*Пащенко Екатерина Александровна*

*Приамурский государственный университет имени Шолом-Алейхема*

*Студент*

### **Аннотация**

В данной работе рассматривается подход к реализации версии сайта для слабовидящих на основе фронтенд-стека React и глобальных CSS-стилей. Описывается механизм переключения режима с помощью кнопки в шапке сайта, использование состояния приложения и локального хранилища браузера для сохранения выбора пользователя, а также применение CSS-класса высшего уровня к элементу `document.documentElement` для переопределения типографики, цветовой палитры и размеров элементов управления на всех страницах. Приводится практический пример реализации: использование состояния `isAccessibilityMode`, сохранение его значения в `localStorage`, а также определение каскада стилей `.accessibility-mode` для увеличения шрифта, улучшения контраста, усиления визуальной заметности ссылок и кнопок, адаптации хедера и футера к потребностям слабовидящих пользователей. Показано, что предложенный подход позволяет реализовать единый режим повышенной доступности без изменения структуры компонентов и логики маршрутизации.

**Ключевые слова:** доступность, версия для слабовидящих, веб-доступность, React, CSS, пользовательский интерфейс, локальное хранилище, UX.

## **Implementation of a web version for the visually impaired based on React and CSS global styles**

*Andrienko Ivan Sergeevich*

*Sholom-Aleichem Priamursky State University*

*Student*

*Paschenko Ekaterina Alexandrovna*

*Sholom-Aleichem Priamursky State University*

*Student*

**Abstract**

This paper considers an approach to implementing a website version for the visually impaired based on the React frontend stack and global CSS styles. It describes the mechanism for switching the mode using a button in the header of the site, using the application state and local browser storage to save user selections, as well as applying a top-level CSS class to the `document.documentElement` element to redefine the typography, color palette, and size of controls on all pages. A practical implementation example is given: using the `isAccessibilityMode` state, storing its value in `localStorage`, and defining a cascade of `styles.accessibility-mode` for enlarging the font, improving contrast, enhancing the visual visibility of links and buttons, and adapting the header and footer to the needs of visually impaired users. It is shown that the proposed approach makes it possible to implement a single mode of increased availability without changing the structure of components and routing logic.

**Keywords:** accessibility, version for the visually impaired, web accessibility, React, CSS, user interface, local storage, UX.

**1 Введение****1.1 Актуальность**

В условиях массового распространения цифровых сервисов обеспечение доступности веб-интерфейсов для людей с ограничениями по зрению становится важной задачей как с социальной, так и с технической точки зрения. Пользователи с ослабленным зрением нуждаются в увеличенных размерах шрифта, повышенном контрасте, понятной визуальной иерархии, достаточной величине интерактивных элементов и предсказуемом поведении интерфейса. Отсутствие таких возможностей существенно осложняет работу с сайтом и может сделать его фактически недоступным для значительной группы пользователей. Разработка отдельной версии интерфейса для слабовидящих или выделенного режима отображения с особыми параметрами является одним из практических способов адаптации веб-приложений к потребностям данной категории пользователей. Для одностраничных приложений на React такой режим должен быть реализован так, чтобы распространяться на весь интерфейс, не нарушая архитектуру компонентов и не требуя дублирования разметки. Это обуславливает актуальность разработки подходов, основанных на управлении глобальными стилями и состоянием приложения.

**1.2 Обзор исследований**

Разработка веб-приложений на основе React и TypeScript активно рассматривается в научной литературе. В работе М.М. Ковцура, В.Е. Филимонова и Е.К. Щёголева анализируются подходы к созданию многопользовательских систем на базе React, где особое внимание уделяется организации типизации и архитектуры клиентской части [1]. А.А. Дудак в своём исследовании проводит сравнительный анализ инструментов разработки для проектных систем, выделяя преимущества стека технологий

React и TypeScript по производительности и удобству сопровождения кода [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 реализация и описание программного подхода к созданию версии веб-приложения для слабовидящих на основе React и глобального CSS-класса, применяемого к корневому элементу документа. Исследование направлено на построение механизма переключения режима слабовидящих на уровне шапки сайта, обеспечение сохранения выбора пользователя между сессиями и применение единого набора адаптивных стилей, изменяющих типографику, цветовую схему и размеры элементов управления на всех страницах приложения.

## **2 Материалы и методы**

Для разработки версии сайта для слабовидящих использовалась фронтенд-библиотека React, обеспечивающая компонентный подход и поддержку работы с состоянием через хуки. В качестве инструмента стилизации применялась система глобальных CSS-стилей, подключаемая через файл index.css, что позволило реализовать единый класс оформления .accessibility-mode, влияющий на всё дерево документа.

## **3 Результаты и обсуждения**

Реализация версии сайта для слабовидящих была выполнена на основе компонента Header, в котором размещена кнопка, отвечающая за включение и отключение режима повышенной доступности. В первую очередь была создана логика управления состоянием, позволяющая сохранять выбранный режим между сеансами работы с сайтом. Для этого использовался хук useState с ленивой инициализацией, читающий значение параметра из localStorage. Такой подход обеспечивает автоматическое применение режима слабовидящих при повторном посещении сайта пользователем. Пример использования состояния и чтения значения из локального хранилища представлен на рисунке, содержащем фрагмент кода компонента Header.

Следующим этапом стала настройка механизма динамического изменения оформления сайта. В компоненте Header был использован хук useEffect, в котором выполнялось добавление или удаление глобального CSS-

класса `accessibility-mode` у элемента `document.documentElement`. Такой метод позволяет не изменять структуру компонентов и не вводить дублирование разметки, а централизованно управлять стилями на уровне всего приложения. Внутри эффекта также выполнялась запись актуального значения в `localStorage`, что обеспечивает синхронизацию состояния и интерфейса. Фрагмент реализации данного механизма приведён на рисунке (рис. 1).

```

47 // Применение режима слабовидящих к документу
48 useEffect(() => {
49   if (isAccessibilityMode) {
50     document.documentElement.classList.add('accessibility-mode');
51   } else {
52     document.documentElement.classList.remove('accessibility-mode');
53   }
54   localStorage.setItem('accessibilityMode', isAccessibilityMode.toString());
55 }, [isAccessibilityMode]);
56
57 const toggleAccessibilityMode = () => {
58   setIsAccessibilityMode(!isAccessibilityMode);
59 };
60

```

Рис. 1. Инициализация состояния и логика применения класса `accessibility-mode` через `useEffect`

Для визуального интерфейса была создана кнопка в шапке сайта, включающая иконку глаза и подпись «Версия для слабовидящих». В JSX-разметке реализовано условное применение классов оформления, позволяющее кнопке менять вид при активированном режиме. Такой подход обеспечивает мгновенную обратную связь для пользователя и соответствует рекомендациям по UX-доступности. Визуальный вид кнопки и её структура показаны на рисунке (рис. 2).

```

110 /* Версия для слабовидящих */
111 <button
112   onClick={toggleAccessibilityMode}
113   className={`flex items-center space-x-2 px-3 py-1.5 rounded text-sm font-medium transition-colors ${
114     isAccessibilityMode
115       ? 'bg-blue-700 ring-2 ring-blue-300'
116       : 'hover:bg-blue-800'
117   }`}
118 >
119   <FaEye className="w-5 h-5" />
120   <span className="hidden sm:inline">Версия для слабовидящих</span>
121 </button>
122
123 </div>

```

Рис. 2. Кнопка переключения режима слабовидящих в компоненте `Header`

Основная логика режима слабовидящих реализована в глобальном CSS-файле `index.css`. Центральным элементом выступает класс `.accessibility-mode`, который применяется ко всему документу и влияет на структуру отображения текста, ссылок, кнопок и графических элементов. В блоке базовых правил определены увеличенный размер шрифта, повышенный контраст цветовой

палитры, усиленная толщина линий и адаптированные значения межстрочного интервала, что соответствует общепринятым принципам доступности веб-контента. Данный класс наследуется всеми элементами благодаря конструкции `.accessibility-mode *`, что обеспечивает единообразие отображения. Пример ключевых CSS-правил приведён на рисунке (рис. 3).

```
68  /* для слабовидящих */
69  .accessibility-mode {
70      font-size: 18px;
71      line-height: 1.6;
72      font-weight: 500;
73  }
74
75  .accessibility-mode * {
76      font-size: inherit !important;
77      line-height: inherit !important;
78  }
79
80  .accessibility-mode h1,
81  .accessibility-mode h2,
82  .accessibility-mode h3,
83  .accessibility-mode h4,
84  .accessibility-mode h5,
85  .accessibility-mode h6 {
86      font-size: 1.5em !important;
87      font-weight: bold !important;
88      margin: 1em 0 0.5em 0 !important;
89  }
90
91  .accessibility-mode p,
92  .accessibility-mode li,
93  .accessibility-mode span,
94  .accessibility-mode div {
95      font-size: 1.1em !important;
96      line-height: 1.8 !important;
97  }
98
99  .accessibility-mode button,
100 .accessibility-mode a,
101 .accessibility-mode input,
102 .accessibility-mode select,
103 .accessibility-mode textarea {
104     font-size: 1.1em !important;
105     padding: 12px 16px !important;
106     min-height: 44px !important;
107 }
108
109 .accessibility-mode img {
110     filter: contrast(1.2) brightness(1.1);
111 }
```

Рис. 3. Фрагмент CSS-правил класса `.accessibility-mode`

Отдельные группы правил были выделены для хедера и футера — двух визуально доминирующих частей сайта. В режиме слабовидящих для них

задаются светлый фон, чёткие границы, тёмный текст и увеличенные размеры иконок. Это позволяет устранить визуальный шум и улучшить различимость элементов навигации, что особенно важно для пользователей с нарушениями зрения. Отдельные фрагменты CSS-правил, регулирующих оформление хедера и футера, представлены на рисунке (рис. 4).

```
217 /* Специфические правила для футера в режиме слабовидящих */
218 .accessibility-mode footer {
219     background-color: #f8f9fa !important;
220     color: #000000 !important;
221     border-top: 2px solid #000000 !important;
222 }
223
224 .accessibility-mode footer * {
225     color: #000000 !important;
226 }
227
228 .accessibility-mode footer a {
229     color: #0066cc !important;
230 }
231
232 .accessibility-mode footer .text-gray-300,
233 .accessibility-mode footer .text-gray-400 {
234     color: #333333 !important;
235 }
236
237 .accessibility-mode footer .text-blue-400 {
238     color: #0066cc !important;
239 }
240
241 .accessibility-mode footer .bg-gray-900 {
242     background-color: #f8f9fa !important;
243 }
244
245 .accessibility-mode footer .border-gray-700 {
246     border-color: #cccccc !important;
247 }
248
249 /* Специфические правила для хедера в режиме слабовидящих */
250 .accessibility-mode header {
251     background-color: #ffffff !important;
252     color: #000000 !important;
253     border-bottom: 2px solid #000000 !important;
254 }
255
256 .accessibility-mode header * {
257     color: #000000 !important;
258 }
259
260 .accessibility-mode header .bg-blue-900 {
261     background-color: #f8f9fa !important;
```

Рис. 4. Специфические правила оформления хедера и футера в режиме слабовидящих

После завершения реализации был проведён тестовый запуск сайта. При активировании режима слабовидящих увеличивались размеры текста, усиливался контраст, изменялись размеры интерактивных элементов, а ссылки приобретали подчёркнутый формат и характерный синий цвет, знакомый пользователям. Все изменения применялись мгновенно, без перезагрузки страницы, что указывает на корректную работу механизма динамического переключения стилей. Тестирование показало, что режим сохраняется после обновления страницы и при повторном запуске сайта благодаря использованию localStorage. Результаты тестирования представлены на рисунке (рис. 5,6).

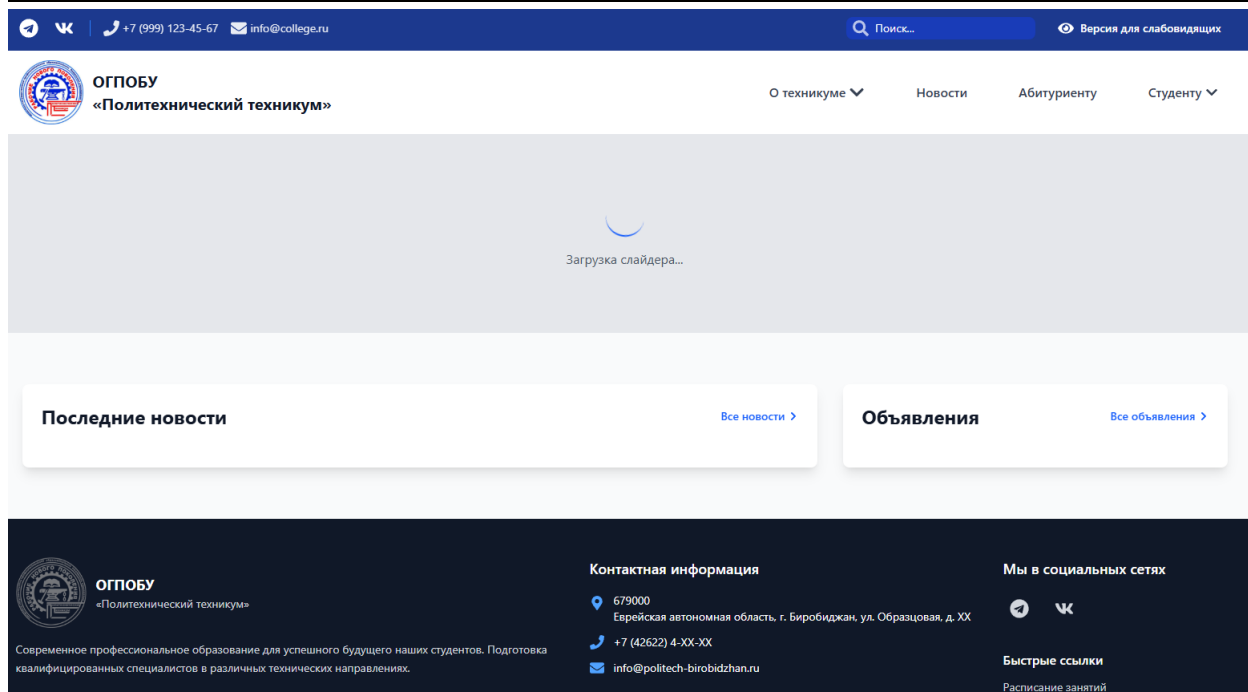


Рис. 5. Отображение сайта при выключенном режиме слабовидящих

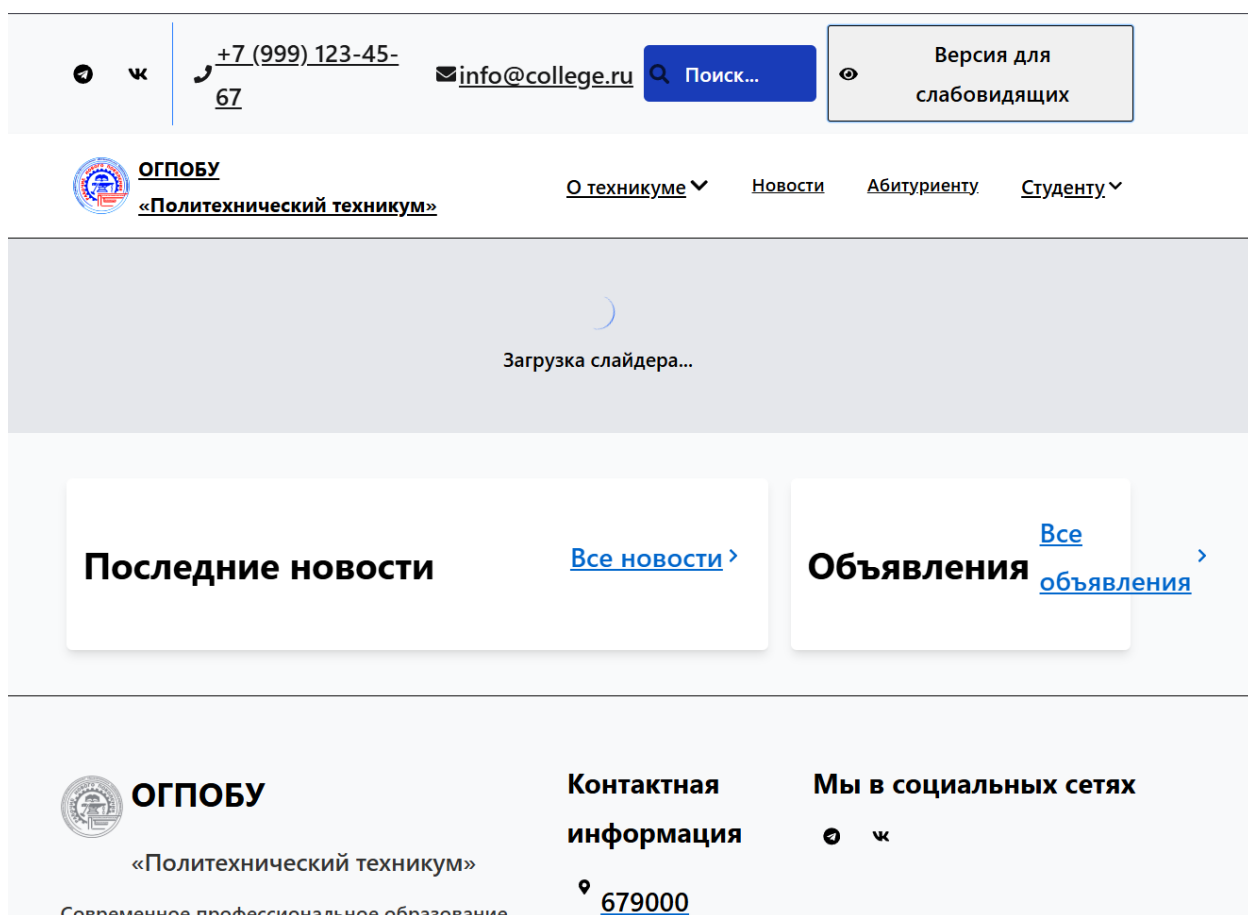


Рис. 6. Отображение сайта при активированном режиме слабовидящих

#### 4 Выводы

В ходе работы была реализована версия веб-приложения для слабовидящих на основе React и глобальных CSS-стилей. Разработан

механизм переключения режима через компонент Header, обеспечивающий мгновенное изменение оформления без перезагрузки страницы. Использование состояния и локального хранилища позволило сохранить выбор пользователя между сессиями, а применение единого класса .accessibility-mode обеспечило адаптацию всех элементов интерфейса под требования доступности. Проведённое тестирование подтвердило корректную работу режима и его влияние на улучшение читабельности и удобства взаимодействия с сайтом для пользователей с нарушениями зрения.

### **Библиографический список**

1. Ковцур М.М., Филимонов В.Е., Щёголев Е.К. Исследование подходов для реализации многопользовательского React-приложения на языке TypeScript // Мобильный бизнес: перспективы развития и реализации систем радиосвязи в России и за рубежом. Материалы 53-й Международной конференции. Москва, 2024. С. 58–61.
2. Дудак А.А. Сравнительный анализ инструментов разработки для систем управления проектами: преимущества стек-технологий TypeScript и React // Новая наука: от идеи к результату. 2024. № 9. С. 32–40.
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 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.