**Профессиональное образовательное частное учреждение**

**«Московский кооперативный техникум им. Г. Н. Альтшуля»**

**КУРСОВАЯ РАБОТА**

ПМ.09 Проектирование, разработка и оптимизация веб-приложений МДК 09.01

Проектирование и разработка веб-приложений

Тема: Разработка одностраничного лэндинга частной школы

*(название темы)*

**Выполнил Ковальков Егор**

Специальность: 09.02.07 «Информационные системы и программирование»

Курс 4

Группа ИС-41

**Руководитель**

/ /\_\_\_\_\_\_\_\_

(фамилия, инициалы, подпись)

Мытищи

2024

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ

**РАЗДЕЛ 1. ТЕХНИЧЕСКИЙ ПРОЕКТ**

* 1. Описание предметной области
  2. Описание технологии обработки задач
  3. Информационно-логическая модель системы
  4. Характеристика инструментальных средств разработки

**РАЗДЕЛ 2. РАБОЧИЙ ПРОЕКТ**

2.1 Разработка веб-приложений

2.2 Отладка и тестирование веб-приложений

2.3 Публикация веб-приложения

2 .4 Руководство программиста

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

ПРИЛОЖЕНИЯ

### Курсовая работа по теме "Разработка веб-сайта для частной школы"

## СОДЕРЖАНИЕ

* **ВВЕДЕНИЕ**
* **РАЗДЕЛ 1. ТЕХНИЧЕСКИЙ ПРОЕКТ**
  + 1.1 Описание предметной области
  + 1.2 Описание технологии обработки задач
  + 1.3 Информационно-логическая модель системы
  + 1.4 Характеристика инструментальных средств разработки
* **РАЗДЕЛ 2. РАБОЧИЙ ПРОЕКТ**
  + 2.1 Разработка веб-приложений
  + 2.2 Отладка и тестирование веб-приложений
  + 2.3 Публикация веб-приложения
  + 2.4 Руководство программиста
* **ЗАКЛЮЧЕНИЕ**
* **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**
* **ПРИЛОЖЕНИЯ**

## ВВЕДЕНИЕ

Веб-разработка в последние десятилетия стала важной частью цифрового мира, включая создание сайтов и веб-приложений для различных целей. В рамках данной курсовой работы рассматривается разработка веб-сайта для частной школы. Этот сайт предназначен для презентации образовательного учреждения, предоставления информации о программах обучения, а также для получения заявок на консультацию и записей в школу. Разработка такого сайта требует знания как фронтенд технологий (HTML, CSS, JavaScript), так и принципов UX/UI дизайна, а также навыков тестирования и отладки веб-приложений.

Целью работы является создание функционального, привлекательного и удобного сайта для частной школы с использованием современных технологий веб-разработки.

## РАЗДЕЛ 1. ТЕХНИЧЕСКИЙ ПРОЕКТ

### 1.1 Описание предметной области

Предметная область курсовой работы — создание веб-сайта для частной школы. Этот сайт включает несколько ключевых секций: информация о школе, учебные программы, отзывы, контактная форма и возможность онлайн-записи. Веб-сайт будет доступен для родителей и потенциальных учеников, предоставляя полное представление о предлагаемых образовательных услугах.

### 1.2 Описание технологии обработки задач

Для разработки сайта использовались основные веб-технологии:

* **HTML** (HyperText Markup Language) — основной язык разметки для создания структуры веб-страниц.
* **CSS** (Cascading Style Sheets) — используется для оформления визуального представления контента на странице, например, шрифтов, цветов, размещения элементов.
* **JavaScript** — для добавления интерактивности, например, в обработке форм, анимации элементов на странице.

Сайт включает страницу, которая предоставляет информацию о школе, а также формы для обратной связи и записи на консультацию.

### 1.3 Информационно-логическая модель системы

Веб-сайт для частной школы включает несколько компонентов, которые образуют логическую структуру системы:

1. **Главная страница** — содержит приветственное сообщение, информацию о школе, кнопку для записи на консультацию.
2. **О нас** — страница с описанием школы, её миссии и методик обучения.
3. **Программы** — раздел, в котором описаны образовательные программы, предлагаемые школой.
4. **Отзывы** — раздел с отзывами учеников и их родителей.
5. **Контакты** — форма для обратной связи, включающая поля для имени, телефона и сообщения.

Логическая структура предполагает, что пользователи, попав на сайт, могут легко перемещаться между разделами и получать всю необходимую информацию о школе.

### 1.4 Характеристика инструментальных средств разработки

Для разработки сайта использовались следующие инструменты:

1. **Редактор кода**: Visual Studio Code — мощный и удобный редактор с поддержкой множества расширений для HTML, CSS и JavaScript.
2. **Браузер**: Google Chrome — используется для тестирования и отладки кода.
3. **Графический редактор**: Adobe Photoshop или Figma — для создания графических элементов сайта, таких как логотип, иконки и изображения для hero-блока.
4. **Инструменты для тестирования**: BrowserStack, для проверки совместимости сайта с различными браузерами и устройствами.

## РАЗДЕЛ 2. РАБОЧИЙ ПРОЕКТ

### 2.1 Разработка веб-приложений

Веб-приложение для частной школы было разработано с использованием HTML, CSS и JavaScript. Была создана веб-страница предназначенная для предоставления информации о различных аспектах работы школы.

Главной особенностью разработки стало внимание к дизайну и удобству пользовательского интерфейса.

### 2.2 Отладка и тестирование веб-приложений

После разработки сайта был проведен процесс отладки и тестирования. Проблемы, возникшие при тестировании, касались несовместимости некоторых стилей и элементов на старых версиях браузеров, а также мелких ошибок в валидации формы. Все ошибки были устранены.

### 2.3 Публикация веб-приложения

После завершения разработки и тестирования веб-сайт был опубликован в интернете(GitHub)

### 2.4 Руководство программиста

В этом руководстве мы подробно рассмотрим этапы создания веб-страницы для частной школы "Логос". Этот сайт включает в себя основные элементы, такие как шапку, секции с информацией, формы для обратной связи и стили для различных элементов.

### 1. Создание структуры HTML

#### a) Структура документа

HTML-документ начинается с объявления типа документа <!DOCTYPE html>, что указывает браузеру на использование HTML5. Далее идет тег <html> с атрибутом lang="ru", который указывает на русский язык сайта.

<!DOCTYPE html>

<html lang="ru">

#### b) Мета-теги

Внутри тега <head> указываются мета-теги для кодировки и настроек адаптивности:

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

Эти теги обеспечивают правильное отображение текста и поддерживают адаптивность сайта на разных устройствах.

#### c) Заголовок страницы

В теге <title> указывается название страницы, которое будет отображаться в вкладке браузера:

<title>Частная школа Логос</title>

#### d) Вставка стилей

В теге <style> прописаны стили для всех элементов страницы (шрифты, цвета, размеры). Эти стили задают общий вид и дизайн сайта.

<style>

body {

font-family: 'Arial', sans-serif;

background-color: #fafafa;

}

/\* и другие стили \*/

</style>

### 2. Шапка сайта (header)

Шапка сайта включает логотип и меню навигации.

#### a) Логотип

Логотип размещен в контейнере с классом .logo-container. Он загружается через ссылку на изображение:

<div class="logo-container">

<img src="https://img.icons8.com/?size=50&id=1954&format=png" alt="Логотип школы">

</div>

#### b) Название и меню

В шапке также находится название школы и меню навигации:

<h1>Частная школа Логос</h1>

<nav>

<ul>

<li><a href="#about">О нас</a></li>

<li><a href="#programs">Программы</a></li>

<li><a href="#testimonials">Отзывы</a></li>

<li><a href="#contact">Контакты</a></li>

</ul>

</nav>

Меню навигации ведет к различным разделам сайта с использованием якорных ссылок (href="#section").

### 3. Главный экран (Hero Section)

Этот раздел сайта содержит приветственное сообщение и призыв к действию (кнопка).

<section class="hero">

<h2>Лучшее образование для вашего ребенка</h2>

<p>Инновационные методики обучения и высококвалифицированные преподаватели ждут ваших детей.</p>

<a href="#contact" class="btn">Записаться на консультацию</a>

</section>

* background-image задает фон, который будет показывать изображение.
* Кнопка имеет класс .btn, который стилизует её с помощью CSS.

### 4. Информация о школе (About Section)

Этот раздел предоставляет информацию о школе и её особенностях.

<section id="about">

<h2>О нашей школе</h2>

<p>Мы — частная школа, которая предоставляет детям уникальные возможности для развития...</p>

</section>

Каждый раздел имеет уникальный id, который позволяет связывать их с навигационным меню.

### 5. Программы обучения (Programs Section)

В этом разделе представлены различные образовательные программы.

<section id="programs">

<h2>Наши программы</h2>

<div class="programs">

<div class="program">

<h3>Начальная школа</h3>

<p>Учебная программа для младших школьников...</p>

</div>

<!-- Другие программы -->

</div>

</section>

Использованный CSS стиль делает блоки с программами интерактивными: при наведении происходит эффект подъема блока (transform: translateY(-10px)).

### 6. Отзывы (Testimonials Section)

Этот раздел включает в себя несколько отзывов от учеников и родителей.

<section id="testimonials" class="testimonials">

<h2>Отзывы наших учеников и родителей</h2>

<div class="testimonial">

<p>"Здесь ребенок не только учится, но и растет как личность."</p>

<span>- Марина, мама ученика</span>

</div>

<!-- Другие отзывы -->

</section>

Каждый отзыв отображается в отдельной карточке с фоном и тенями для улучшения визуального восприятия.

### 7. Контактная форма (Contact Section)

В этом разделе присутствует форма для записи и отправки заявки на консультацию.

<section id="contact">

<h2>Контакты и форма записи</h2>

<div class="contact-form">

<input type="text" id="name" placeholder="Ваше имя" required>

<input type="tel" id="phone" placeholder="Ваш телефон" required>

<textarea id="message" placeholder="Сообщите, что вас интересует" required></textarea>

<button type="submit" onclick="submitForm()">Отправить заявку</button>

</div>

</section>

Каждое поле ввода (имя, телефон и сообщение) оформлено с помощью стандартных элементов HTML. Кнопка "Отправить заявку" вызывает JavaScript функцию submitForm().

### 8. Подвал сайта (Footer)

Подвал сайта содержит копирайт, контактную информацию и ссылку на email.

<footer>

<p>&copy; 2024 Частная школа Логос. Все права защищены.</p>

<p>Контакты: <a href="mailto:info@school.ru">info@school.ru</a> | Телефон: +7 123 456-78-90</p>

</footer>

### 9. Скрипт для формы (JavaScript)

При отправке формы вызывается функция submitForm(), которая собирает данные из полей и выводит сообщение с благодарностью. Также форма сбрасывается после отправки.

function submitForm() {

const name = document.getElementById("name").value;

const phone = document.getElementById("phone").value;

const message = document.getElementById("message").value;

alert(`Спасибо за заявку, ${name}! Мы свяжемся с вами по телефону: ${phone}`);

document.querySelector(".contact-form").reset();

}

Теория

**HTML** (от английского HyperText Markup Language) — это язык гипертекстовой разметки текста. Он нужен, чтобы размещать на веб-странице элементы: текст, картинки, таблицы и видео.

Когда вы заходите на сайт, браузер подгружает HTML-файл с информацией о структуре и контенте веб-страницы. HTML как бы выстраивает визуальный фундамент сайта, но не «запускает» сайт самостоятельно. Он всего лишь указывает, где располагаются элементы, какой у них будет базовый дизайн, откуда брать стили для элементов и скрипты (обычно их пишут на JavaScript).

HTML состоит из тегов — команд, которые указывают браузеру, как отображать помещённый в них текст. Это и есть элементы веб-страницы. У каждого тега есть имя, которое заключается в угловые скобки < и >.

Теги бывают парные и непарные. Парные состоят из двух тегов — открывающего и закрывающего, а непарные — из одного. При этом непарный тег тоже можно закрывать, но это необязательно.

У каждого тега есть атрибуты. С их помощью можно передавать элементам веб-страницы дополнительные данные: размеры, уникальный id элемента, ссылки на изображения и так далее.

Элемент <img> включает в себя два основных атрибута — src и alt. Первый добавляет ссылку на файл, а второй — подпись к картинке, которая покажется, если файл не загрузится.

Тег <p> — парный, он всегда должен закрываться тегом </p>. Парные теги применяются для разметки блоков — элементов, в которые можно вкладывать другие элементы (в том числе блоки). К ним относятся:

* контейнеры (div);
* абзацы;
* заголовки;
* списки;
* таблицы.

## **Как HTML работает на сайтах**

Давайте разберём, как браузеры отрисовывают веб-страницы с помощью HTML-файлов.

Принцип работы разметки следующий:

* Вы вводите в адресную строку адрес сайта или страницы.
* Браузер отправляет запрос по этому адресу и получает файл в формате HTML.
* Код из полученного файла последовательно преобразуется в визуальные объекты.

При этом каждый браузер руководствуется своими правилам отображения элементов. Раньше разработчикам приходилось создавать отдельные версии сайтов для Internet Explorer, Firefox, Opera и других браузеров. Стоило только не учесть какую-то особенность браузера, и сайт падал на глазах у разочарованных пользователей. К счастью, организация W3C разработала [веб-стандарты](https://skillbox.ru/media/code/vebstandarty-zachem-i-kak-ikh-izuchat/?utm_source=media&utm_medium=link&utm_campaign=all_all_media_links_links_articles_all_all_skillbox) — чтобы сайты отображались более или менее одинаково во всех браузерах.

Но и от обыкновенных ошибок никто не застрахован. Если разработчик по невнимательности не закроет тег или добавит содержимое куда-нибудь не туда, на странице вылезет что-то неожиданное.

## **Что можно и нельзя сделать на HTML**

HTML — это каркас сайта. По нему браузер отрисовывает веб-страницу, наводит «красоту» с помощью CSS и добавляет логику через JavaScript. Например, в HTML-файле можно прописать:

* гиперссылки;
* таблицы;
* изображения;
* блоки;
* абзацы;
* формы;
* заголовки.

Можно даже задавать простой дизайн напрямую в HTML — например, устанавливать цвет и шрифт текста или фоновый цвет блока. Однако профессионалы не рекомендуют так делать — лучше указывать стили в файле CSS. Там-то ваша фантазия ничем не будет ограничена: задавайте любые отступы, размещайте элементы нестандартными способами, играйтесь с прозрачностью, тенями и анимацией. Всё что угодно — но в CSS.

Хорошо, давайте поговорим об основных знаниях JavaScript. Я постараюсь охватить наиболее важные концепции, которые вам понадобятся для начала работы.

**1. Что такое JavaScript?**

* **Язык программирования:** JavaScript (часто сокращают как JS) - это язык программирования, который в основном используется для создания интерактивных веб-сайтов.
* **Клиентская сторона:** Он выполняется в веб-браузере пользователя (клиентская сторона), что позволяет взаимодействовать с HTML и CSS и делать веб-страницы динамичными.
* **Но не только:** Хотя JS в первую очередь ассоциируется с браузерами, он также используется и на серверной стороне (Node.js), в мобильных приложениях (React Native, NativeScript) и много где еще.

**2. Как JavaScript подключается к HTML?**

Есть два основных способа:

* **Встроенный скрипт:** Можно вставить JS-код непосредственно в HTML-документ, используя тег <script>.

<!DOCTYPE html>

<html>

<head>

<title>Мой сайт</title>

</head>

<body>

<p>Привет, мир!</p>

<script>

alert("Привет из JavaScript!");

</script>

</body>

</html>

* **Внешний файл:** Рекомендуемый способ – размещать JS-код в отдельных файлах (.js) и подключать их через атрибут src тега <script>.

<!DOCTYPE html>

<html>

<head>

<title>Мой сайт</title>

</head>

<body>

<p>Привет, мир!</p>

<script src="script.js"></script>

</body>

</html>

И в файле script.js:

alert("Привет из внешнего JavaScript файла!");

**3. Основные концепции JS:**

* **Переменные (Variables):** Контейнеры для хранения данных. Объявляются с помощью let, const или var (в основном лучше использовать let и const):

let name = "Алексей";

const age = 30;

let isStudent = false;

* + let: Переменные, значения которых могут меняться.
  + const: Переменные, значения которых не могут меняться после присвоения (константы).
* **Типы данных (Data Types):**
  + number: Числа (например, 10, 3.14, -5).
  + string: Строки (текст, например, "Привет", 'JavaScript').
  + boolean: Логическое значение (true или false).
  + null: Пустое значение (представляет отсутствие объекта).
  + undefined: Значение переменной, которой еще не присвоено значение.
  + object: Сложные структуры данных, включая объекты, массивы, функции и т.д.
  + symbol (добавлен в ES6): Уникальные и неизменяемые значения, часто используются в качестве ключей объекта.
* **Операторы (Operators):**
  + Арифметические: + (сложение), - (вычитание), \* (умножение), / (деление), % (остаток от деления).
  + Присваивания: =, +=, -=, \*=, /=, %=.
  + Сравнения: == (равно), === (строго равно), != (не равно), !== (строго не равно), >, <, >=, <=.
  + Логические: && (И), || (ИЛИ), ! (НЕ).
* **Условные операторы (Conditional statements):** if, else if, else. Позволяют выполнять код в зависимости от условий.

let age = 20;

if (age >= 18) {

console.log("Совершеннолетний");

} else {

console.log("Несовершеннолетний");

}

* **Циклы (Loops):** for, while, do...while. Позволяют выполнять код несколько раз.

for (let i = 0; i < 5; i++) {

console.log(i);

}

let j = 0;

while (j < 3) {

console.log(j);

j++;

}

* **Функции (Functions):** Блоки кода, которые можно вызывать многократно.

function greet(name) {

console.log("Привет, " + name + "!");

return "Возвращаемое значение"; *// Необязательно*

}

greet("Иван");

let result = greet("Мария");

console.log(result)

* **Объекты (Objects):** Коллекции свойств (ключ-значение).

let person = {

name: "Анна",

age: 25,

city: "Москва"

};

console.log(person.name); *// Доступ к свойству через точку*

console.log(person["age"]); *// Доступ через квадратные скобки*

* **Массивы (Arrays):** Упорядоченные коллекции элементов.

let colors = ["красный", "зеленый", "синий"];

console.log(colors[0]); *// Доступ к элементу по индексу*

colors.push("желтый"); *// Добавить в конец массива*

* **DOM (Document Object Model):** Представление HTML-документа в виде дерева объектов. JavaScript может использовать DOM для доступа и управления элементами веб-страницы.

*// Пример доступа к элементу по ID и изменения его текста*

let heading = document.getElementById("myHeading");

heading.innerHTML = "Новый заголовок";

* **События (Events):** Действия, которые происходят на веб-странице (например, клик, наведение мыши, ввод текста). JavaScript позволяет обрабатывать события и выполнять действия при их возникновении.

let button = document.getElementById("myButton");

button.addEventListener("click", function() {

alert("Кнопка была нажата!");

* });

**4. Практические шаги для начала:**

* **Консоль разработчика:** Откройте консоль разработчика в вашем браузере (обычно F12 или Ctrl+Shift+I или Cmd+Option+I на macOS). Здесь вы можете писать и тестировать JS-код.
* **Редактор кода:** Используйте текстовый редактор или IDE (например, VS Code) для написания JavaScript.
* **Учитесь и практикуйтесь:** Постоянно читайте документацию (MDN Web Docs), смотрите учебники и, самое главное, пишите код.

**5. Что изучать дальше:**

* **ES6+:** Современные возможности JavaScript (стрелочные функции, классы, модули и т.д.).
* **Асинхронность:** Promise, async/await.
* **Работа с DOM:** Более продвинутое манипулирование HTML-элементами.
* **Фреймворки и библиотеки:** React, Angular, Vue.js (для разработки пользовательских интерфейсов) и Node.js (для серверной разработки).

**Советы:**

* Начинайте с малого: не пытайтесь сразу охватить все.
* Разбивайте задачи: Разбейте большие проблемы на более мелкие.
* Гуглите и задавайте вопросы: В процессе обучения это нормально.
* Практика: Чем больше вы будете писать код, тем лучше вы будете его понимать.

Хорошо, давайте поговорим об основах CSS (Cascading Style Sheets). CSS — это язык, который используется для описания внешнего вида и форматирования HTML-документов. Он позволяет вам управлять цветами, шрифтами, размерами, позиционированием элементов и другими визуальными аспектами веб-страниц.

**1. Что такое CSS?**

* **Язык стилей:** CSS не является языком программирования, как JavaScript, а скорее языком стилей, который определяет, как HTML-элементы должны отображаться в браузере.
* **Разделение содержимого и представления:** CSS отделяет представление (стиль) от содержания (HTML), что делает веб-сайты более организованными, удобными для поддержки и модификации.
* **Стилизация:** CSS позволяет задавать стили (цвета, размеры, шрифты, макеты и т.д.) для элементов HTML.

**2. Как CSS подключается к HTML?**

Есть три основных способа подключения CSS к HTML-документу:

**Встроенные стили (Inline Styles):** Стили применяются непосредственно к HTML-элементу через атрибут style.

<p style="color: blue; font-size: 16px;">Это параграф с inline стилями.</p>

* + **Не рекомендуется**, так как усложняет поддержку и противоречит принципу разделения содержимого и представления.

**Внутренние стили (Internal Styles):** CSS-код размещается внутри тега <style> в <head> HTML-документа.

<!DOCTYPE html>

<html>

<head>

<title>Мой сайт</title>

<style>

p {

color: green;

}

</style>

</head>

<body>

<p>Это параграф с внутренними стилями.</p>

</body>

</html>

* + Удобно для небольших проектов, но для больших сайтов лучше использовать внешние стили.

**Внешние стили (External Styles):** CSS-код размещается в отдельных файлах (.css) и подключается к HTML-документу через тег <link> в <head>.

<!DOCTYPE html>

<html>

<head>

<title>Мой сайт</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<p>Это параграф с внешними стилями.</p>

</body>

</html>

* + **Рекомендуемый способ** для больших проектов. CSS-код в style.css:
  + p {
  + color: red;
  + }

**3. Основные концепции CSS:**

* **Селекторы (Selectors):** Определяют, к каким HTML-элементам применяются стили.
  + **Селектор по тегу:** p (выбирает все параграфы), h1 (все заголовки первого уровня).
  + **Селектор по классу:** .my-class (выбирает все элементы с классом my-class).
  + **Селектор по ID:** #my-id (выбирает элемент с ID my-id).
  + **Комбинированные селекторы:** div p (все параграфы внутри div), div.my-class (все div с классом my-class).
  + **Селекторы атрибутов:** [type="text"] (все элементы с атрибутом type="text").
  + **Псевдоклассы:** :hover, :active, :focus.
  + **Псевдоэлементы:** ::before, ::after.
* **Свойства (Properties):** Атрибуты стиля, которые вы хотите изменить (например, color, font-size, margin, padding, background-color).
* **Значения (Values):** Значения, которые вы присваиваете свойствам (например, red, 16px, 10px, #ffffff, center).
* **Правила CSS:** Комбинация селектора, свойства и значения.
* p { */\* Селектор \*/*
* color: blue; */\* Свойство: Значение \*/*
* font-size: 14px; */\* Свойство: Значение \*/*
* }
* **Модель Box Model (блочная модель):** Каждый HTML-элемент представляет собой прямоугольную коробку. Она состоит из:
  + **Содержимого (content):** Фактический текст, изображения, контент.
  + **Внутреннего отступа (padding):** Пространство между содержимым и границей.
  + **Границы (border):** Линия вокруг отступов.
  + **Внешнего отступа (margin):** Пространство между границей и другими элементами.
* **Цвета:**
  + Имена (например, red, blue, green).
  + Шестнадцатеричные коды (например, #ffffff (белый), #000000 (черный), #ff0000 (красный)).
  + RGB (например, rgb(255, 0, 0) (красный)).
  + RGBA (например, rgba(255, 0, 0, 0.5) (красный с прозрачностью 50%)).
  + HSL/HSLA.
* **Размеры:**
  + px (пиксели).
  + em (относительно размера шрифта родительского элемента).
  + rem (относительно размера шрифта корневого элемента).
  + % (проценты).
  + vw (процент от ширины viewport).
  + vh (процент от высоты viewport).
* **Шрифты:**
  + font-family (название шрифта, например, Arial, Times New Roman).
  + font-size (размер шрифта).
  + font-weight (жирность шрифта).
  + font-style (курсив).
  + line-height (межстрочный интервал).
* **Позиционирование:**
  + static (по умолчанию).
  + relative (относительное позиционирование).
  + absolute (абсолютное позиционирование).
  + fixed (фиксированное позиционирование).
  + sticky (липкое позиционирование).
* **Flexbox и Grid:** Мощные инструменты для создания сложных макетов (расположение элементов, выравнивание).

**4. Порядок применения стилей (Каскадирование):**

* **Специфичность:** Чем более специфичен селектор, тем выше его приоритет. Например, id > class > tag.
* **Порядок объявления:** Если у двух правил одинаковая специфичность, то применяется то, которое указано позже.
* **Важность (!important):** Свойство, отмеченное !important, имеет самый высокий приоритет. **Используется редко и с осторожностью**.

**5. Практические шаги для начала:**

* **Редактор кода:** Используйте VS Code, Sublime Text или другой редактор.
* **Браузер:** Проверяйте изменения в браузере.
* **Консоль разработчика:** Используйте инструменты разработчика для просмотра и отладки CSS.
* **Учитесь постепенно:** Начните с основ, пробуйте, экспериментируйте.

**6. Что изучать дальше:**

* **Flexbox и Grid:** Для создания адаптивных и сложных макетов.
* **Медиа-запросы (Media Queries):** Для адаптации сайта к разным размерам экрана (мобильные, планшеты, десктопы).
* **Анимация и переходы:** Для добавления интерактивности и динамики.
* **Препроцессоры CSS:** Sass, Less (упрощают написание и организацию CSS).
* **CSS Frameworks:** Bootstrap, Tailwind CSS (готовые стили и компоненты).

**Дополнительная информация по JavaScript:**

**Область видимости (Scope):**

* + **Глобальная область видимости:** Переменные, объявленные вне функций, доступны отовсюду.
  + **Локальная область видимости:** Переменные, объявленные внутри функций, доступны только внутри этих функций.
  + **Блочная область видимости (let, const):** Переменные, объявленные внутри блока ({}) доступны только внутри этого блока. Это особенно важно для for и if циклов.

let globalVar = "Я глобальная переменная";

function myFunction() {

let localVar = "Я локальная переменная";

console.log(globalVar); *// Доступно*

console.log(localVar); *// Доступно*

if(true){

let blockVar = 'Я блочная переменная'

console.log(blockVar); *// доступно*

}

*//console.log(blockVar) // Ошибка, недоступно*

}

myFunction();

console.log(globalVar); *// Доступно*

*//console.log(localVar) // Ошибка, недоступно*

**Замыкания (Closures):**

* + Функция “запоминает” свое лексическое окружение. Это позволяет внутренним функциям получать доступ к переменным внешней функции, даже если внешняя функция уже завершила своё выполнение.

function outerFunction(x) {

return function innerFunction(y) {

return x + y;

};

}

let add5 = outerFunction(5);

console.log(add5(3)); *// Выведет 8*

**Методы массивов:**

* + map(): Создает новый массив, применяя функцию к каждому элементу старого массива.
  + filter(): Создает новый массив из элементов, которые прошли проверку.
  + reduce(): Сводит массив к одному значению.
  + forEach(): Выполняет функцию для каждого элемента массива (не возвращает новый массив).
  + find(): Возвращает первый элемент, который удовлетворяет условию.
  + findIndex(): Возвращает индекс первого элемента, который удовлетворяет условию.

let numbers = [1, 2, 3, 4, 5];

let doubled = numbers.map(num => num \* 2); *// [2, 4, 6, 8, 10]*

let even = numbers.filter(num => num % 2 === 0); *// [2, 4]*

let sum = numbers.reduce((acc, num) => acc + num, 0); *// 15*

numbers.forEach(num => console.log(num));

**Работа с JSON:**

* + JSON.stringify(): Преобразует объект JavaScript в JSON-строку.
  + JSON.parse(): Преобразует JSON-строку в объект JavaScript.

**Работа с this:**

* + Значение this зависит от того, как вызывается функция.
  + В методах объекта this ссылается на этот объект.
  + В стрелочных функциях this наследуется от контекста родительской функции.

**Асинхронное программирование:**

* + Promise (промисы): Для работы с асинхронными операциями (например, запросы к серверу).
  + async/await: Более удобный синтаксис для работы с промисами.

**Дополнительная информация по CSS:**

1. **Адаптивный дизайн (Responsive design):**
   * Использование медиа-запросов (media queries) для изменения стилей в зависимости от размера экрана.
   * Применение гибких макетов (flexbox, grid) для адаптации контента к разным устройствам.
   * Использование относительных единиц измерения (em, rem, %, vw, vh).
2. **Свойства для работы с текстом:**
   * text-align (выравнивание текста).
   * text-decoration (подчеркивание, зачеркивание).
   * text-transform (регистр текста: uppercase, lowercase, capitalize).
   * letter-spacing (расстояние между буквами).
   * word-spacing (расстояние между словами).
   * white-space (обработка пробелов).
   * text-shadow (тень текста).
3. **Свойства для работы с фоном:**
   * background-image (изображение фона).
   * background-repeat (повторение фона).
   * background-position (позиция фона).
   * background-size (размер фона).
   * background-attachment (поведение фона при прокрутке).
   * background-blend-mode (режим смешивания фона с элементом).
4. **Трансформации и анимации:**
   * transform (вращение, масштабирование, сдвиг, наклон элемента).
   * transition (создание плавных переходов).
   * animation (создание анимаций).
5. **Псевдоэлементы и псевдоклассы:**
   * ::before, ::after: Добавление контента до или после элемента.
   * :hover: Стиль при наведении мыши.
   * :active: Стиль при нажатии (клика).
   * :focus: Стиль при фокусе (например, на текстовом поле).
6. **Свойства для работы с таблицами:**
   * border-collapse (управление границами).
   * border-spacing (расстояние между ячейками).
   * table-layout (алгоритм макета таблицы).
7. **Свойства для работы с формами:**
   * Стилизация текстовых полей, кнопок, чекбоксов и радиокнопок.

## ЗАКЛЮЧЕНИЕ

В рамках данной курсовой работы был разработан функциональный веб-сайт для частной школы, который позволяет эффективно презентовать школу потенциальным ученикам и их родителям. Все ключевые функции, такие как форма записи на консультацию и разделы с информацией о школе, были успешно реализованы.

Разработка такого веб-приложения позволила углубить знания в области веб-технологий, а также развить навыки работы с инструментами для отладки и публикации веб-сайтов.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Григорьев И. Веб-дизайн. Практическое руководство. — М.: Эксмо, 2023.
2. W3Schools Online Web Tutorials.
3. iStock.
4. Skillbox.
5. MDN Web Docs
6. Javascript.info
7. **Node.js**
8. **CSS-Tricks**
9. **Дока**
10. **Stepik**

## ПРИЛОЖЕНИЯ(отдельно)

1. Код веб-сайта.
2. Скриншоты страниц сайта.