**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ АЗЕРБАЙДЖАНСКОЙ РЕСПУБЛИКИ**

**АЗЕРБАЙДЖАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ПРОМЫШЛЕННОСТИ**

**КАФЕДРА: «КОМПЬЮТЕРНАЯ ИНЖЕНЕРИЯ»**

**ПРЕДМЕТ:**

**«ВЕБ-СИСТЕМЫ И ТЕХНОЛОГИИ»**

**КУРСОВАЯ РАБОТА**

**Тема: «Система регистрации студентов»**

**Группа: 680.22**

**Студент: Зейналова Фидан**

**Преподаватель: Халилов Мятляб**

**Баку-2025**

**ОГЛАВЛЕНИЕ**

**1.Введение**

Общее описание проекта

**2. Использованные технологии**

HTML, CSS и JavaScript с объяснением их роли

**3. Структуру сайта**

Перечисление основных страниц

**4. Подробное описание HTML-тегов**

Все теги с объяснениями их назначения

**5. Описание CSS-свойств**

Принципы стилизации и используемые свойства

**6. Описание JavaScript функциональности**

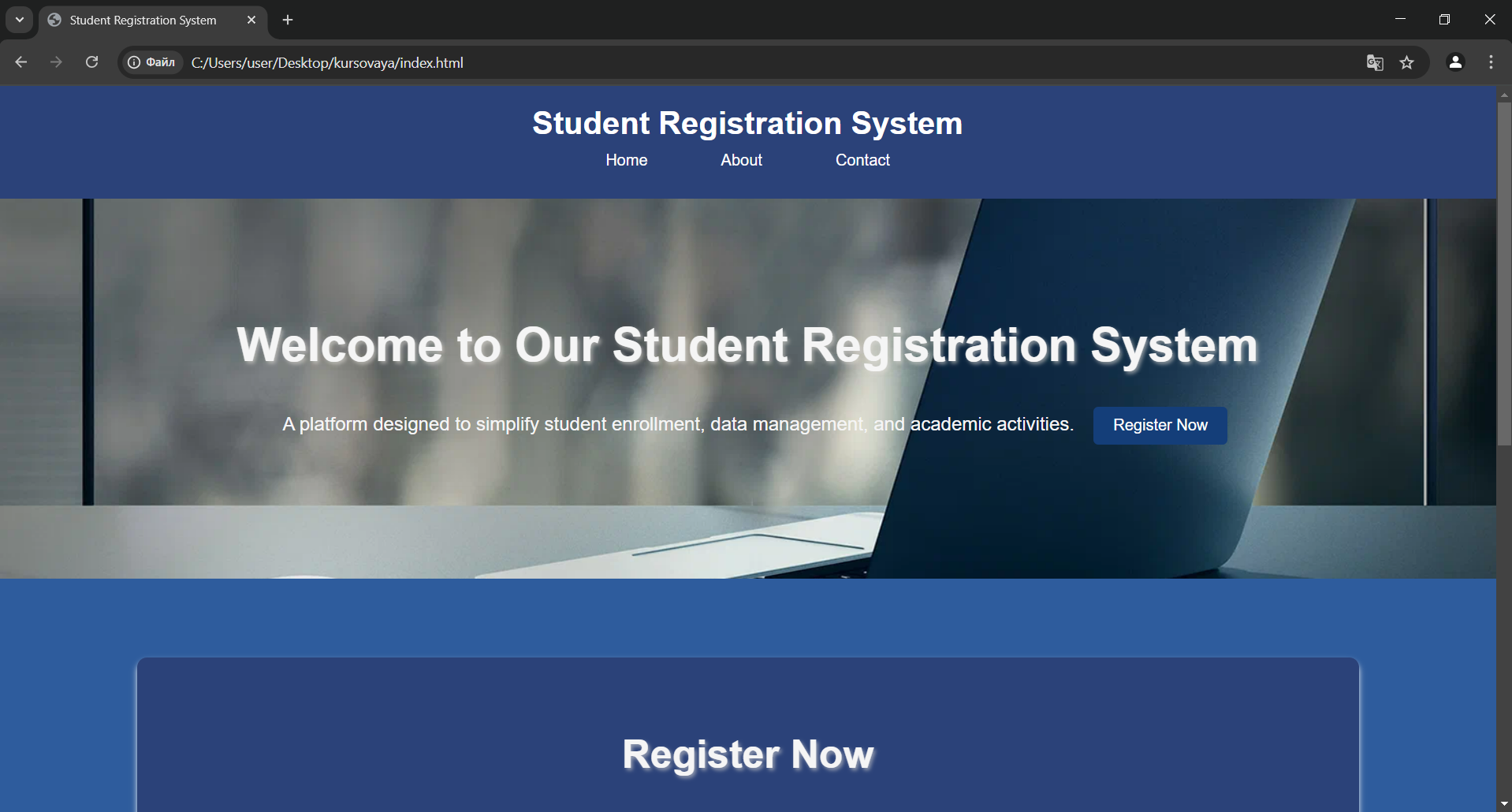
Как работает обработка форм

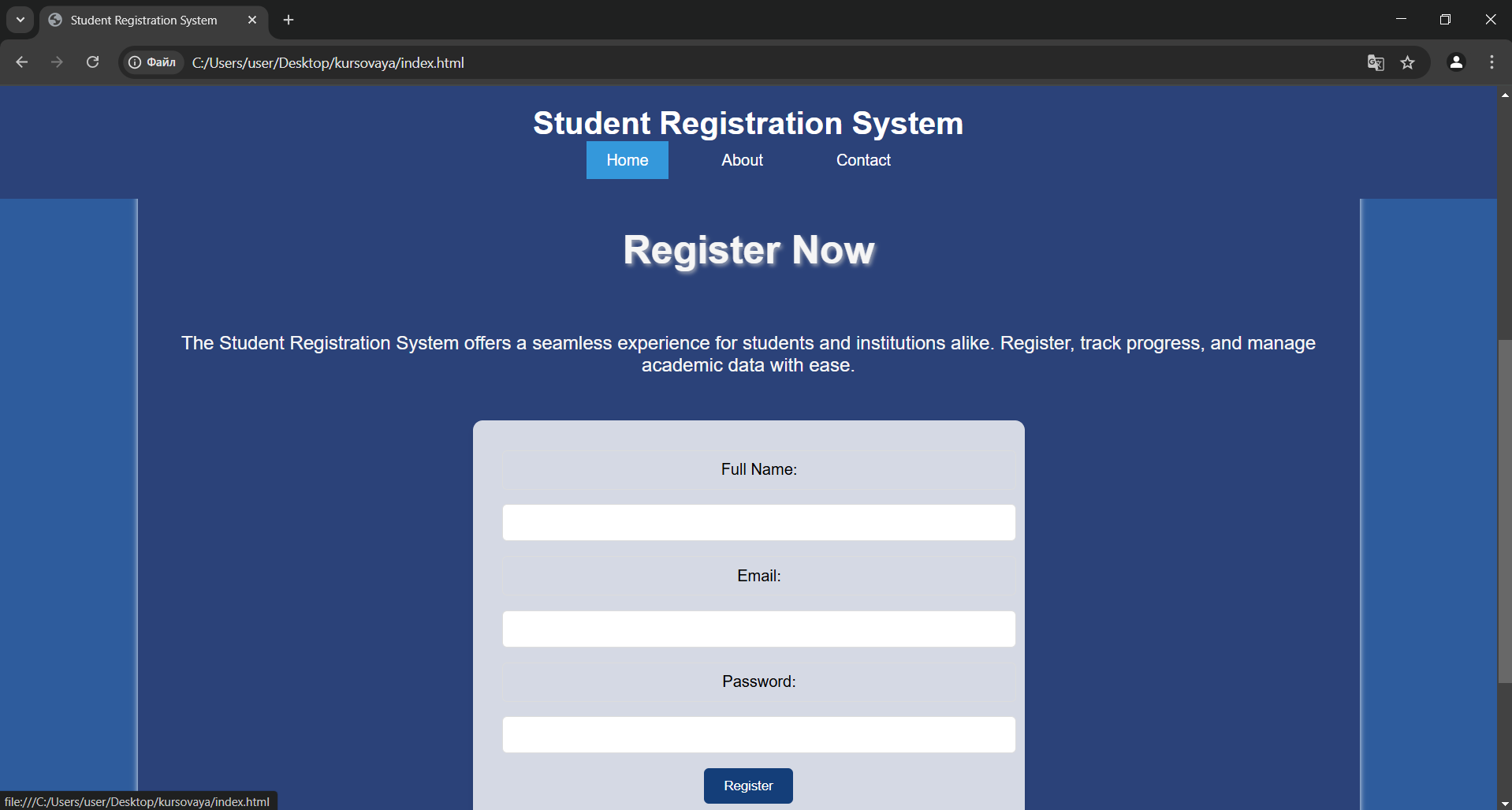
**7.Заключение**

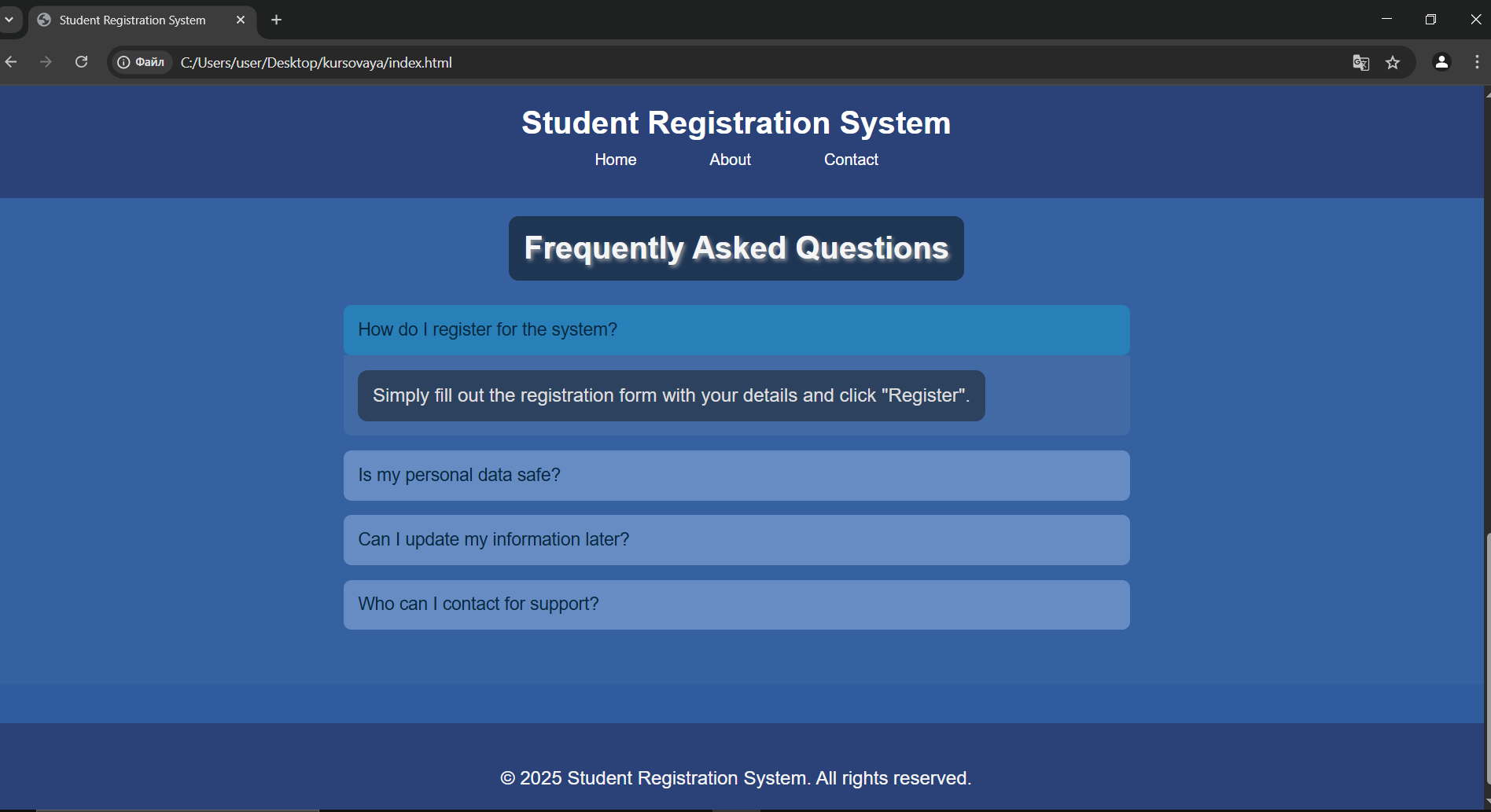
Обобщение проекта

**Веб-сайт: "Система регистрации студентов"**

**Страница: Home**

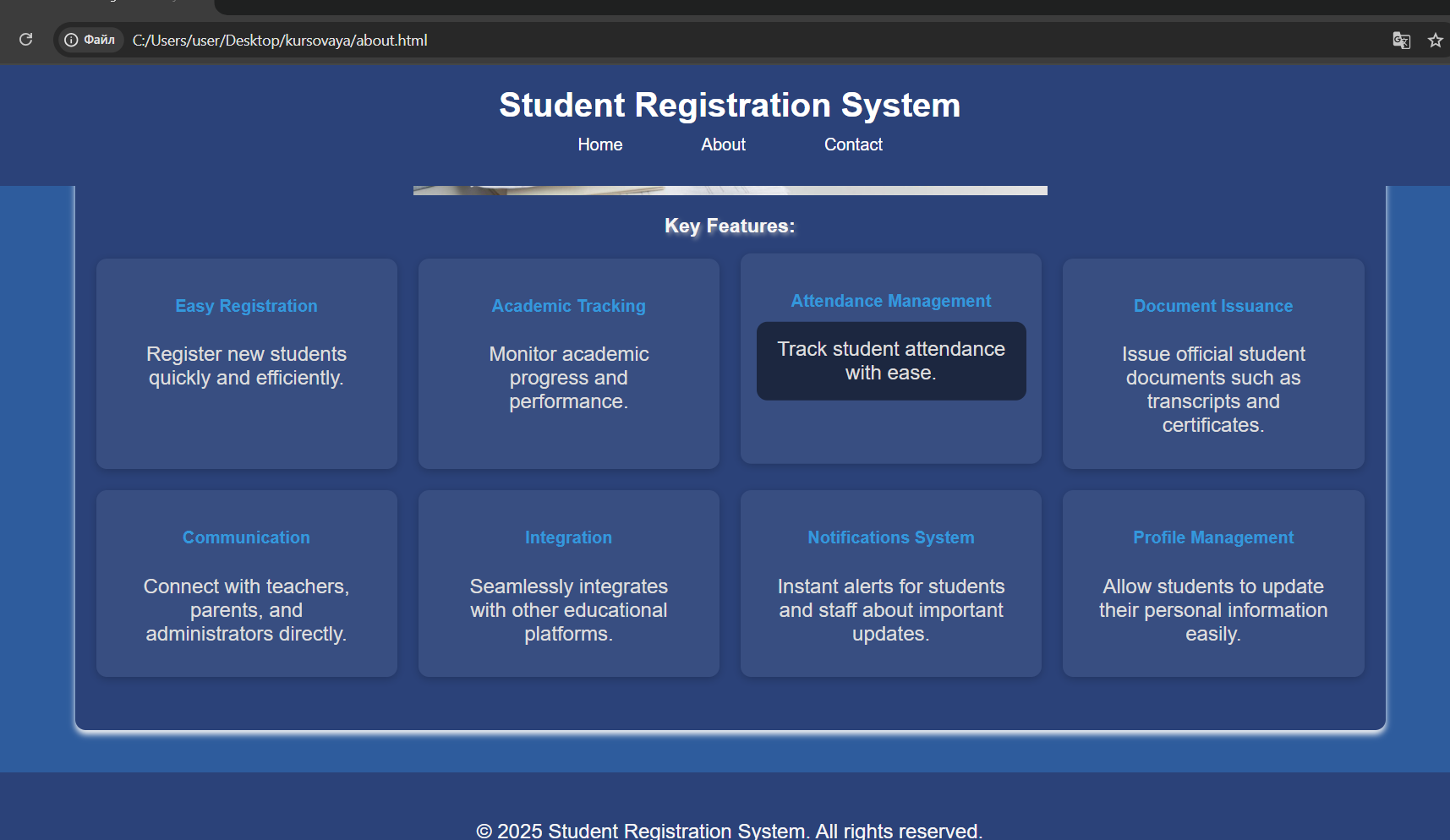




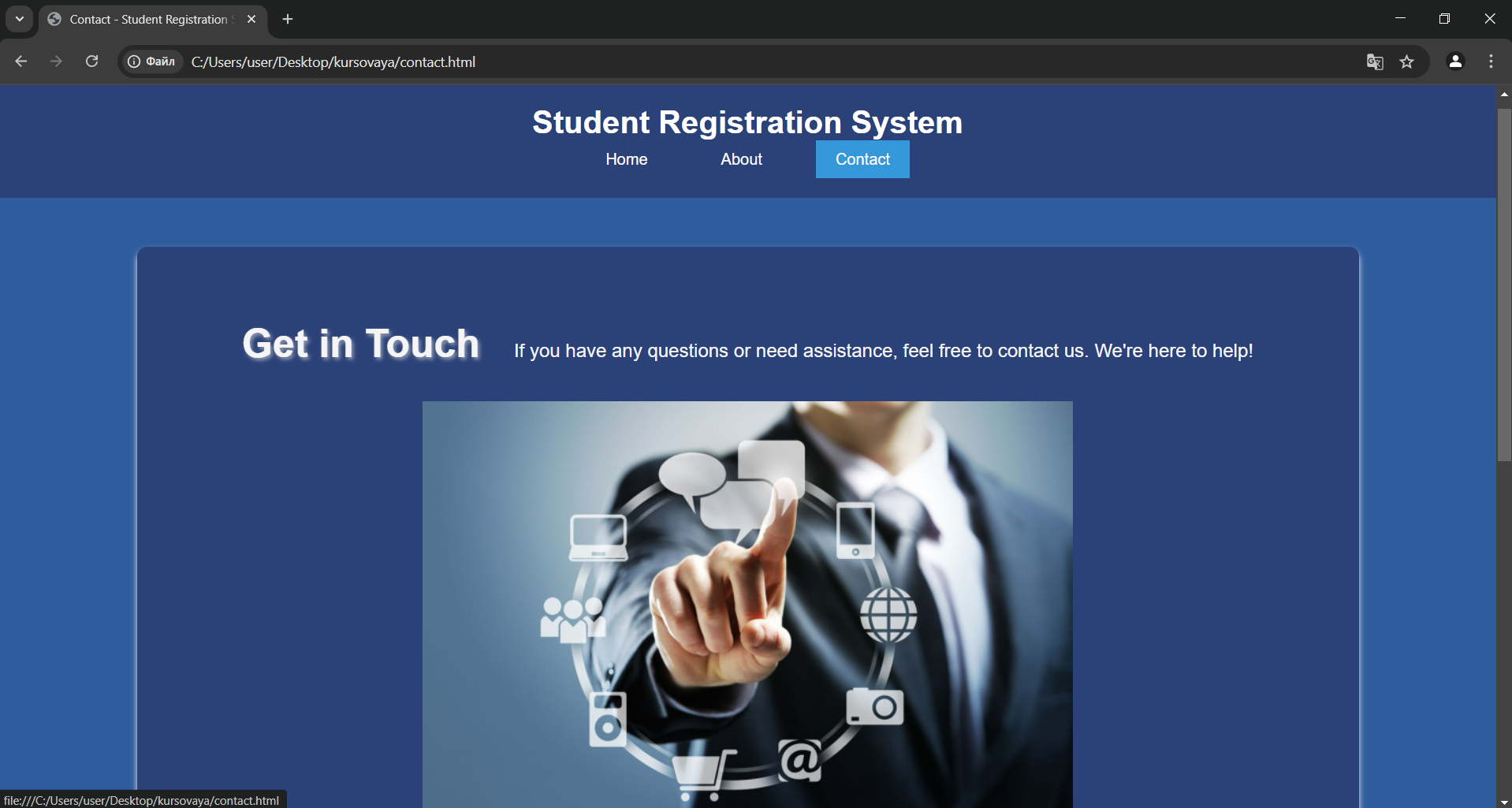


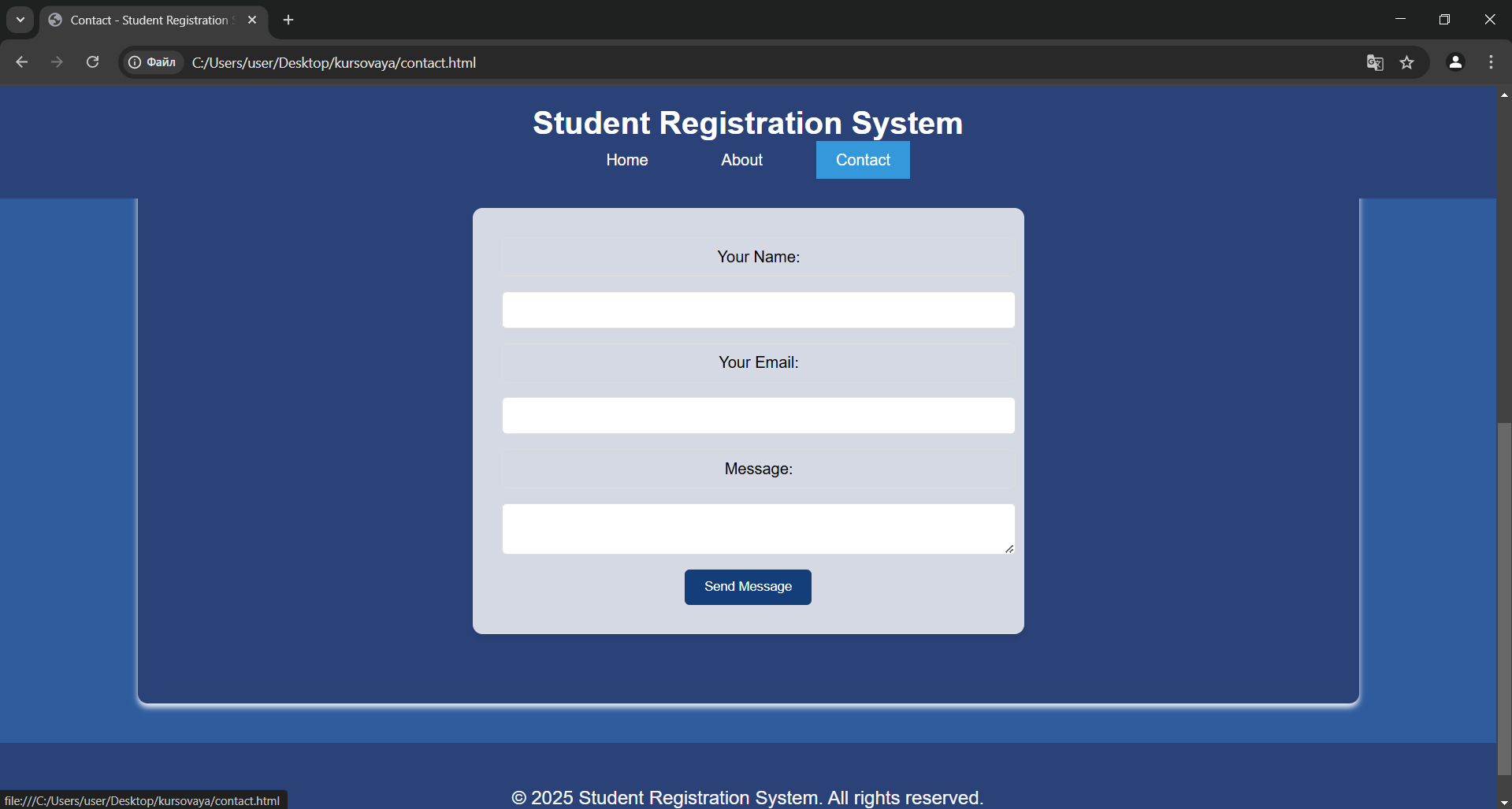
**Страница: About**





**Страница: Contact**





**1. Введение**

Данный проект представляет собой веб-сайт "Система регистрации студентов", разработанный для упрощения процесса регистрации и управления данными студентов в образовательных учреждениях. Сайт предоставляет удобный интерфейс для регистрации новых пользователей, просмотра информации о системе и связи с администраторами.

**2. Использованные технологии**

В ходе разработки проекта были использованы следующие технологии и языки программирования:

**2.1. HTML (HyperText Markup Language)**

HTML используется для создания структуры веб-страниц и размещения на них контента. Это основа любого веб-сайта, определяющая его семантическую структуру.

**2.2. CSS (Cascading Style Sheets)**

CSS используется для стилизации и оформления веб-страниц, что позволяет придать сайту привлекательный внешний вид и обеспечить удобство использования.

**2.3. JavaScript (JS)**

JavaScript применяется для добавления интерактивности на веб-страницы, обработки форм и улучшения пользовательского опыта.

**3. Структура сайта**

Сайт состоит из трех основных страниц:

1. Главная страница (index.html)
2. Страница "О системе" (about.html)
3. Страница "Контакты" (contact.html)

**4. Теоретическая часть: описание использованных HTML-тегов**

**4.1. Структурные теги**

**<!DOCTYPE html>**

Объявление типа документа, указывающее на использование HTML5. Это первая строка в HTML-документе, которая сообщает браузеру, какую версию HTML использовать для отображения страницы.

**<html>**

Корневой элемент HTML-документа, содержащий все остальные элементы. Атрибут lang="en" указывает на английский язык содержимого.

**<head>**

Контейнер для метаданных и служебной информации о странице, которая не отображается на самой странице, но важна для браузеров и поисковых систем.

**<body>**

Контейнер для всего видимого содержимого веб-страницы, включая текст, изображения, ссылки и другие элементы.

**4.2. Метаданные**

**<meta>**

Предоставляет метаданные о HTML-документе:

* charset="UTF-8" - указывает кодировку символов для документа
* name="viewport" - используется для настройки отображения страницы на мобильных устройствах

**<title>**

Определяет заголовок страницы, который отображается в заголовке вкладки браузера и используется поисковыми системами.

**<link>**

Устанавливает связь с внешними ресурсами, например, с CSS-файлами. Атрибут rel="stylesheet" указывает тип связи, а href содержит путь к файлу стилей.

**4.3. Структурные элементы разделов**

**<header>**

Представляет вводную часть страницы, обычно содержит логотип, заголовок сайта и навигационное меню.

**<nav>**

Содержит навигационные ссылки для перемещения между страницами сайта.

**<section>**

Представляет самостоятельный раздел документа, который обычно имеет заголовок.

**<footer>**

Нижняя часть страницы, обычно содержит информацию об авторских правах, ссылки на политику конфиденциальности и другую вспомогательную информацию.

**4.4. Текстовые элементы**

**<h1>, <h2>, <h3>**

Заголовки разных уровней, где <h1> - самый важный заголовок, а <h6> (не используется в моем проекте) - наименее важный.

**<p>**

Определяет параграф или абзац текста.

**<a>**

Создает гиперссылку на другую страницу или ресурс. Атрибут href указывает адрес ссылки.

**<strong>**

Выделяет текст как важный, обычно отображается жирным шрифтом.

**4.5. Списки**

**<ul>**

Создает неупорядоченный (маркированный) список.

**<li>**

Определяет элемент списка внутри <ul> или <ol> (упорядоченный список).

**4.6. Мультимедиа**

**<img>**

Вставляет изображение на страницу. Атрибут src указывает путь к изображению, а alt предоставляет альтернативный текст для тех, кто не может видеть изображение.

**4.7. Формы**

**<form>**

Создает HTML-форму для сбора пользовательского ввода, как правило, для последующей отправки на сервер.

**<label>**

Определяет метку (название) для элемента формы. Атрибут for связывает метку с конкретным элементом формы.

**<input>**

Создает поле ввода в форме. Атрибут type определяет тип поля (текст, электронная почта, пароль и т.д.).

**<textarea>**

Создает многострочное текстовое поле для ввода.

**<button>**

Создает кнопку, чаще всего используется для отправки формы. Атрибут type="submit" указывает, что кнопка отправляет форму.

**4.8. Скрипты**

**<script>**

Используется для вставки или ссылки на исполняемый код JavaScript. Атрибут src указывает путь к внешнему JavaScript-файлу.

**5. Описание CSS-свойств**

**5.1. Селекторы и правила**

В CSS файле используются различные типы селекторов:

* Селекторы тегов (body, header, nav, h1, p и т.д.)
* Селекторы классов (.content-section, .hero, .cta-button)
* Селекторы атрибутов (не используются в моем проекте)
* Псевдоклассы (:hover для эффектов при наведении)
* Псевдоэлементы (::before для создания фона за текстом)

**5.2. Ключевые CSS-свойства**

**Свойства текста**

* font-family - задает шрифт текста
* font-size - определяет размер шрифта
* font-weight - задает толщину шрифта
* color - устанавливает цвет текста
* text-align - определяет выравнивание текста
* text-decoration - используется для добавления или удаления подчеркивания
* text-shadow - добавляет тень к тексту

**Свойства блоков**

* margin - задает внешние отступы элемента
* padding - определяет внутренние отступы элемента
* width, height - устанавливают ширину и высоту элемента
* max-width - задает максимальную ширину элемента
* background-color - устанавливает цвет фона
* border - определяет рамку вокруг элемента
* border-radius - скругляет углы элемента
* box-shadow - добавляет тень к элементу
* box-sizing - определяет, как вычисляется общая ширина и высота элемента

**Свойства позиционирования**

* position - определяет метод позиционирования элемента (fixed, relative, absolute)
* top, left - задают положение элемента относительно его родителя
* z-index - контролирует вертикальное наложение элементов

**Свойства фона**

* background - краткая запись для нескольких свойств фона
* background-image - задает фоновое изображение
* background-size - определяет размер фонового изображения
* background-position - задает положение фонового изображения

**Свойства для гибкого макета**

* display - определяет, как элемент должен отображаться
* flex - создает гибкий контейнер
* justify-content - выравнивает элементы по горизонтали в flex-контейнере
* align-items - выравнивает элементы по вертикали в flex-контейнере

**Свойства для сетки**

• display: grid - устанавливает элемент как грид-контейнер, позволяя распределять дочерние элементы в сетке.

• grid-template-columns - задает количество и размер колонок в сетке (например, repeat(auto-fit, minmax(250px, 1fr)) — автоматически подстраивает количество колонок под размер экрана).

• gap - устанавливает промежуток между элементами сетки.

• grid-column **-** определяет, на каких колонках будет располагаться элемент.

• grid-row - определяет, на каких строках будет располагаться элемент.

• align-items - выравнивает элементы по вертикали внутри сетки.

• justify-items - выравнивает элементы по горизонтали внутри сетки.

**6. Описание JavaScript функциональности**

JavaScript в данном проекте используется для обработки форм. Основная функциональность включает:

1. **Обработка отправки формы:**
   * Перехват события отправки формы
   * Предотвращение стандартного поведения браузера при отправке (предотвращение перезагрузки страницы)
   * Сбор данных из формы
2. **Обработка данных формы:**
   * Создание объекта FormData для сбора введенных пользователем данных
   * Преобразование данных формы в удобный формат объекта JavaScript
3. **Обратная связь пользователю:**
   * Вывод сообщения об успешной отправке формы
   * Сброс формы после отправки

**7. Заключение**

Веб-сайт "Система регистрации студентов" представляет собой полноценное веб-приложение, разработанное с использованием современных веб-технологий (HTML, CSS и JavaScript). Сайт имеет интуитивно понятный интерфейс, адаптивный дизайн и необходимый функционал для регистрации студентов и управления данными.

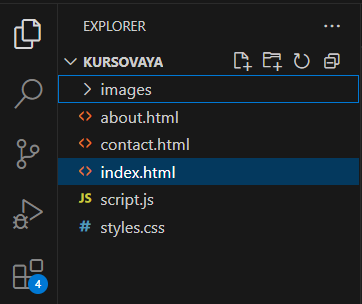
**План практической части проекта "Система регистрации студентов":**

1. Структура проекта
2. Детальный анализ главной страницы (index.html)
3. Детальный анализ страницы "О системе" (about.html)
4. Детальный анализ страницы контактов (contact.html)
5. Детальный анализ CSS-стилей (styles.css)
6. Детальный анализ JavaScript-функциональности (script.js)

**1. Структура проекта**

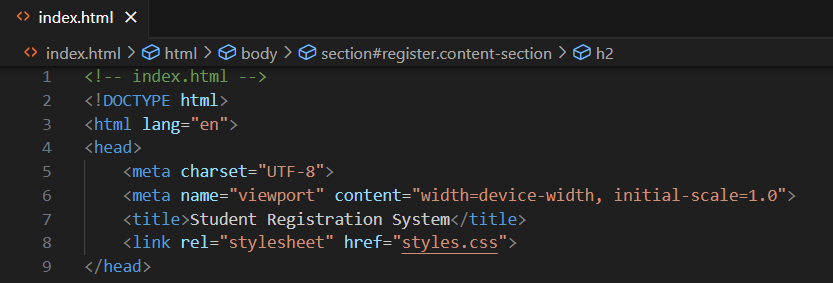
Проект "Система регистрации студентов" включает:

* Три HTML-страницы: главная (index.html), информационная (about.html), контактная (contact.html)
* Один файл стилей (styles.css)
* Один файл JavaScript (script.js)
* Папку с изображениями (images)



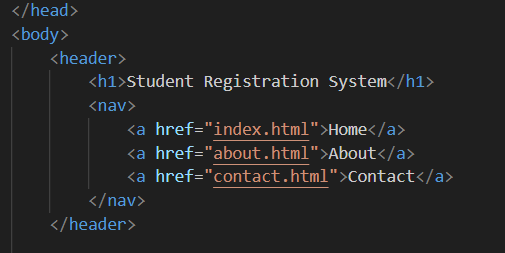
**2. Детальный анализ главной страницы (index.html)**

**Структура HTML-документа:**



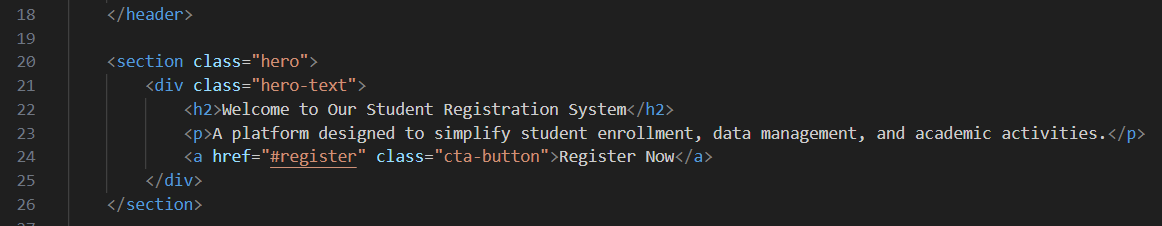
Это основа HTML-документа с указанием языка, кодировки, настройками для мобильных устройств и подключением файла стилей.

**Шапка страницы с навигацией:**



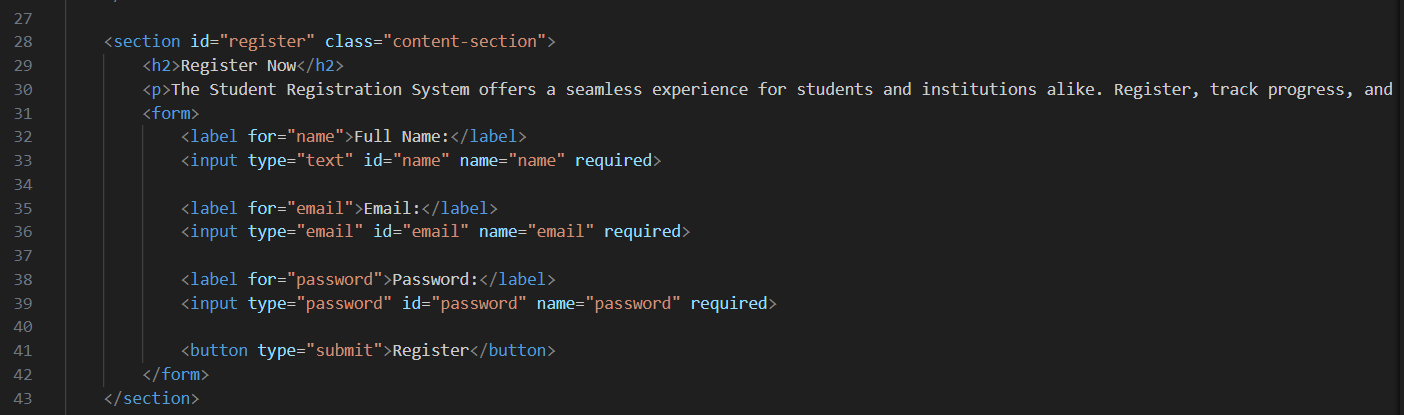
Эта часть создает шапку сайта с заголовком и навигационным меню, которое содержит ссылки на все три страницы сайта.

**Секция-герой:**



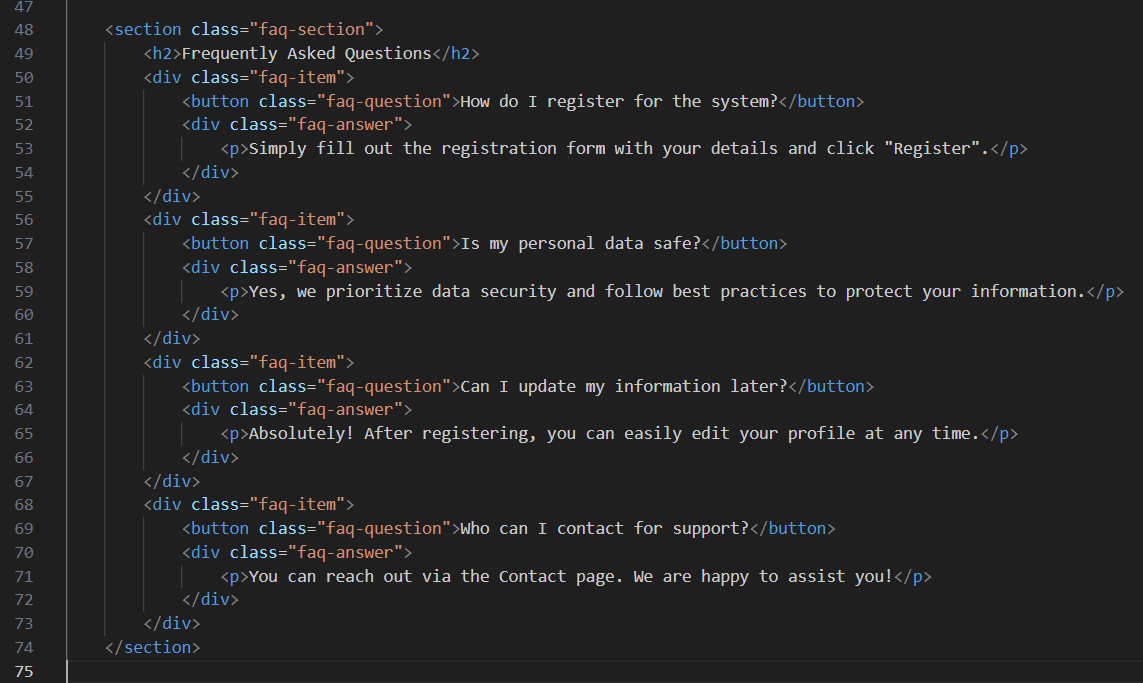
Эта секция создает привлекательный баннер с заголовком, кратким описанием и кнопкой призыва к действию, которая ведет к разделу регистрации ниже на странице.

**Секция регистрации:**



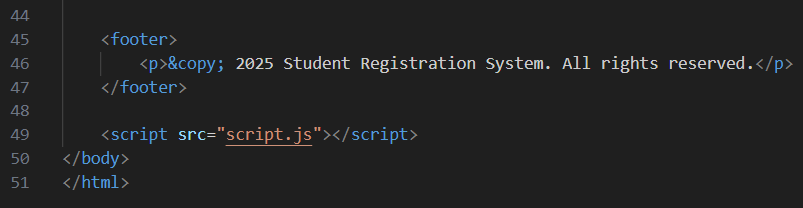
Этот раздел содержит форму регистрации с полями для ввода имени, email и пароля. Атрибут required обеспечивает базовую валидацию, требуя заполнения всех полей перед отправкой.

**Секция FAQ:**



Здесь сделана секция с часто задаваемыми вопросами (FAQ). Каждый вопрос оформлен как кнопка (button), а ответ — как скрытый блок (div).  
Когда пользователь нажимает на вопрос, появляется соответствующий ответ.

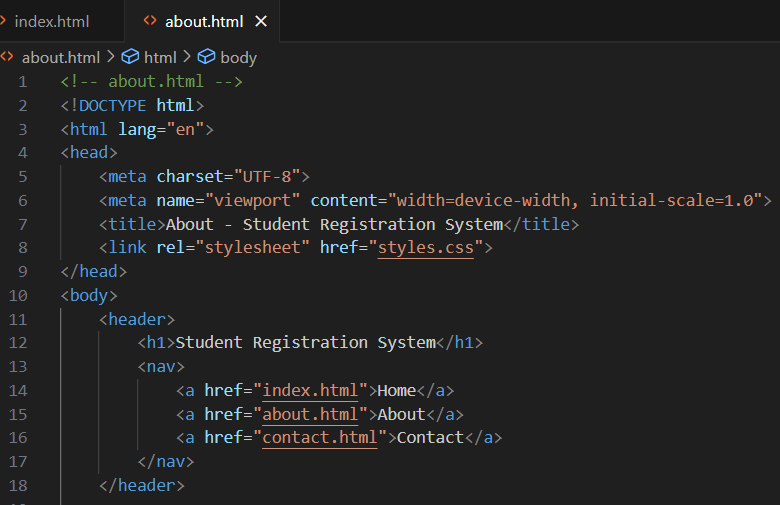
**Подвал и подключение JavaScript:**



Подвал содержит информацию об авторских правах. В конце документа подключается файл JavaScript, который обрабатывает отправку форм.

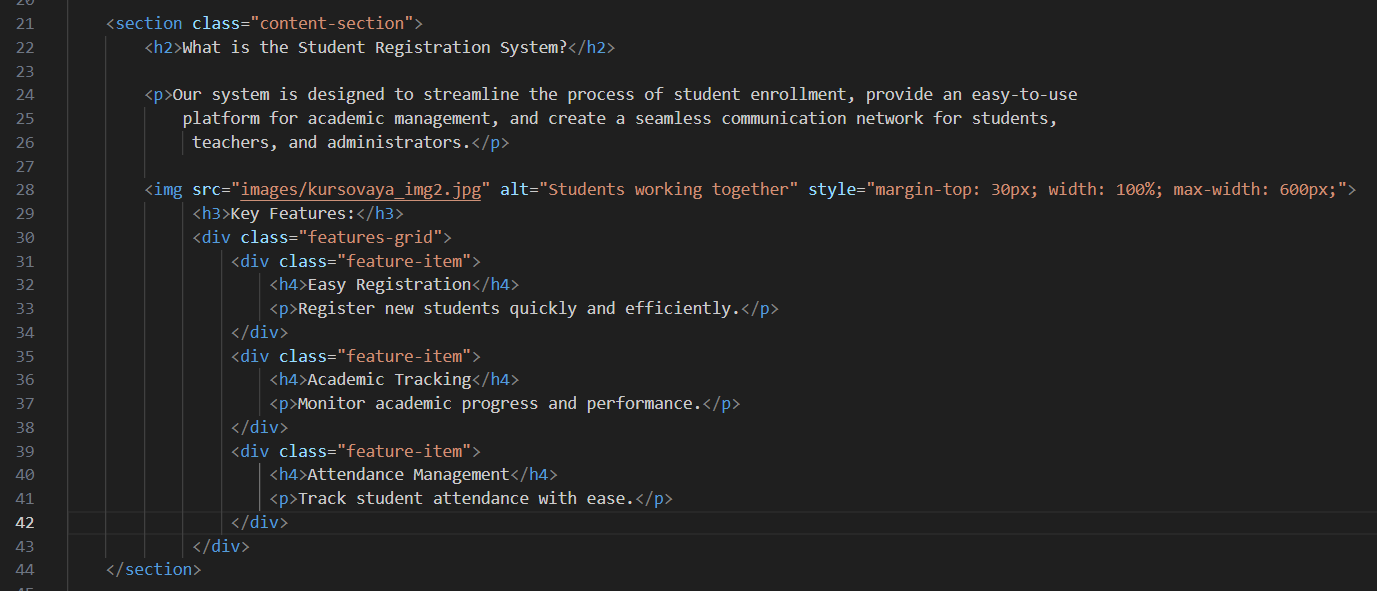
**3. Детальный анализ страницы "О системе" (about.html)**

**Структура HTML-документа и шапка:**



Эта часть идентична структуре главной страницы, обеспечивая единообразие интерфейса, но с измененным заголовком вкладки браузера.

**Основное содержимое:**



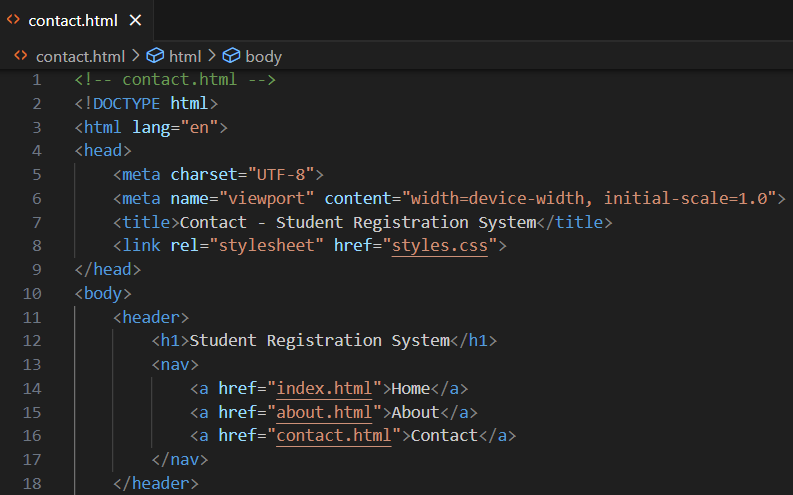
**Секция описания системы регистрации студентов**: короткий текст объясняет, зачем нужна система и какие задачи она решает.  
**Картинка** для визуального усиления и разбавки текста.  
**Блок с "Key Features"** — красиво разложены основные функции системы в виде сетки карточек (8 штук), каждая с заголовком и кратким описанием.

**Подвал:**

Подвал и подключение JavaScript аналогичны главной странице.

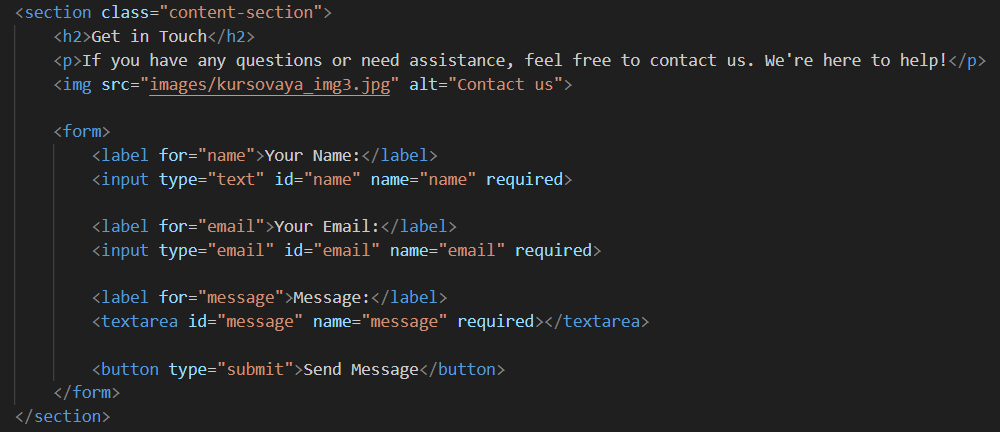
**4. Детальный анализ страницы контактов (contact.html)**

**Структура HTML-документа и шапка:**



Структура аналогична другим страницам, но с соответствующим заголовком страницы.

**Секция контактов:**



Эта секция содержит заголовок, краткий текст, изображение и форму обратной связи с полями для имени, email и текстового сообщения. В отличие от формы регистрации, здесь используется элемент <textarea> для многострочного ввода текста.

**Подвал:**

Подвал и подключение JavaScript идентичны другим страницам.

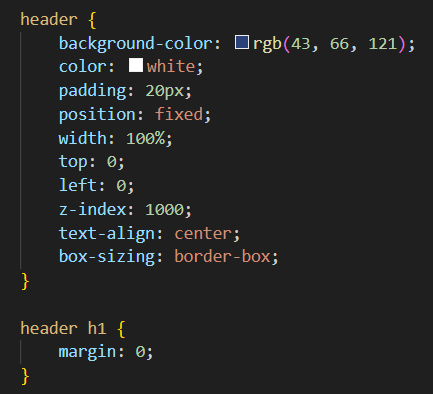
**5. Детальный анализ CSS-стилей (styles.css)**

**Общие стили для тела страницы:**



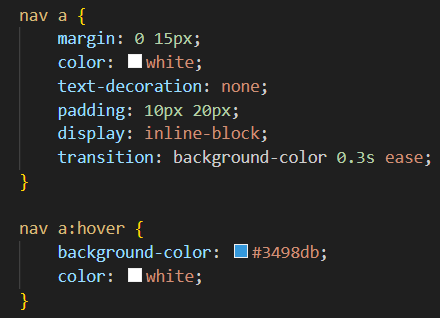
Устанавливает шрифт Arial, удаляет стандартные отступы, задает синий фон и добавляет отступ сверху для размещения фиксированной шапки.

**Стили для шапки:**



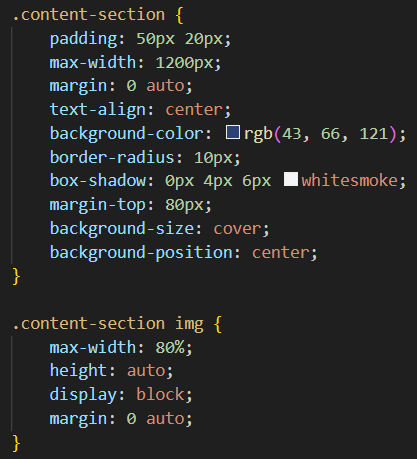
Шапка имеет темно-синий фон, белый текст и фиксированное позиционирование, чтобы оставаться видимой при прокрутке.

**Стили для навигации:**



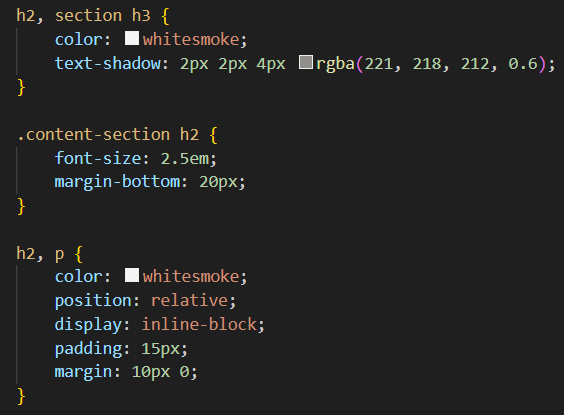
Ссылки навигации имеют отступы, белый цвет и эффект плавного изменения фона при наведении.

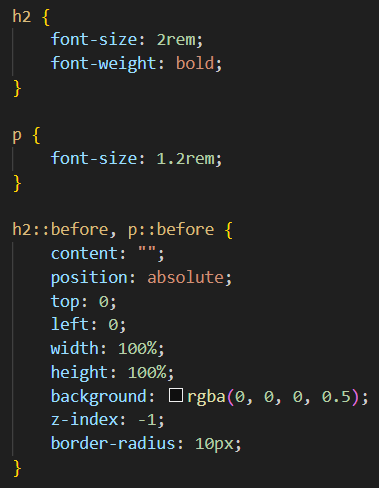
**Стили для контентных секций:**



Контентные секции имеют отступы, ограниченную ширину, выравнивание по центру, темно-синий фон и тень. Изображения ограничены по ширине с сохранением пропорций.

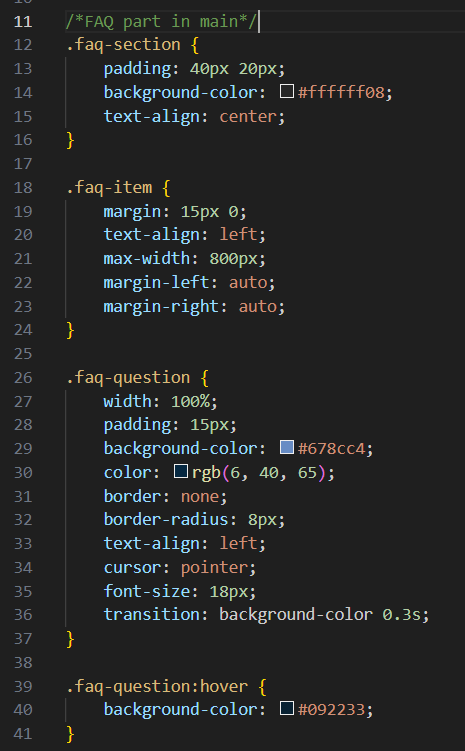
**Стили для текстовых элементов:**

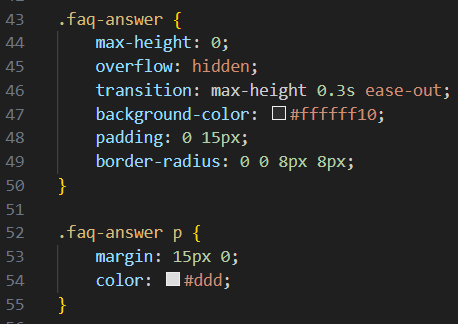




Заголовки и параграфы имеют светлый цвет и темный полупрозрачный фон, создаваемый с помощью псевдоэлемента ::before.

**Стили для FAQ:**

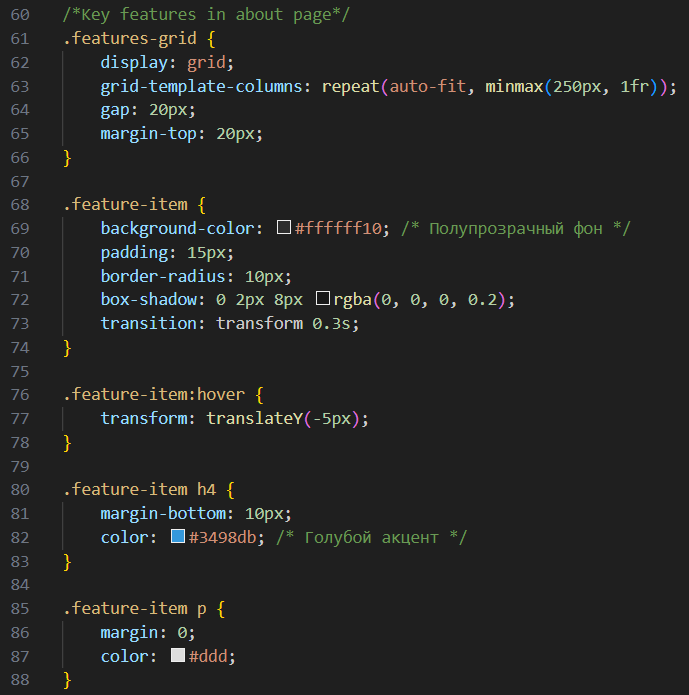




1. **.faq-section:** Добавляет отступы и полупрозрачный фон для всей секции с вопросами.
2. **.faq-item:** Центрирует каждый вопрос и ограничивает его ширину.
3. **.faq-question:** Кнопка с вопросом имеет синий фон, скругленные углы, и плавное изменение цвета при наведении.
4. **.faq-answer:** Ответы скрыты по умолчанию, плавно раскрываются с анимацией при клике, с полупрозрачным фоном.
5. **.faq-answer p:** Текст внутри ответов светло-серый, с отступами.

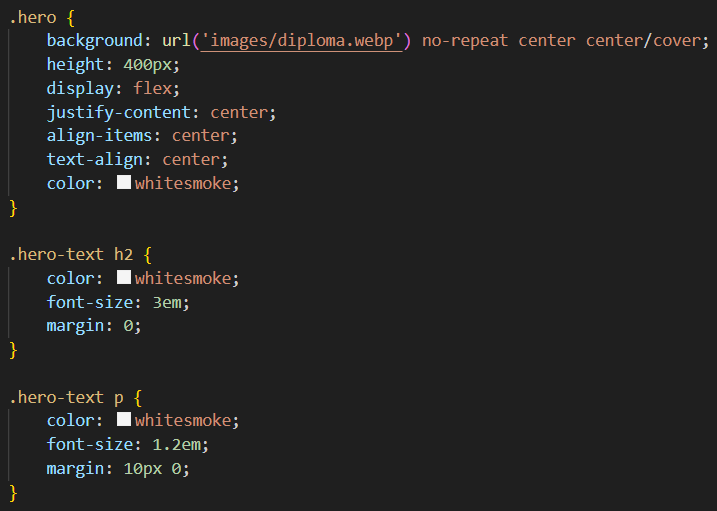
Когда кликаешь на вопрос, его ответ плавно раскрывается, создавая интерактивный и стильный элемент на странице.

**Стили для "Key Features":**



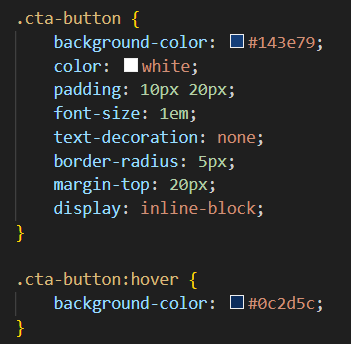
1. **.features-grid**: Создает сетку с элементами, которые автоматически адаптируются по размеру экрана (минимум 250px для каждого элемента) и имеют промежуток 20px между собой.
2. **.feature-item:** Каждый элемент имеет полупрозрачный фон, скругленные углы, небольшую тень для глубины и плавный эффект при наведении, который поднимет элемент вверх на 5px.
3. **.feature-item:hover:** Элемент слегка поднимается при наведении, что создает эффект интерактивности.
4. **.feature-item h4:** Заголовок внутри элемента получает голубой акцент и отступ снизу.
5. **.feature-item p:** Текст внутри элемента имеет светло-серый цвет и нулевой отступ.

**Стили для секции-героя:**



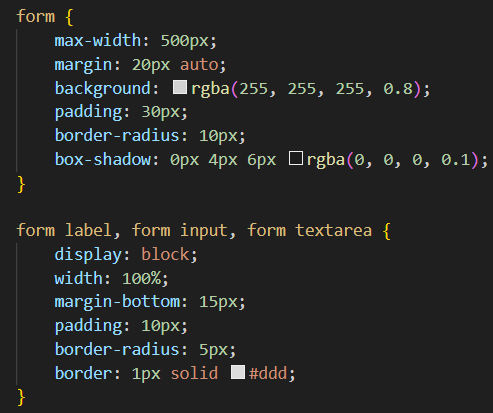
Секция-герой имеет фоновое изображение, фиксированную высоту и центрированное содержимое с использованием flexbox.

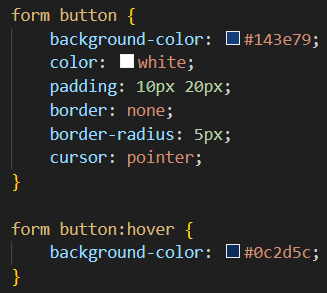
**Стили для кнопок:**



Кнопка призыва к действию имеет синий фон, белый текст, закругленные углы и эффект затемнения при наведении.

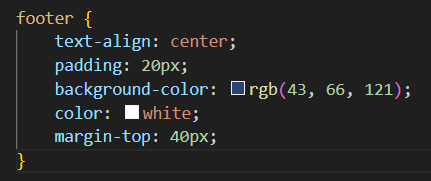
**Стили для форм:**





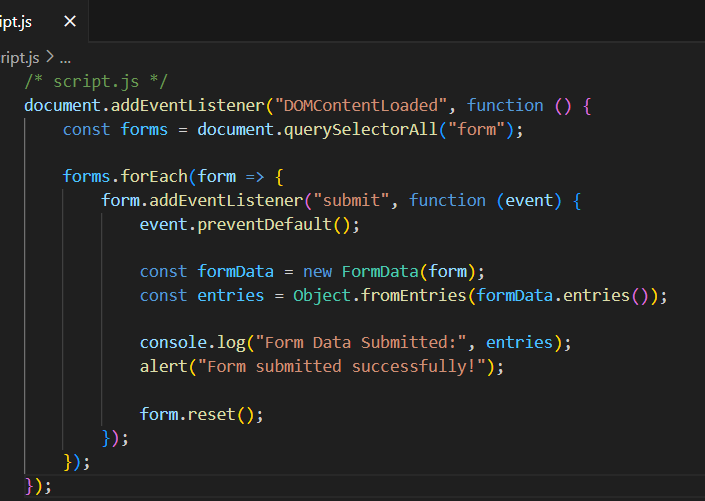
Формы имеют светлый полупрозрачный фон, тень и ограниченную ширину. Поля ввода занимают всю ширину и имеют отступы и закругленные углы. Кнопка отправки имеет стили, аналогичные другим кнопкам сайта.

**Стили для подвала:**



Подвал имеет центрированный текст, отступы, темно-синий фон и светлый текст.

**6. Детальный анализ JavaScript-функциональности (script.js)**



Этот JavaScript-код подключается к HTML-документу и **перехватывает отправку всех форм на странице**, чтобы:

* **не перезагружать страницу** (по умолчанию форма отправляется на сервер и обновляет страницу),
* **собрать данные из формы**,
* **вывести их в консоль**,
* и показать пользователю всплывающее окно (alert).

**Подробный разбор по строкам:**

document.addEventListener("DOMContentLoaded", function () {

* Ждём, пока весь HTML загрузится, чтобы скрипт не начал искать формы до того, как они появятся в DOM.

const forms = document.querySelectorAll("form");

* Получаем **все элементы <form>** на странице.
* querySelectorAll возвращает **List**, похожий на массив, который можно перебирать.

forms.forEach(form => {

* Запускаем цикл по всем найденным формам. Даже если форм 10 — всё схвачено.

form.addEventListener("submit", function (event) {

* Навешиваем обработчик на **событие отправки** (submit) каждой формы.

event.preventDefault();

* Блокируем **стандартное поведение**: форма **не будет отправляться на сервер**, и страница **не будет перезагружаться**.
* Теперь всё в руках JavaScript'а.

const formData = new FormData(form);

* Создаём объект FormData, в котором автоматически собраны все <input>, <select>, <textarea> и т.п. из формы.
* FormData — это как корзина, в которую собрали всё, что ввёл пользователь.

const entries = Object.fromEntries(formData.entries());

* Преобразуем FormData в обычный объект JavaScript ({ ключ: значение }), чтобы было удобно работать.
* entries() возвращает пары ключ-значение, а Object.fromEntries() превращает их в объект.

console.log("Form Data Submitted:", entries);

* Выводим собранные данные в консоль. Типа: "Вот что заполнил пользователь".

alert("Form submitted successfully!");

* Показываем пользователю сообщение — чисто для обратной связи.

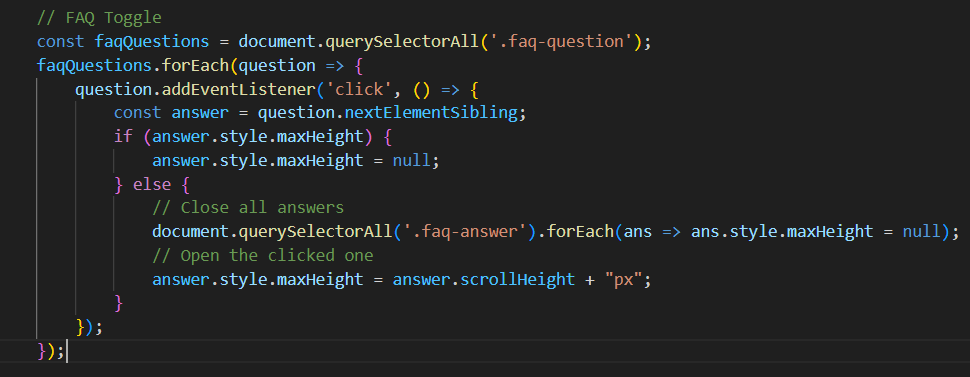
form.reset();

* Сбрасываем форму — все поля становятся пустыми.
* Можно сказать, форма "обнуляется".

JavaScript-код выполняет следующие действия:

1. Дожидается полной загрузки DOM-структуры страницы
2. Находит все формы на странице с помощью querySelectorAll
3. Для каждой формы добавляет обработчик события submit
4. При отправке формы:
   * Предотвращает стандартное поведение (перезагрузку страницы) с помощью event.preventDefault()
   * Создает объект FormData для сбора данных формы
   * Преобразует данные в объект JavaScript с помощью Object.fromEntries
   * Выводит собранные данные в консоль для отладки
   * Показывает пользователю сообщение об успешной отправке
   * Сбрасывает форму для нового ввода

Этот код работает одинаково для всех форм на сайте, как для формы регистрации на главной странице, так и для формы обратной связи на странице контактов.



const faqQuestions = document.querySelectorAll('.faq-question');

* Эта строка находит все элементы с классом .faq-question (которые являются кнопками для вопросов) и сохраняет их в переменную faqQuestions в виде коллекции List.

faqQuestions.forEach(question => {

* Цикл, который перебирает все элементы в коллекции faqQuestions. Для каждого вопроса будет выполняться функция, которая принимает один элемент question в качестве параметра.

question.addEventListener('click', () => {

* Для каждого вопроса добавляется обработчик события, который срабатывает при клике. Когда пользователь кликает на вопрос, выполняется функция.

const answer = question.nextElementSibling;

* Находит следующий элемент после question, который в нашем случае является элементом с ответом (то есть .faq-answer). Сохраняет его в переменную answer.

if (answer.style.maxHeight) {

answer.style.maxHeight = null;

}

* Проверяет, если у ответа уже установлена высота (maxHeight), то она сбрасывается (делаем скрытым ответ).

else {

* Если же у ответа нет заданной высоты (то есть он скрыт), то выполняется следующий код.

// Close all answers

document.querySelectorAll('.faq-answer').forEach(ans => ans.style.maxHeight = null);

* Эта строка закрывает все ответы на других вопросах, устанавливая их maxHeight в null, чтобы они скрылись.

// Open the clicked one

answer.style.maxHeight = answer.scrollHeight + "px";

* Открывает только тот ответ, по которому был произведен клик. answer.scrollHeight — это полная высота содержимого элемента (включая скрытые части), и мы задаем maxHeight на основе этой высоты, чтобы ответ плавно раскрылся.

**Заключение**

Проект "Система регистрации студентов" представляет собой сайт включающий три страницы: главную, информационную и контактную, а также связанные с ними файлы стилей и JavaScript. В процессе разработки были применены основные принципы веб-разработки, такие как создание структурированных HTML-страниц, оформление с помощью CSS и добавление интерактивности через JavaScript.

Основное внимание уделено созданию формы регистрации и формы обратной связи с базовой валидацией на стороне клиента, что позволяет избегать перезагрузки страницы при отправке данных. Формы и элементы интерфейса стилизованы для обеспечения единообразия и простоты восприятия.

Тем не менее, проект находится на начальной стадии. Для дальнейшего развития потребуется внедрить серверную обработку данных, хранение информации в базе данных и усовершенствование механизмов аутентификации. Также важно улучшить безопасность, производительность и адаптивность сайта для различных устройств.

В будущем проект можно расширить с использованием фреймворков, таких как React или Node.js, для создания более динамичного интерфейса и улучшения пользовательского опыта.

**Ссылка**: https://github.com/Cerezzas/student-regisrtration-system.git