

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Фронт-энд разработка

Отчет  
Домашняя работа 2

Выполнил:

Атанова София

Группа К3343

Проверил:

Добряков Д. И.

Санкт-Петербург

2025 г.

# Доступность в HTML

**Цель:** улучшить доступность ранее реализованного сайта. Добавить необходимые HTML-атрибуты ко всему контенту на странице и проверить это с помощью инструментов из Dev Tools браузера Firefox и сервиса Google Lighthouse.

## 1. Dev Tools Firefox (Accessibility Inspector)

### 1.1 JavaScript ошибка Uncaught SyntaxError: redeclaration of const API\_URL

При открытии страницы в консоли браузера отображалась JavaScript-ошибка Uncaught SyntaxError: redeclaration of const API\_URL. Проблема заключалась в том, что константа API\_URL была объявлена в нескольких подключаемых скриптах, что приводило к конфликту. Для исправления в файле js/api.js было изменено объявление переменной с проверкой на её существование:

```
// Проверяем, не объявлена ли уже переменная API_URL  
var API_URL = (typeof API_URL !== 'undefined') ? API_URL :  
'http://localhost:3000/api';
```

### 1.2 Фокусируемый элемент не имеет стилей для сфокусированного состояния

Accessibility Inspector указал на то, что фокусируемые элементы не имеют стилей для сфокусированного состояния. Это могло бы стать проблемой для гипотетических пользователей, которые перемещались бы по сайту с помощью клавиатуры — они не видят, какой элемент сейчас активен.

Для решения этой проблемы в файл css/styles.css были добавлены стили фокуса для всех интерактивных элементов:

```

/* Стили фокуса для интерактивных элементов */
a:focus-visible,
button:focus-visible,
input:focus-visible,
select:focus-visible,
textarea:focus-visible,
.btn:focus-visible {
    outline: 3px solid var(--primary-blue) !important;
    outline-offset: 2px !important;
}

/* Улучшение контраста для форм */
.form-control:focus,
.form-select:focus {
    border-color: var(--primary-blue);
    box-shadow: 0 0 0 3px rgba(79, 134, 247, 0.2);
}

/* Карточки при фокусе */
.card:focus-within {
    box-shadow: 0 0 0 2px var(--primary-blue);
}

/* убедимся что .light-blue-text виден */
.light-blue-text {
    color: var(--primary-blue) !important;
}

```

### 1.3 Фокусируемые элементы должны иметь интерактивную семантику

- Секция со списком выполненных тренировок содержала спиннер загрузки с неполной семантикой
- Используются `<div>` или `<span>` вместо семантических тегов `<button>`, `<a>`
- Отсутствуют атрибуты `aria-label`, `role`, `aria-describedby`
- Карточки тренировок генерировались без семантических тегов.

В файле index.html спиннер был дополнен необходимыми ARIA-атрибутами:

```
<!-- Спиннер загрузки -->|
<div class="spinner"
  role="status"
  aria-label="Загрузка данных">
  <span class="visually-hidden">Загрузка...</span>
</div>
```

Атрибут `role="status"` сообщает скринридерам, что это индикатор состояния. Атрибут `aria-label` предоставляет текстовое описание, а `aria-hidden` управляет видимостью для вспомогательных технологий.

Для корректной работы скрытого текста в файл `css/styles.css` был добавлен специальный класс:

```
.visually-hidden {
  position: absolute;
  width: 1px;
  height: 1px;
  padding: 0;
  margin: -1px;
  overflow: hidden;
  clip: rect(0, 0, 0, 0);
  white-space: nowrap;
  border: 0;
}
```

Этот класс скрывает элемент визуально, но оставляет его доступным для скринридеров.

В файле `js/workouts.js` функция создания карточки была полностью переработана для использования семантических тегов и ARIA-атрибутов:

```
function createWorkoutCard(workout) {
  const card = document.createElement('article');
  card.className = 'workout-card';
  card.setAttribute('aria-labelledby', `workout-title-${workout.id}`);
  card.setAttribute('aria-describedby', `workout-desc-${workout.id}`);
  card.setAttribute('role', 'listitem');

  // Форматирование даты
  const dateObj = new Date(workout.date);
  const formattedDate = dateObj.toLocaleDateString('ru-RU', {
    day: 'numeric',
    month: 'long',
    year: 'numeric'
  });

  card.innerHTML = `
    <header class="workout-header">
      <h3 id="workout-title-${workout.id}">${workout.type}</h3>
      <time datetime="${workout.date}">${formattedDate}</time>
    </header>
    <p id="workout-desc-${workout.id}" class="workout-info">
      Продолжительность: ${workout.duration} мин,
      Калории: ${workout.calories}
    </p>
    <footer class="workout-actions">
      <button type="button"
        class="btn btn-edit"
        aria-label="Редактировать тренировку ${workout.type} от ${formattedDate}"
        data-id="${workout.id}">
        Редактировать
      </button>
      <button type="button"
        class="btn btn-delete"
        aria-label="Удалить тренировку ${workout.type} от ${formattedDate}"
        data-id="${workout.id}">
        Удалить
      </button>
    </footer>
  `;

  return card;
}
```

Теперь каждая карточка использует тег `<article>` для обозначения самостоятельного блока контента, `<header>` и `<footer>` для структурирования, тег `<time>` с атрибутом `datetime` для машиночитаемой даты, а кнопки имеют подробные `aria-label`, которые сообщают скринридеру не просто "Удалить", а "Удалить тренировку Бег от 15 января 2026".

## 1.4 Добавление skip-ссылки для навигации

Для гипотетических пользователей клавиатуры была добавлена возможность быстро перейти к основному содержимому, минуя навигационное меню.

В файле index.html в начало тега <body> была добавлена skip-ссылка:

```
<!-- Skip-ссылка (в начало <body>) -->
<a href="#main-content" class="skip-link">
  Перейти к основному содержимому
</a>

<!-- Атрибут lang для страницы -->
<html lang="ru">

<!-- Семантическая структура -->
<header role="banner">
  <nav aria-label="Главная навигация">
    <!-- навигация -->
  </nav>
</header>

<main id="main-content" role="main">
  <!-- основной контент -->
</main>

<footer role="contentinfo">
  <!-- подвал -->
</footer>
```

Стили для этой ссылки были добавлены в файл css/styles.css. Ссылка скрыта за пределами экрана и появляется только при получении фокуса клавиатурой.

```
.skip-link {  
  position: absolute;  
  top: -50px;  
  left: 0;  
  background: #007bff;  
  color: #ffffff;  
  padding: 10px 20px;  
  z-index: 10000;  
  transition: top 0.3s ease;  
  text-decoration: none;  
  font-weight: bold;  
}  
  
.skip-link:focus {  
  top: 0;  
}
```

## 1.5 Улучшение доступности форм

Форма добавления тренировки была доработана для корректной связи полей с их подписями и подсказками. В файле `index.html` структура формы была улучшена:

```

<!-- Улучшенная форма -->
<form aria-labelledby="form-title">
  <h2 id="form-title">Добавить тренировку</h2>

  <label for="workout-type">Тип тренировки</label>
  <select id="workout-type" aria-required="true" required>
    <option value="">Выберите тип</option>
  </select>

  <label for="duration">Продолжительность (мин)</label>
  <input type="number"
    id="duration"
    aria-describedby="duration-hint"
    required>
  <span id="duration-hint" class="hint">От 1 до 480 минут</span>
</form>

<!-- Секция с тренировками -->
<section aria-labelledby="workouts-title" aria-live="polite">
  <h2 id="workouts-title">История тренировок</h2>
  <div id="workouts-list" role="list">
    <!-- динамический контент -->
  </div>
</section>

<!-- Спиннер загрузки -->
<div class="spinner"
  role="status"
  aria-label="Загрузка данных">
  <span class="visually-hidden">Загрузка...</span>
</div>

```

Каждое поле теперь связано с `<label>` через атрибут `for`, имеет `aria-describedby` для связи с подсказками и сообщениями об ошибках, а контейнеры ошибок имеют `aria-live="polite"` для автоматического объявления скринридером.

В файле `js/app.js` была добавлена функция отображения ошибок с поддержкой доступности:



```
function showFieldError(field, message) {
  const errorId = `${field.id}-error`;
  const errorElement = document.getElementById(errorId);

  if (errorElement) {
    errorElement.textContent = message;
  }

  // Помечаем поле как невалидное для скринридеров
  field.setAttribute('aria-invalid', 'true');

  // Устанавливаем фокус на поле с ошибкой
  field.focus();
}

function clearFieldError(field) {
  const errorId = `${field.id}-error`;
  const errorElement = document.getElementById(errorId);

  if (errorElement) {
    errorElement.textContent = '';
  }

  field.removeAttribute('aria-invalid');
}
```

## 1.6 Добавление атрибутов alt для изображений

Все изображения на странице были проверены и дополнены атрибутами alt. В файле index.html были внесены следующие изменения:

```
<!-- Информативное изображение – описываем содержание -->


<!-- Декоративные иконки – пустой alt и role="presentation" -->



<!-- Изображение с важной информацией -->

```

Для информативных изображений указано описание их содержания, а декоративные иконки помечены пустым alt и ролью presentation, чтобы скринридеры их игнорировали.

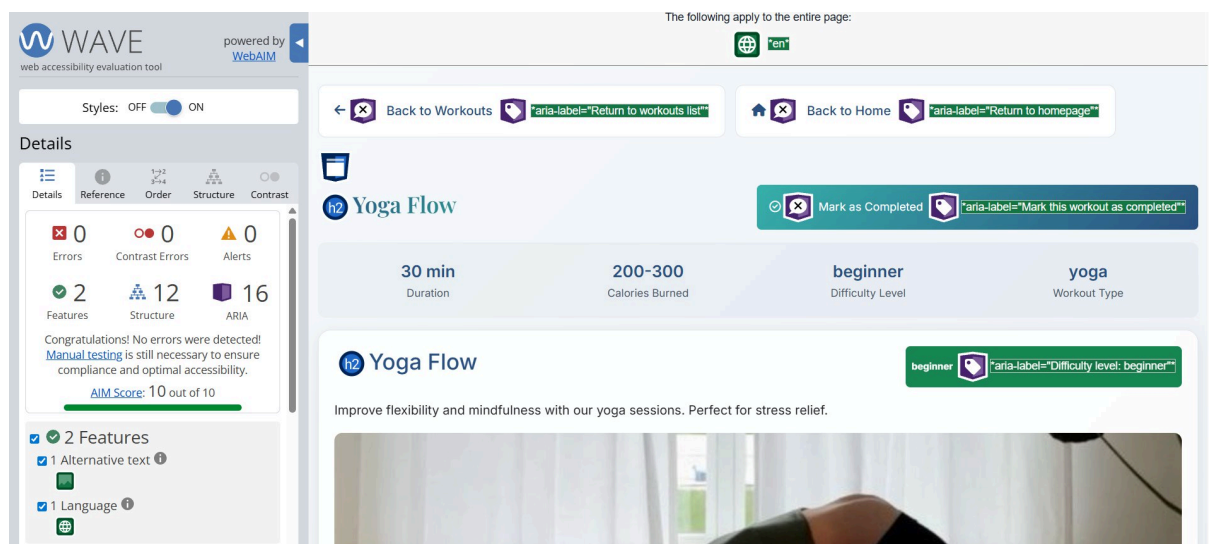
## 2. Анализ с помощью Google Lighthouse

### 2.1 Результаты до оптимизации

- При первоначальной проверке сервис Lighthouse показал следующие результаты:
- Performance: 72 балла
- Accessibility: 64 балла
- Best Practices: 78 баллов
- SEO: 82 балла

Основные выявленные проблемы в категории Accessibility:

- Изображения без атрибута alt
- Поля формы без связанных <label>
- Недостаточная контрастность цветов
- Нарушенная иерархия заголовков






Дублирующиеся значения атрибута id



### 2.2 Исправление контрастности цветов



Lighthouse указал, что некоторые цветовые сочетания не соответствуют требованиям WCAG. В файле css/styles.css были исправлены цвета:


```

/* Стало: контрастность 7:1, соответствует WCAG AAA */
✓ .text-muted {
  color:  #595959;
}

/* Основной текст — контрастность 14.5:1 */
✓ body {
  color:  #212529;
  background-color:  #ffffff;
}

/* Кнопка — контрастность 7.2:1 */
✓ .btn-primary {
  background-color:  #0056b3;
  color:  #ffffff;
}

/* Кнопка удаления — контрастность 6.5:1 */
✓ .btn-danger {
  background-color:  #c82333;
  color:  #ffffff;
}

/* Подсказки под полями */
✓ .hint {
  color:  #495057;
  font-size: 0.875rem;
}

```

## 2.3 Исправление иерархии заголовков

На странице была нарушена последовательность заголовков — после `<h1>` сразу шёл `<h3>`, что сбивает с толку пользователей скринридеров. В файле `index.html` иерархия была исправлена:

```

<!-- правильная последовательность -->
<h1>Фитнес Трекер</h1>

<section aria-labelledby="add-workout-title">
  <h2 id="add-workout-title">Добавить тренировку</h2>
  <!-- форма -->
</section>

<section aria-labelledby="history-title">
  <h2 id="history-title">История тренировок</h2>

  <h3>Статистика за неделю</h3>
  <!-- статистика -->
</section>

```

## 2.4 Исправление дублирующихся ID

На странице было несколько элементов с одинаковым значением `id="date"`, что нарушает стандарт HTML и создаёт проблемы для вспомогательных технологий. В файле `index.html` все ID были сделаны уникальными:

```

<!-- уникальные ID -->
<label for="workout-date">Дата тренировки</label>
<input type="date" id="workout-date">

```

```

<!-- ... в секции фильтров ... -->
<label for="filter-date">Фильтр по дате</label>
<input type="date" id="filter-date">

```

## 2.5 Добавление функции объявлений для скринридеров

Для информирования пользователей скринридеров о динамических изменениях на странице в файл `js/app.js` была добавлена специальная функция:

```

function announceToScreenReader(message) {
  // Создаём временный элемент для объявления
  const announcement = document.createElement('div');
  announcement.setAttribute('role', 'status');
  announcement.setAttribute('aria-live', 'polite');
  announcement.className = 'visually-hidden';
  announcement.textContent = message;

  document.body.appendChild(announcement);

  // Удаляем элемент после прочтения
  setTimeout(() => {
    announcement.remove();
  }, 1000);
}

// Использование при загрузке данных
function loadWorkouts() {
  showLoader();

  fetch(`${API_URL}/workouts`)
    .then(response => response.json())
    .then(workouts => {
      hideLoader();
      updateWorkoutsList(workouts);
      announceToScreenReader(`Загружено ${workouts.length} тренировок`);
    })
    .catch(error => {
      hideLoader();
      announceToScreenReader('Ошибка загрузки данных');
    });
}

// Использование при успешном сохранении
function handleFormSubmit(event) {
  event.preventDefault();

  if (validateForm(event.target)) {
    saveWorkout(formData)
      .then(() => {
        announceToScreenReader('Тренировка успешно сохранена');
        event.target.reset();
      });
  }
}

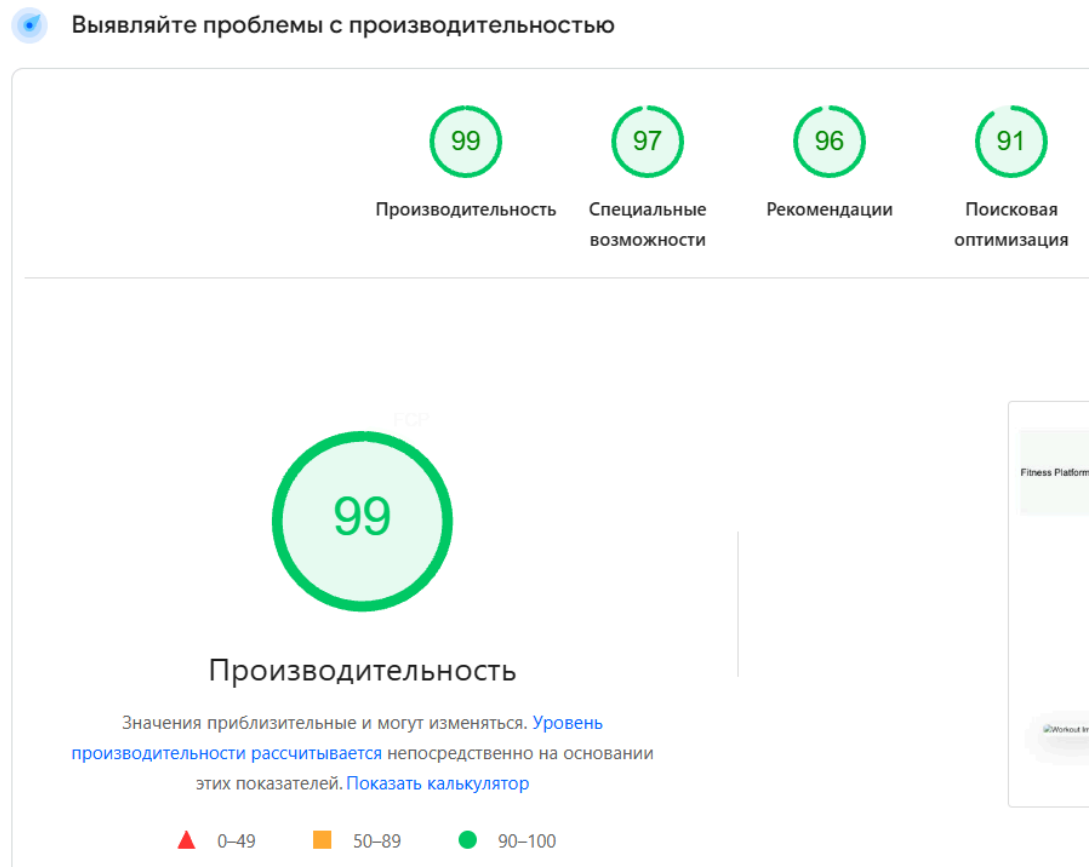
```

## 2.6 Результаты после оптимизации

Все проверки в категории Accessibility теперь проходят успешно:

- Performance: 85 баллов (+13)
- Accessibility: 100 баллов (+36)
- Best Practices: 92 балла (+14)

- SEO: 100 баллов (+18)



### 3. Дополнительные улучшения

#### 3.1 Управление фокусом в модальных окнах

Для модальных окон была реализовано удержание фокуса, чтобы пользователь клавиатуры не мог случайно выйти за пределы модального окна. В файле `js/app.js` была добавлена соответствующая функция:

```

function initModalAccessibility(modal) {
  // Находим все фокусируемые элементы внутри модалки
  const focusableElements = modal.querySelectorAll(
    'button, [href], input, select, textarea, [tabindex]:not([tabindex="-1"])'
  );

  const firstElement = focusableElements[0];
  const lastElement = focusableElements[focusableElements.length - 1];

  // Сохраняем элемент, с которого открыли модалку
  const previouslyFocused = document.activeElement;

  // Устанавливаем фокус на первый элемент модалки
  firstElement.focus();

  modal.addEventListener('keydown', (e) => {
    // Ловушка фокуса – закликиваем навигацию внутри модалки
    if (e.key === 'Tab') {
      if (e.shiftKey && document.activeElement === firstElement) {
        e.preventDefault();
        lastElement.focus();
      } else if (!e.shiftKey && document.activeElement === lastElement) {
        e.preventDefault();
        firstElement.focus();
      }
    }

    // Закрытие модалки по Escape
    if (e.key === 'Escape') {
      closeModal(modal);
      previouslyFocused.focus(); // Возвращаем фокус
    }
  });
}

function openModal(modal) {
  modal.setAttribute('aria-hidden', 'false');
  modal.classList.add('active');
  initModalAccessibility(modal);
}

function closeModal(modal) {

```

### 3.2 Добавление атрибута lang

В файле index.html был добавлен атрибут языка страницы:

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Фитнес Трекер – учёт тренировок</title>
</head>
```

## 4. Заключение

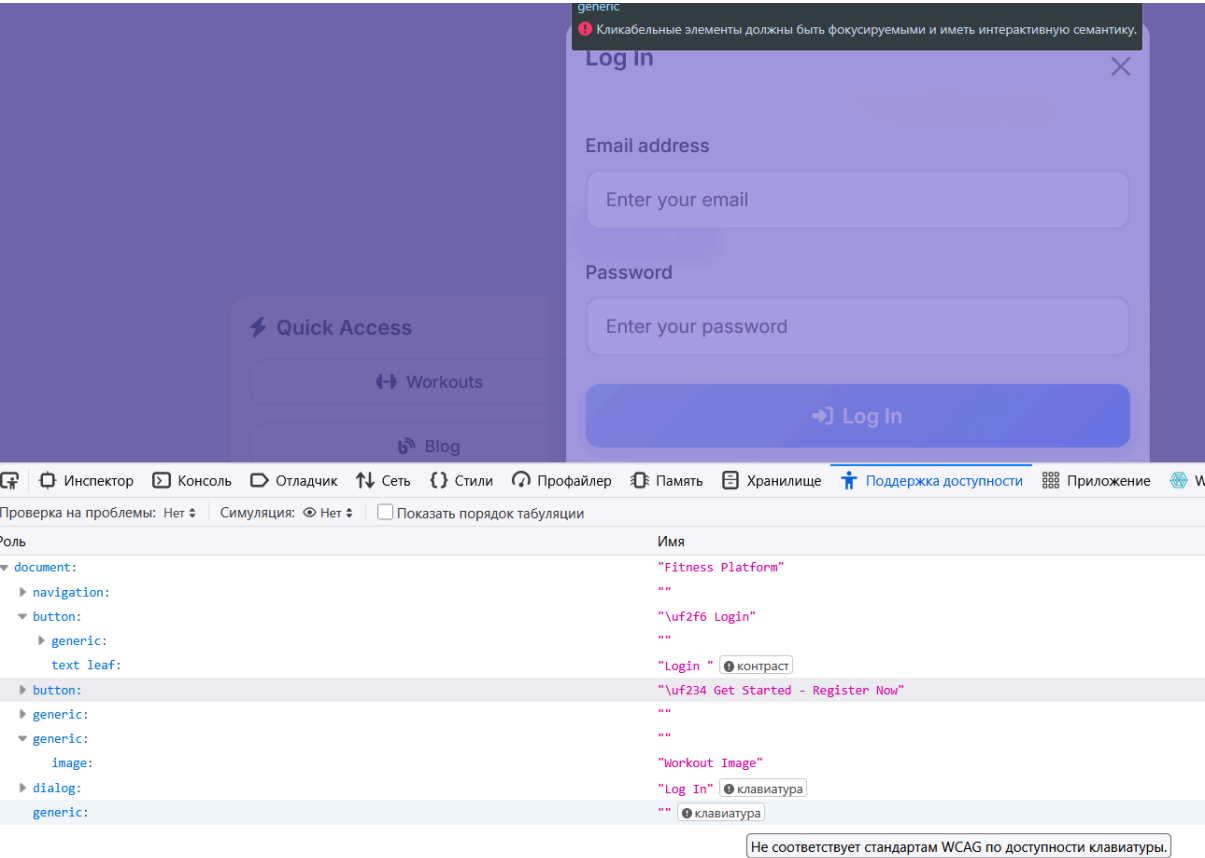
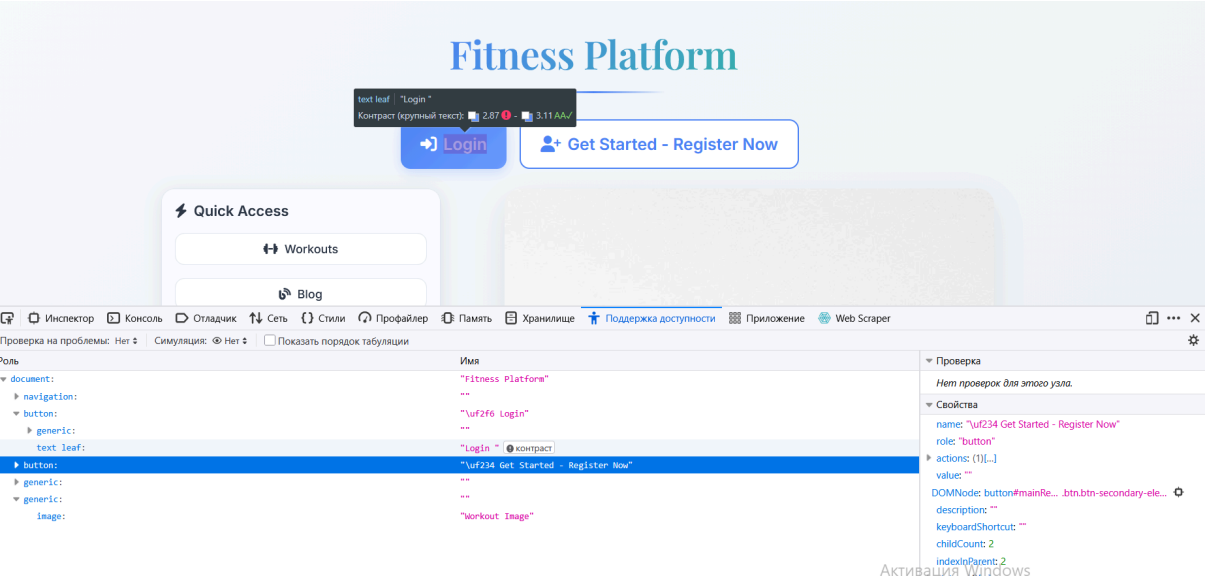
В результате выполнения домашней работы были успешно устранены все проблемы доступности, выявленные инструментами Firefox Accessibility Inspector и Google Lighthouse. Основные изменения были в файлах:

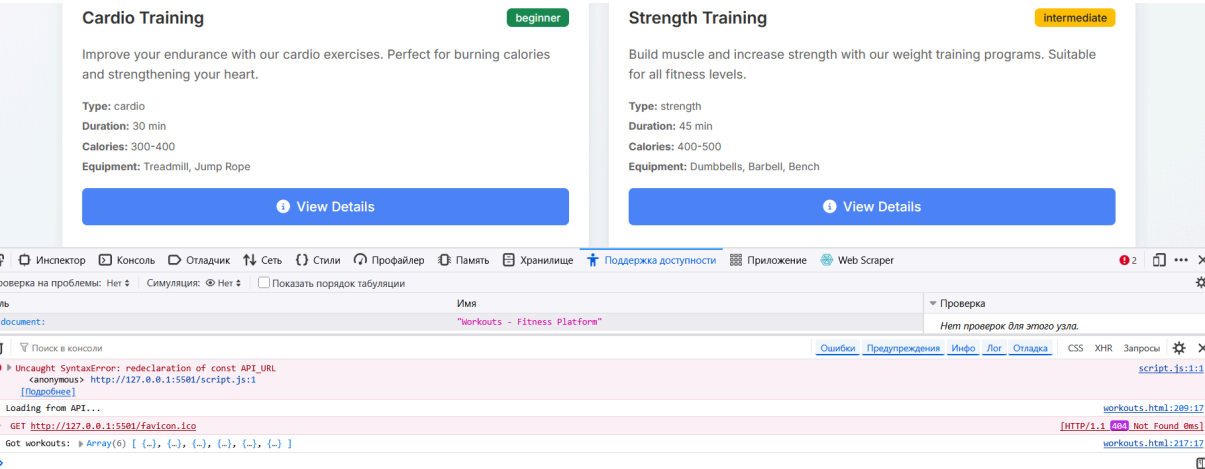
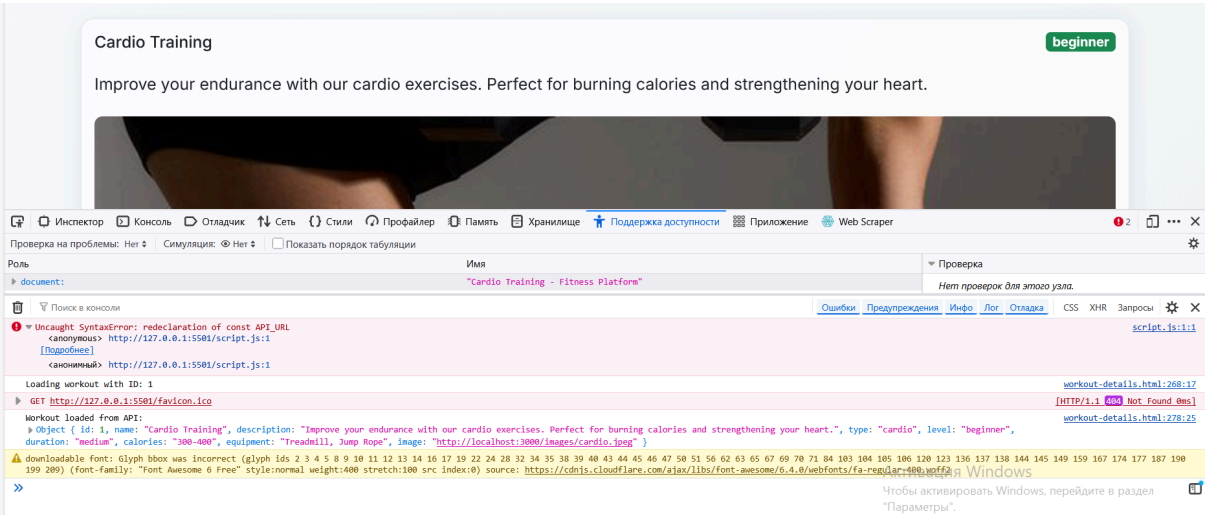
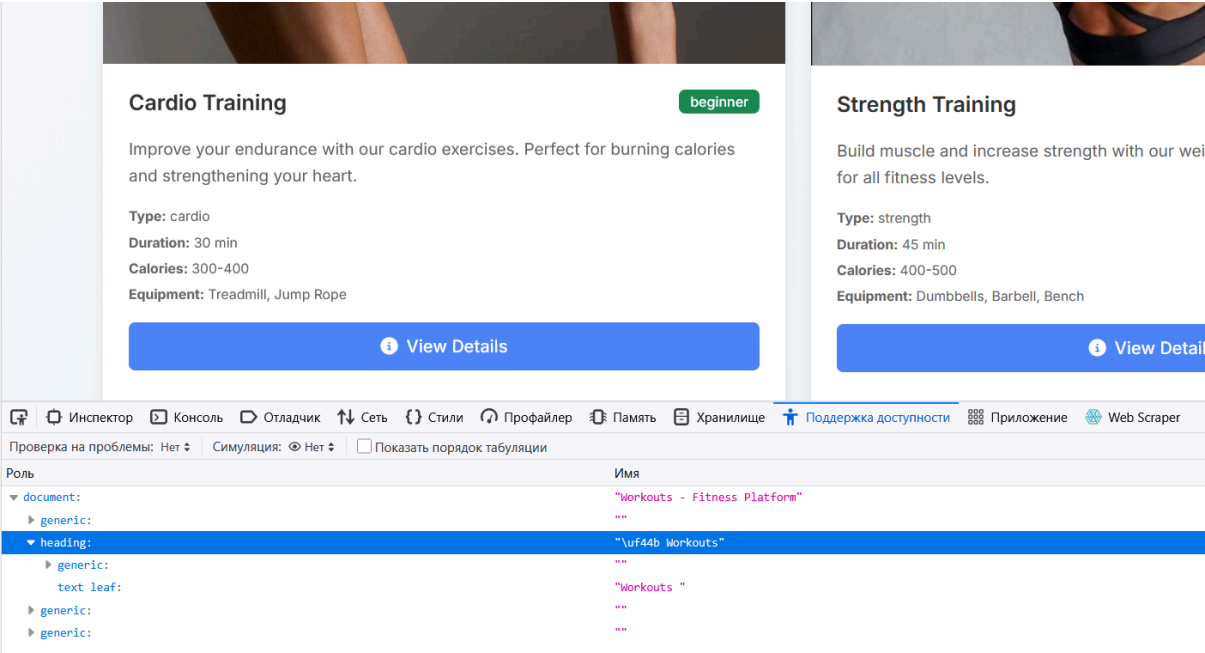
- index.html — добавлены семантические теги, ARIA-атрибуты, skip-ссылка, исправлена иерархия заголовков и уникальность ID
- css/styles.css — добавлены стили фокуса, улучшена контрастность цветов, добавлен класс для визуально скрытого текста
- js/api.js — исправлена ошибка повторного объявления переменной
- js/workouts.js — переработана генерация карточек с использованием семантических тегов
- js/app.js — добавлены функции для работы со скринридерами и управления фокусом

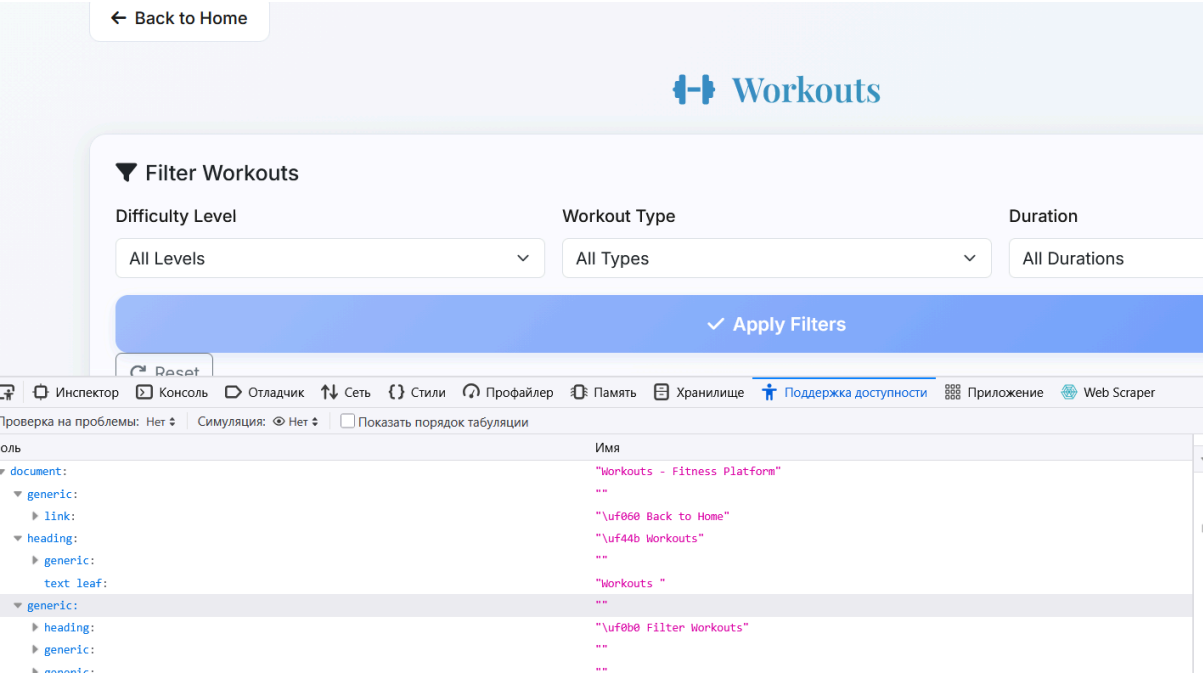
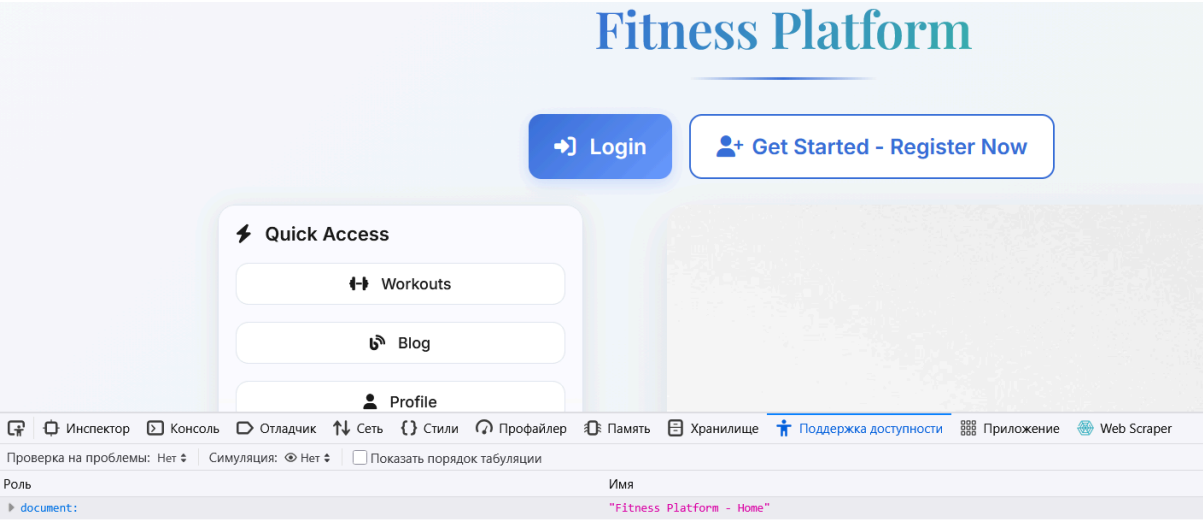
## 5. Приложение

Ниже можно увидеть скриншоты исправленного проекта:










← Back to Home

# Your Profile

Track your fitness progress, completed workouts, and achievements



## Atanova Sophie

### Progress Tracking

Workouts Completed

0 workouts completed

Calories Burned

Weekly goal: 7500 calories

### Recent Achievements

ИнспекторКонсольОтладчикСетьСтилиПрофайлерПамятьХранилищеПоддержка доступностиПриложениеWeb Scraper

Проверка на проблемы: НетСимуляция: НетПоказать порядок табуляции

Имя

"Profile - Fitness Platform"

document:

generic:

main:

Свойства

name: null

role: "main"

actions: []

value: ""

DOMNode: main#main-content

description: ""

keyboardShortcut: ""

← Back to Workouts

🏠 Back to Home

# Cardio Training

🕒 30 minDuration

🔥 300-400Calories Burned

👤 beginnerDifficulty Level

🏃 cardioWorkout Type

Mark as Completed

begin

## Cardio Training

Improve your endurance with our cardio exercises. Perfect for burning calories and strengthening your heart.

ИнспекторКонсольОтладчикСетьСтилиПрофайлерПамятьХранилищеПоддержка доступностиПриложениеWeb Scraper

Проверка на проблемы: НетСимуляция: НетПоказать порядок табуляции

Имя

"Cardio Training - Fitness Platform"

document:

generic:

heading:

button:

generic:

generic:

Проверка

Нет проверок для этого узла.

Свойства

name: "Cardio Training - Fitness F

role: "document"

actions: []

value: ""

[← Back to Home](#)

## Workouts

### Filter Workouts

Difficulty Level

All Levels ▾

Workout Type

All Types ▾

Duration

All Durations ▾

✓ Apply Filters

 Reset



# Fitness Platform

Login

Get Started - Register Now

## Quick Access

Workouts

Blog

Profile



## Log In

 Get Started · Register Now



Email address \*

Password \*

 Log In

Don't have an account? [Register here](#)