ДИСЦИПЛИНА	Фронтенд и бэкенд разработка
ИНСТИТУТ	ИПТИП
КАФЕДРА	Индустриального программирования
ВИД УЧЕБНОГО МАТЕРИАЛА	Методические указания к практическим занятиям
ПРЕПОДАВАТЕЛЬ	Астафьев Рустам Уралович
СЕМЕСТР	1 семестр, 2025/2026 уч. год

Ссылка на материал:
https://github.com/astafiev-rustam/frontend-and-backend-development/tree/practice-1-18

Практическое занятие 18: Использование Lighthouse и анализ проблем доступности

В рамках данного занятия будут использоваться основные подходы к формированию метрик доступности и использованию Lighthouse.

Для восполнения знаний по данной теме рекомендуется повторить материалы лекции. Дополнительно можно ознакомиться с материалом по ссылке:

https://habr.com/ru/companies/htmlacademy/articles/585866/

Примеры

Рассмотрим несколько примеров на обеспечение доступности в веб-приложениях и снятие метрик с использованием Lighthouse

Пример 1: Анализ базовых проблем доступности

Рассмотрим пример использования Lighthouse для выявления базовых проблем доступности на простой странице.

Исходная страница:

```
<!DOCTYPE html>
<html>
<head>
    <title>Мой сайт</title>
    <style>
        .btn { background: blue; color: white; padding: 10px; }
        .text { color: #888; }
        .image { width: 300px; }
    </style>
</head>
<body>
```

```
    <div onclick="alert('Clicked!')" class="btn">Нажми меня</div>
    <p class="text">Важная информация</p>
    <img src="chart.jpg" class="image">
    <input type="text" placeholder="Введите имя">
</body>
</html>
```

Запускаем Lighthouse анализ:

1. Открываем DevTools (F12)
2. Переходим в вкладку Lighthouse
3. Выбираем "Accessibility"
4. Нажимаем "Generate report"

Lighthouse покажет следующие проблемы (может варьироваться):

```
[aria] Элементы с обработчиками клика должны иметь семантическую роль
[color] Контрастность текста недостаточна (2.8:1)
[image] Изображения должны иметь alt атрибуты
[form] Поля ввода должны иметь связанные labels
```

Вот как интерпретировать и исправить эти проблемы:

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <title>Мой сайт - Главная страница</title>
    <style>
        .btn {
            background: #0056b3; /* Увеличили контраст */
            color: white;
            padding: 10px;
            border: none;
            cursor: pointer;
        }
        .text {
            color: #333; /* Увеличили контраст с #888 до #333 */
        }
        .image { width: 300px; }
    </style>
</head>
<body>
    <!-- Было: div с onclick -> Стало: семантическая кнопка -->
    <button class="btn" onclick="alert('Clicked!')">Нажми меня</button>

    <p class="text">Важная информация</p>

    <!-- Было: img без alt -> Стало: с описательным alt -->
    <img src="chart.jpg" class="image" alt="График продаж за 2024 год">
```

```
    <!-- Было: input с placeholder -> Стало: с label -->
    <label for="username">Имя пользователя</label>
    <input type="text" id="username" name="username">
</body>
</html>
```

Таким образом, Lighthouse помог нам выявить критические проблемы доступности, которые мы успешно исправили, сделав страницу более доступной для всех пользователей.

Пример 2: Анализ сложной формы с помощью Lighthouse

Рассмотрим пример глубокого анализа формы обратной связи с помощью Lighthouse.

Исходная страница с формой:

```
<!DOCTYPE html>
<html>
<head>
    <title>Контакты</title>
    <style>
        .form-group { margin: 10px 0; }
        .required { color: red; }
        .error { color: red; font-size: 12px; }
    </style>
</head>
<body>
    <h1>Свяжитесь с нами</h1>

    <form>
        <div class="form-group">
            <span class="required">*</span>
            <span>Имя:</span>
            <input type="text" name="name">
            <div class="error" id="name-error">Поле обязательно</div>
        </div>

        <div class="form-group">
            <span>Email:</span>
            <input type="email" name="email">
        </div>

        <div class="form-group">
            <span>Тема:</span>
            <select name="topic">
                <option>Вопрос</option>
                <option>Жалоба</option>
                <option>Предложение</option>
            </select>
        </div>
```

```
        <div class="form-group">
            <span>Сообщение:</span>
            <textarea name="message"></textarea>
        </div>

        <input type="submit" value="Отправить">
    </form>
</body>
</html>
```

Запускаем расширенный анализ Lighthouse:

1. В Lighthouse выбираем "Desktop"
2. Ставим галочку "Accessibility"
3. Запускаем анализ
4. Изучаем детальный отчет

Lighthouse выявит следующие проблемы (может варьироваться):

```
[label] Элементы формы должны иметь связанные labels
[aria] Обязательные поля должны иметь aria-required
[aria] Сообщения об ошибках должны быть связаны с полями
[heading] Страница должна иметь один заголовок h1
[select] Выпадающие списки должны иметь понятные options
```

Вот комплексное исправление на основе отчета Lighthouse:

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <title>Форма обратной связи - Контакты</title>
    <style>
        .form-group { margin: 15px 0; }
        .required { color: #d63384; }
        .error {
            color: #d63384;
            font-size: 14px;
            margin-top: 5px;
        }
        label { display: block; margin-bottom: 5px; font-weight: bold; }
    </style>
</head>
<body>
    <header role="banner">
        <h1>Свяжитесь с нами</h1>
    </header>

    <main role="main">
```

```
<form aria-labelledby="form-title">
    <!-- Группа поля имени -->
    <div class="form-group">
        <label for="name">
            Имя <span class="required" aria-hidden="true">*</span>
        </label>
        <input type="text"
               id="name"
               name="name"
               required
               aria-required="true"
               aria-describedby="name-error"
               aria-invalid="true">
        <div id="name-error" class="error" role="alert">
            Поле обязательно для заполнения
        </div>
    </div>

    <!-- Группа поля email -->
    <div class="form-group">
        <label for="email">Email адрес</label>
        <input type="email"
               id="email"
               name="email"
               aria-describedby="email-help">
        <div id="email-help" class="help-text">
            Мы отправим ответ на этот адрес
        </div>
    </div>

    <!-- Группа выпадающего списка -->
    <div class="form-group">
        <label for="topic">Тема обращения</label>
        <select id="topic" name="topic">
            <option value="question">Общий вопрос</option>
            <option value="complaint">Жалоба на обслуживание</option>
            <option value="suggestion">Предложение по улучшению</option>
        </select>
    </div>

    <!-- Группа текстовой области -->
    <div class="form-group">
        <label for="message">Ваше сообщение</label>
        <textarea id="message"
                  name="message"
                  aria-describedby="message-help"
                  rows="5"></textarea>
        <div id="message-help" class="help-text">
            Опишите вашу проблему или вопрос подробно
        </div>
    </div>

    <button type="submit" aria-label="Отправить форму обратной связи">
        Отправить сообщение
```

```
            </button>
        </form>
    </main>
</body>
</html>
```

Таким образом, Lighthouse не только показал проблемы, но и помог нам создать полностью доступную форму с правильной семантикой, связанными элементами и понятной структурой для скринридеров.

Пример 3: Анализ и оптимизация сложного интерфейса

Рассмотрим пример анализа сложного интерфейса с модальными окнами и динамическим контентом.

Исходная страница с динамическими элементами:

```
<!DOCTYPE html>
<html>
<head>
    <title>Интернет-магазин</title>
    <style>
        .modal { display: none; position: fixed; top: 50%; left: 50%; transform:
translate(-50%, -50%); background: white; padding: 20px; }
        .overlay { display: none; position: fixed; top: 0; left: 0; width: 100%;
height: 100%; background: rgba(0,0,0,0.5); }
        .card { border: 1px solid #ccc; padding: 10px; margin: 10px; }
        .success { color: green; display: none; }
    </style>
</head>
<body>
    <button onclick="openModal()">Добавить в корзину</button>

    <div class="overlay" onclick="closeModal()"></div>
    <div class="modal" id="modal">
        <h2>Товар добавлен</h2>
        <p>Товар был успешно добавлен в вашу корзину</p>
        <span onclick="closeModal()">✕</span>
    </div>

    <div class="card">
        <img src="product.jpg">
        <h3>Новый смартфон</h3>
        <p>Цена: <span style="color: red;">25 000 руб.</span></p>
    </div>

    <div class="success" id="success">Успешно!</div>

    <script>
        function openModal() {
            document.getElementById('modal').style.display = 'block';
```

```
            document.querySelector('.overlay').style.display = 'block';
            document.getElementById('success').style.display = 'block';
        }
        function closeModal() {
            document.getElementById('modal').style.display = 'none';
            document.querySelector('.overlay').style.display = 'none';
        }
    </script>
</body>
</html>
```

Запускаем Lighthouse с дополнительными опциями (может варьироваться):

1. В Lighthouse выбираем "Mobile"
2. Включаем все категории для комплексного анализа
3. Запускаем анализ
4. Анализируем рекомендации по доступности

Lighthouse выявит сложные проблемы:

```
[aria] Модальные окна должны иметь правильные ARIA атрибуты
[focus] При открытии модалки фокус должен перемещаться внутрь
[keyboard] Модальные окна должны закрываться по Escape
[color] Информация не должна передаваться только цветом
[image] Карточки товаров должны иметь alt тексты
[aria] Динамические уведомления должны использовать aria-live
```

Вот как исправить эти сложные проблемы доступности:

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <title>Интернет-магазин - Каталог товаров</title>
    <style>
        .modal {
            display: none;
            position: fixed;
            top: 50%; left: 50%;
            transform: translate(-50%, -50%);
            background: white;
            padding: 30px;
            border-radius: 8px;
            z-index: 1000;
        }
        .overlay {
            display: none;
            position: fixed;
            top: 0; left: 0;
            width: 100%; height: 100%;
```

```
            background: rgba(0,0,0,0.5);
            z-index: 999;
        }
        .card {
            border: 1px solid #ccc;
            padding: 15px;
            margin: 15px;
            border-radius: 4px;
        }
        .success {
            color: #198754;
            background: #f8f9fa;
            padding: 15px;
            border-radius: 4px;
            display: none;
        }
        /* Стили для фокуса */
        button:focus,
        .close-btn:focus {
            outline: 3px solid #0066cc;
            outline-offset: 2px;
        }
        .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;
        }
    </style>
</head>
<body>
    <header>
        <h1>Каталог товаров</h1>
    </header>

    <main>
        <!-- Карточка товара с доступностью -->
        <article class="card">
            <img src="product.jpg" alt="Смартфон XYZ Model 2024 - черный цвет,
6.7-дюймовый экран">
            <h2>Новый смартфон XYZ</h2>
            <p>
                Цена:
                <strong style="color: #dc3545;">25 000 руб.</strong>
                <span class="visually-hidden">Акционная цена</span>
            </p>

            <!-- Доступная кнопка -->
            <button onclick="openModal()"
```

```
                    aria-label="Добавить смартфон XYZ в корзину покупок">
                Добавить в корзину
            </button>
        </article>
    </main>

    <!-- Доступное модальное окно -->
    <div class="overlay" onclick="closeModal()"></div>
    <div class="modal"
         id="modal"
         role="dialog"
         aria-modal="true"
         aria-labelledby="modal-title"
         aria-describedby="modal-desc"
         tabindex="-1">

        <h2 id="modal-title">Товар добавлен в корзину</h2>
        <p id="modal-desc">Смартфон XYZ был успешно добавлен в вашу корзину
покупок</p>

        <!-- Доступная кнопка закрытия -->
        <button class="close-btn"
                onclick="closeModal()"
                aria-label="Закрыть уведомление">
            ✕ <span class="visually-hidden">Закрыть</span>
        </button>
    </div>

    <!-- Динамическое уведомление с aria-live -->
    <div class="success"
         id="success"
         role="status"
         aria-live="polite"
         aria-atomic="true">
        Товар успешно добавлен в корзину!
    </div>

    <script>
        let previousActiveElement;

        function openModal() {
            // Запоминаем активный элемент
            previousActiveElement = document.activeElement;

            // Показываем модалку
            const modal = document.getElementById('modal');
            modal.style.display = 'block';
            document.querySelector('.overlay').style.display = 'block';

            // Показываем уведомление
            document.getElementById('success').style.display = 'block';

            // Скрываем основной контент от скринридера
            document.querySelectorAll('main > *').forEach(el => {
```

```
                el.setAttribute('aria-hidden', 'true');
            });

            // Фокусируемся на модалке
            modal.focus();

            // Добавляем обработчик Escape
            document.addEventListener('keydown', handleEscape);
        }

        function closeModal() {
            // Скрываем модалку
            document.getElementById('modal').style.display = 'none';
            document.querySelector('.overlay').style.display = 'none';

            // Возвращаем видимость основному контенту
            document.querySelectorAll('[aria-hidden="true"]').forEach(el => {
                el.removeAttribute('aria-hidden');
            });

            // Возвращаем фокус
            if (previousActiveElement) {
                previousActiveElement.focus();
            }

            // Убираем обработчик
            document.removeEventListener('keydown', handleEscape);
        }

        function handleEscape(event) {
            if (event.key === 'Escape') {
                closeModal();
            }
        }

        // Ловим фокус внутри модалки
        document.getElementById('modal').addEventListener('keydown',
function(event) {
            if (event.key === 'Tab') {
                const focusableElements = this.querySelectorAll(
                    'button, [href], input, select, textarea,
[tabindex]:not([tabindex="-1"])'
                );
                const firstElement = focusableElements[0];
                const lastElement = focusableElements[focusableElements.length -
1];

                if (event.shiftKey && document.activeElement === firstElement) {
                    event.preventDefault();
                    lastElement.focus();
                } else if (!event.shiftKey && document.activeElement ===
lastElement) {
                    event.preventDefault();
                    firstElement.focus();
```

```
                }
            }
        });
    </script>
</body>
</html>
```

Таким образом, Lighthouse помог нам проанализировать сложный интерфейс с динамическими элементами и выявить проблемы, которые не очевидны при поверхностном тестировании. Мы создали полностью доступный интерфейс с правильным управлением фокусом, семантикой модальных окон и доступными динамическими уведомлениями.

После обзора примеров можем перейти к самостоятельной работе.

Самостоятельная работа

В рамках самостоятельной работы необходимо:

1. Проанализировать доступность с помощью Lighthouse для страницы контактов и устранить все проблемы доступности, отражённые в отчёте.
2. Провести анализ 5 страниц/сервисов в интернете:
    - сохранить отчёт по каждой странице в формате .pdf и подписать "страницаX.pdf", где X - номер страницы;
    - для каждого отчёта просмотреть, какие конкретно элементы вызывают ошибки;
    - продумать варианты изменения в случае комплексных ошибок.

P.S. По последним двум подпунктам задания фиксация выполнения задания не нужна, то есть в качестве ответа размещаете ссылку на репозиторий, где хранятся исходные файлы проекта, а также добавлены 5 отчётов по исследованным страницам.

Контрольная работа №3

Контрольная работа №3 представляет из себя выполнение набора заданий по практикам 15-18. Срок выставления оценки по контрольной работе №3 - 13 неделя.