**АЗЕРБАЙДЖАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ПРОМЫШЛЕННОСТИ**

**Факультет: Информационные технологии и управление**

**Кафедра: Компьютерная инженерия**

**Предмет: «Веб-системы и технологии»**

Курсовая работа

**Тема:** «Разработка онлайн информационной системы для отельных ресторанов»

**Группа:** 680.22

**Курc:** 3

**Специальность: 050616** Информационные технологии

**Студент**: Мамалиева Эмилия

**Руководитель:** **асс. Халилов М. Э.**

**Зав.кафедрой: доц.Рагимова Н.А.**

**Баку – 2025**

**АЗЕРБАЙДЖАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ПРОМЫШЛЕННОСТИ**

**ЗАДАНИЕ ПО КУРСОВОЙ РАБОТЕ**

**ФАКУЛЬТЕТ**: Информационные технологии и управление

**КАФЕДРА**: Компьютерная инженерия

**Группа** 680.22 **курс** III

**Специальность** **050616** Информационные технологии

**Студент** Мамалиева Эмилия

**Зачетная книжка**

**Руководитель курсовой работы**  **асс. Халилов М. Э.**

**Срок выдачи**

**Дата сдачи**

**Тема курсовой работы** «Разработка онлайн информационной системы

для отельных ресторанов»

**Отзыв руководителя курсовой**:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Подпись студента**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Подпись рук.курсовой**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Подпись зав.кафедрой**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Дата защиты курсовой**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Оценка**\_\_\_\_\_\_\_\_\_\_

**Пред.комиссии**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

**Члены комиссии**: 1.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

2.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

3.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

**СОДЕРЖАНИЕ**

[**Введение** 4](#_Toc196943091)

[**1.ТЕОРЕТИЧЕСКАЯ ЧАСТЬ** 5](#_Toc196943092)

[**1.1 HTML** 5](#_Toc196943093)

[**1.2 CSS** 8](#_Toc196943094)

[**1.3 JavaScript** 14](#_Toc196943095)

[**2.ПРАКТИЧЕСКАЯ ЧАСТЬ** 17](#_Toc196943096)

[**2.1 Описание предметной области** 17](#_Toc196943097)

[**2.2 Основные компоненты HTML-документа** 21](#_Toc196943098)

[**2.3 CSS стили для сайта** 28](#_Toc196943099)

[**2.4 JavaScript для функционала** 41](#_Toc196943100)

[**3. РЕЗУЛЬТАТ** 45](#_Toc196943101)

[**4. ЛИТЕРАТУРА** 47](#_Toc196943102)

# **Введение**

**Веб-программирование** — это процесс создания веб-сайтов и веб-приложений с использованием различных технологий. Оно охватывает как разработку интерфейса, с которым взаимодействует пользователь, так и серверную логику, отвечающую за обработку данных и работу приложения. Современные сайты являются неотъемлемой частью бизнеса, образования, развлечений и повседневной жизни. Они позволяют автоматизировать процессы, обмениваться информацией и предоставлять круглосуточный доступ к услугам. Благодаря веб-технологиям пользователи могут совершать покупки, обучаться, общаться и управлять различными сервисами онлайн.

**Основные технологии веб-программирования**

1. **HTML (HyperText Markup Language)**  
   HTML — основа любой веб-страницы. Он задаёт структуру документа: заголовки, абзацы, списки, изображения, таблицы, формы и ссылки. HTML-код организован в виде вложенных тегов с атрибутами, которые уточняют поведение и внешний вид элементов.
2. **CSS (Cascading Style Sheets)**  
   CSS отвечает за визуальное оформление HTML-элементов: цвета, шрифты, отступы, расположение блоков, эффекты и анимации. CSS позволяет создавать адаптивный дизайн, благодаря чему сайт корректно отображается на компьютерах, планшетах и смартфонах. Используются медиазапросы, чтобы стили подстраивались под размер экрана.
3. **JavaScript**  
   JavaScript делает сайты интерактивными. Он реагирует на действия пользователя — нажатия кнопок, ввод данных, прокрутку — и может изменять содержимое страницы без перезагрузки. Современные фреймворки (например, React, Vue, Angular) позволяют строить сложные одностраничные приложения (SPA) с высокой производительностью и удобным интерфейсом.
4. **Серверные технологии и базы данных**  
   Серверная часть (backend) обрабатывает запросы от клиента, выполняет бизнес-логику и взаимодействует с базой данных. Популярные серверные языки — PHP, Python, Node.js, Java. Базы данных, такие как MySQL, PostgreSQL (SQL) и MongoDB (NoSQL), хранят данные пользователей, заказы, товары и другую информацию. С помощью SQL-запросов данные можно искать, изменять и удалять.

# **1.ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

## **1.1 HTML**

HTML (HyperText Markup Language) — это стандартный язык разметки, используемый для создания и структурирования содержимого веб-страниц. Он является основой для большинства веб-сайтов и интернет-приложений, обеспечивая структуру и форматирование текста, изображений, ссылок и других элементов на страницах. HTML позволяет браузерам интерпретировать и отображать данные, которые мы видим в интернете.

**Основы HTML**

HTML не является языком программирования, а языком разметки. Он использует теги для определения элементов на веб-странице. Каждый тег обрабатывает определённый тип контента, будь то текст, изображения, ссылки или формы. Теги HTML обычно имеют открывающие и закрывающие элементы, между которыми содержится контент.

**Структура документа HTML**

HTML-документ имеет четко определенную структуру, которая начинается с объявления типа документа <!DOCTYPE html> и заканчивается тегом </html>. Основными частями документа являются:

1. **<html>** — корневой элемент, который содержит весь HTML-контент.
2. **<head>** — элемент, который содержит метаинформацию о странице (например, мета-теги, ссылку на стили, шрифты и т. д.).
3. **<body>** — часть документа, в которой размещается видимое содержимое веб-страницы (тексты, изображения, ссылки и другие элементы).

**Теги HTML**

**Структурные элементы**

HTML предоставляет несколько важных структурных тегов, которые определяют содержимое и его структуру на странице:

* **<html>**: Оборачивает весь HTML-контент страницы.
* **<head>**: Содержит метаинформацию, которая не отображается на странице, например, название страницы, кодировку, ссылки на стили и скрипты.
* **<title>**: Определяет название веб-страницы, которое отображается на вкладке браузера.
* **<body>**: Содержит видимое содержимое страницы, такое как текст, изображения, формы и другие элементы.

**Теги для текста**

HTML включает различные теги для форматирования и организации текста на веб-странице:

* **<h1> - <h6>**: Заголовки разного уровня. <h1> — самый важный заголовок, а <h6> — самый низкий по уровню.
* **<p>**: Абзац текста.
* **<strong>**: Тег для выделения текста жирным шрифтом.
* **<em>**: Тег для выделения текста курсивом.
* **<br>**: Перенос строки.
* **<a>**: Тег для создания гиперссылок. Содержит атрибут href, который указывает URL.

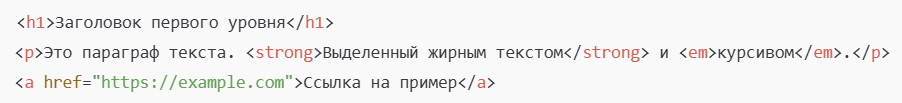


Рис. 1.1. Пример использования тегов для текста

**Теги для изображения и медиа**

HTML позволяет вставлять изображения и медиа-файлы с помощью соответствующих тегов:

* **<img>**: Вставка изображения. Этот тег не имеет закрывающего тега и включает атрибуты src (путь к изображению) и alt (описание изображения).
* **<audio>**: Вставка аудиофайла.
* **<video>**: Вставка видеоролика.

**Списки**

HTML позволяет создавать два типа списков:

* **<ul>**: Ненумерованный список.
* **<ol>**: Нумерованный список.
* **<li>**: Элемент списка.

A screenshot of a computer program

AI-generated content may be incorrect.

Рис. 1.2. Пример создания списка

**Таблицы**

Таблицы в HTML создаются с использованием следующих тегов:

* **<table>**: Тег для создания таблицы.
* **<tr>**: Строка таблицы.
* **<th>**: Ячейка заголовка таблицы.
* **<td>**: Ячейка данных таблицы.

A screenshot of a computer code

AI-generated content may be incorrect.

Рис. 1.3. Пример таблицы

**Формы**

HTML формы используются для сбора данных от пользователей. Основные элементы формы:

* **<form>**: Определяет саму форму.
* **<input>**: Используется для различных типов ввода (текст, число, кнопка и т. д.).
* **<textarea>**: Многострочное текстовое поле.
* **<button>**: Кнопка для отправки формы.

**Атрибуты HTML**

Атрибуты в HTML предоставляют дополнительную информацию об элементах и определяют их свойства. Например:

* **href**: Указывает URL для гиперссылки.
* **src**: Указывает источник изображения или видео.
* **alt**: Описание изображения для пользователей с ограниченными возможностями.
* **class**: Применяет CSS-стили к элементу.
* **id**: Уникальный идентификатор элемента.

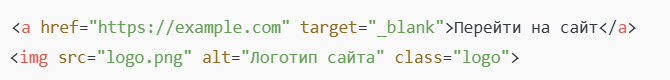


Рис. 1.4. Пример с атрибутами

## **1.2 CSS**

CSS (Cascading Style Sheets) — это язык стилей, используемый для описания внешнего вида веб-страниц, созданных с помощью HTML. CSS позволяет отделить структуру документа (HTML) от его представления, что упрощает редактирование и поддержание кода. Благодаря CSS можно задавать оформление текста, макет страницы, цвета, шрифты и другие визуальные элементы, а также адаптировать сайт под разные устройства с помощью адаптивного дизайна.

**Основы CSS**

CSS состоит из селекторов и деклараций. Селектор указывает на элемент HTML, к которому применяется стиль, а декларация определяет, какие свойства и значения будут применены. Декларации заключаются в фигурные скобки {}, а каждое свойство с его значением разделяются двоеточием :, после чего ставится точка с запятой ; для разделения нескольких свойств.



Рис. 1.5. Пример CSS-кода

В этом примере селектор h1 применяет стиль к заголовкам первого уровня. Свойства color и font-size определяют цвет текста и размер шрифта, соответственно.

**Вставка CSS в HTML**

CSS можно вставить в HTML-документ несколькими способами:

**1.Inline-стили**:



Рис. 1.6. Применение стилей непосредственно в атрибуте элемента HTML

Этот метод удобен для применения стилей к отдельным элементам, но не рекомендуется для крупных проектов, так как он делает код громоздким.

**2.Встроенные стили**:

A computer code with text

AI-generated content may be incorrect.

Рис. 1.7. Вставка CSS с использованием тега <style>.

Этот метод удобен для небольших сайтов или страниц, но с увеличением объема кода лучше использовать внешний CSS.

**3.Внешние стили**: Подключение внешнего файла CSS с помощью тега <link> в секции <head>.

A close-up of a white background

AI-generated content may be incorrect.

Рис. 1.8. Подключение внешнего файла CSS с помощью тега <link>

Это наиболее гибкий и масштабируемый способ, который позволяет разделить структуру (HTML) и стиль (CSS) на разные файлы, что упрощает поддержку и переработку сайта.

**Селекторы CSS**

Селекторы CSS используются для выбора элементов на странице, к которым будут применяться стили. Существует несколько типов селекторов:

**1.Селекторы по тегу (типу элемента)**: Применяются ко всем элементам определенного типа. Например, p применяет стиль ко всем параграфам на странице.

A black and green text

AI-generated content may be incorrect.

Рис. 1.9. Селектор по тегу

**2.Селекторы по классу**: Применяются ко всем элементам, которые имеют определенный класс. Классы обозначаются точкой перед именем.

A computer code with text

AI-generated content may be incorrect.

Рис. 1.10. Пример селектора по классу



Рис. 1.11. Пример применения к элементу

**3.Селекторы по идентификатору**: Применяются к элементам с определенным уникальным идентификатором. Идентификаторы обозначаются решеткой перед именем.

A close-up of a computer code

AI-generated content may be incorrect.

Рис. 1.12. Пример селектора по идентификатору



Рис. 1.13. Пример применения к элементу

**4.Комбинированные селекторы**: Можно комбинировать несколько селекторов для более точного выбора элементов. Например, селектор для всех параграфов внутри элемента с классом .content.

A close-up of a computer code

AI-generated content may be incorrect.

Рис. 1.14. Пример комбинированного селектора

**5.Селекторы потомков и дочерних элементов**: Селекторы позволяют выбрать дочерние элементы внутри родительских. Селектор для всех элементов li, которые находятся внутри ul, выглядит так:

A close-up of a text

AI-generated content may be incorrect.

Рис. 1.15. Пример селектора потомков и дочерних элементов

**Свойства CSS**

**1.Цвет и фон**:

**color**: Определяет цвет текста.

**background-color**: Задает цвет фона элемента.

**background-image**: Устанавливает фоновое изображение.

A close-up of a computer code

AI-generated content may be incorrect.

Рис. 1.16. Пример добавления стилей цвета и фона

**2.Шрифт**:

**font-family**: Устанавливает шрифт для текста.

**font-size**: Определяет размер шрифта.

**font-weight**: Устанавливает жирность шрифта.

**line-height**: Задает высоту строки.

A screenshot of a computer code

AI-generated content may be incorrect.

Рис. 1.17. Пример добавления стилей шрифта

**3.Отступы и поля**:

**margin**: Определяет внешние отступы элемента.

**padding**: Устанавливает внутренние отступы (между содержимым элемента и его границей).

A close-up of a computer code

AI-generated content may be incorrect.

Рис. 1.18. Пример добавления отступов и полей

**4.Границы**:

**border**: Задает стиль, ширину и цвет границы вокруг элемента.

**border-radius**: Устанавливает скругленные углы для элемента.

A close-up of a computer code

AI-generated content may be incorrect.

Рис. 1.19. Пример добавления границ

**5.Размеры и позиции**:

**width и height**: Определяют ширину и высоту элемента.

**position**: Управляет позиционированием элемента (например, absolute, relative, fixed, sticky).

**top, right, bottom, left**: Используются вместе с позиционированием для задания координат элемента.

A screen shot of a computer code

AI-generated content may be incorrect.

Рис. 1.20. Пример добавления размеров и позиции

**6.Тени**:

**box-shadow**: Добавляет тень вокруг элемента.

**text-shadow**: Добавляет тень к тексту.

A screenshot of a computer code

AI-generated content may be incorrect.

Рис. 1.21. Пример добавления теней

**Адаптивный дизайн и медиа-запросы**

Одной из главных задач современного веб-дизайна является создание сайтов, которые корректно отображаются на различных устройствах. Для этого используется подход адаптивного дизайна, который позволяет изменять стили в зависимости от характеристик устройства, на котором просматривается сайт.

Медиа-запросы (@media) — это CSS-инструкция, которая позволяет применять стили в зависимости от ширины экрана, ориентации устройства и других параметров.

A screenshot of a computer code

AI-generated content may be incorrect.

Рис. 1.22. Пример медиа-запросов

Этот код применяет стили только для экранов шириной 600 пикселей и менее. Это позволяет сделать сайт адаптивным, изменяя его оформление на мобильных устройствах.

## **1.3 JavaScript**

**JavaScript** — это один из самых популярных языков программирования, используемый для создания динамичных и интерактивных веб-страниц. JavaScript позволяет добавлять функциональность и логику на клиентской стороне, а также взаимодействовать с сервером.

**Основные особенности JavaScript**

JavaScript выполняется в браузере, что позволяет создавать динамичные веб-страницы. JavaScript используется для обработки событий, например, кликов, ввода текста и других действий пользователя.



Рис. 1.23. Пример на JavaScript

В этом примере мы добавляем обработчик события для кнопки с id myButton. Когда пользователь нажимает на кнопку, появляется окно с сообщением.

**Переменные и типы данных**

JavaScript поддерживает несколько типов данных, среди которых:

* **Числа** (например, 42, 3.14)
* **Строки** (например, "Привет, мир!")
* **Логические значения** (например, true или false)
* **Массивы** (например, [1, 2, 3])
* **Объекты** (например, { name: 'Иван', age: 25 })

Переменные можно объявлять с помощью ключевых слов **var, let** или **const**. На практике предпочтительнее использовать **let** и **const**, так как они обеспечивают лучшую обработку областей видимости.

A computer screen shot of a computer code

AI-generated content may be incorrect.

Рис. 1.24. Пример с переменными на JavaScript

В этом примере name — это переменная, значение которой можно изменять, а age — константа, значение которой не изменяется.

**Условия и циклы**

JavaScript поддерживает различные способы работы с логическими условиями и повторяющимися действиями через операторы **if, else, switch** и циклы **for, while**.

A computer code with different colored text

AI-generated content may be incorrect.

Рис. 1.25. Пример с условиями на JavaScript

Циклы используются для выполнения одного и того же кода несколько раз. Например, цикл **for** позволяет перебрать все элементы массива.

A screenshot of a computer code

AI-generated content may be incorrect.

Рис. 1.26. Пример с циклом на JavaScript

**Функции**

Функции — это блоки кода, которые выполняют определенные задачи. Они могут принимать параметры и возвращать результаты. Функции помогают организовать код и делают его более модульным и удобным для повторного использования.

A close-up of a computer code

AI-generated content may be incorrect.

Рис. 1.27. Пример с функцией на JavaScript

В этом примере функция greet принимает имя как аргумент и возвращает приветственное сообщение.

# **2.ПРАКТИЧЕСКАЯ ЧАСТЬ**

## **2.1 Описание предметной области**

Предметной областью данной курсовой работы является **отельный ресторан**, а именно — его представление в виде информационного веб-ресурса. Ресторан, входящий в структуру гостиничного комплекса, играет важную роль в обеспечении комфорта и высокого уровня обслуживания гостей, а также является точкой притяжения для туристов и местных жителей.

Сайт отельного ресторана выполняет ряд ключевых функций:

* Информирует пользователей о ресторане, его истории, концепции, кухне и услугах.
* Представляет интерактивное меню с описанием блюд и ценами.
* Предоставляет возможность онлайн-бронирования столиков.
* Показывает фотогалерею интерьера и блюд, что помогает создать у посетителя первое визуальное впечатление.

Таким образом, предметная область сочетает в себе элементы ресторанного сервиса, гостиничного бизнеса и веб-технологий. Основная цель разработки сайта — создание привлекательного, функционального и интуитивно понятного веб-приложения, ориентированного как на гостей отеля, так и на внешнюю аудиторию. Разработка сайта основывается на применении **HTML**, **CSS** и **JavaScript**.

**Секция «Главная страница»**

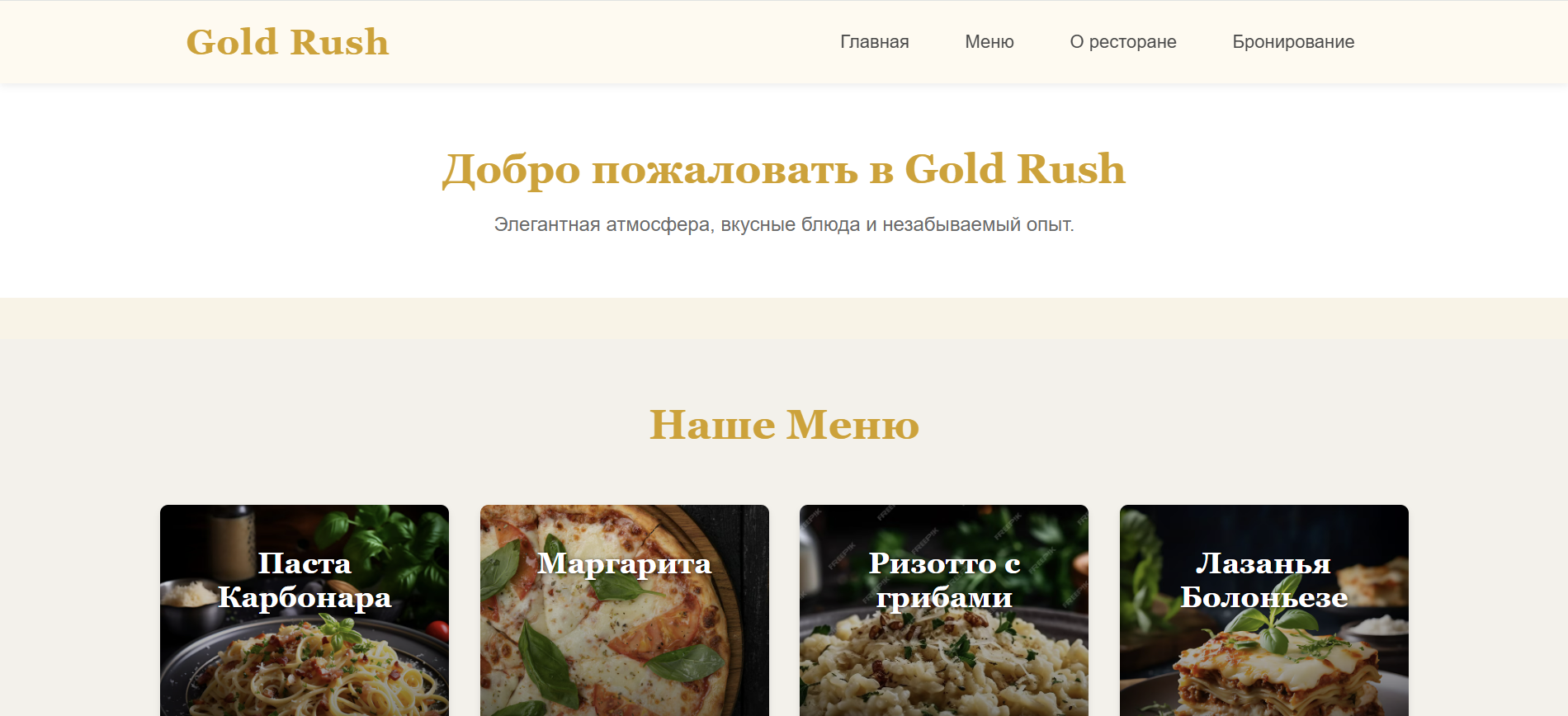


Рис. 2.1. Главная страница

Секция "Главная страница" — это первая визуально заметная часть сайта, которую видит пользователь после навигационного меню. Она служит **приветствием** и создает **первое впечатление** о ресторане.

Секция "Главная" связана с навигацией: если пользователь нажимает в меню на слово «Главная», сайт плавно прокручивается именно к этой части. Также, если сайт открывается с самого начала, пользователь сразу видит именно эту зону.

**Секция «Наше Меню»**

A screenshot of a menu

AI-generated content may be incorrect.

Рис. 2.2. Наше Меню

Секция **«Наше меню»** — это вторая основная часть сайта, предназначенная для того, чтобы **ознакомить посетителей с блюдами**, которые предлагает ресторан. Секция находится ниже «Главной» и доступна при прокрутке или через пункт меню «Меню».

**Содержимое**

Внутри секции представлены:

1. **Заголовок** — «Наше Меню». Он даёт понять, что дальше будет информация о блюдах.
2. **Список блюд в виде карточек** — визуальные блоки с фотографиями, названиями и ценами блюд. Каждое блюдо отображается в виде прямоугольника с фоном-фотографией (привлекательное фото еды), а поверх изображения размещены:
   * Название блюда
   * Цена блюда
3. **Кликабельность некоторых карточек** — например, при нажатии на «Паста Карбонара» или «Маргарита» можно перейти на отдельную страницу.

**Секция «О ресторане»**

A screenshot of a website

AI-generated content may be incorrect.

Рис. 2.3. О ресторане

Секция **«О ресторане»** рассказывает о самом заведении Gold Rush. В ней содержится краткое описание ресторана, подчёркивающее его уютную атмосферу в стиле старинной Италии. Дополнительно в этой секции размещена **фотогалерея**, где показаны интерьер и обстановка заведения. Главная цель этой части сайта — **создать эмоциональную связь**, показать уникальность ресторана и вызвать у пользователя желание его посетить. Всё оформлено в спокойном, элегантном стиле, соответствующем общей эстетике сайта.

**Секция «Бронирование**

A screenshot of a computer

AI-generated content may be incorrect.

Рис. 2.4. Бронирование

Секция **«Бронирование»** предназначена для того, чтобы посетители сайта могли удобно и быстро забронировать столик в ресторане **Gold Rush**. Здесь размещена простая и понятная форма, где пользователь указывает своё имя, дату и время бронирования, а также количество гостей.

Форма интуитивно оформлена: каждому полю предшествует понятная подпись. Внизу формы расположена кнопка, по нажатию на которую отправляется заявка на бронирование.

Основная цель этой секции — **обеспечить удобный онлайн-сервис**, чтобы клиент мог заранее запланировать визит в ресторан, не тратя время на звонки или визиты лично.

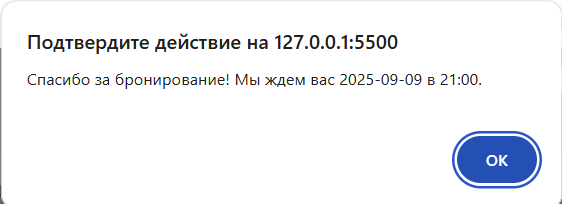


Рис. 2.5. Всплывающее окно

После того, как мы забронировали столик, на экран выходит всплывающее окно с текстом.

**Секция «Футер»**

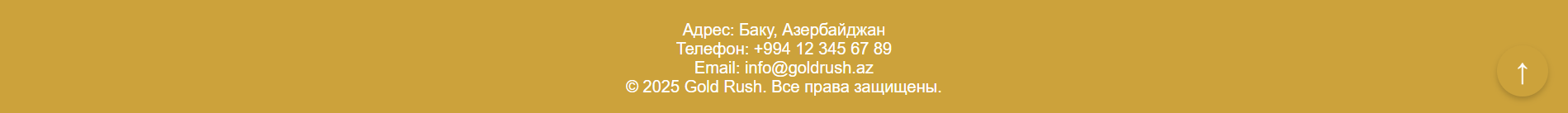


Рис. 2.6. Футер

**Футер** внизу страницы содержит контактную информацию ресторана **Gold Rush**: адрес (Баку, Азербайджан), телефон и электронную почту для связи. Также указаны авторские права, подтверждающие, что сайт защищён. Это важная часть сайта для пользователей, которые хотят быстро найти контактные данные.

## **2.2 Основные компоненты HTML-документа**

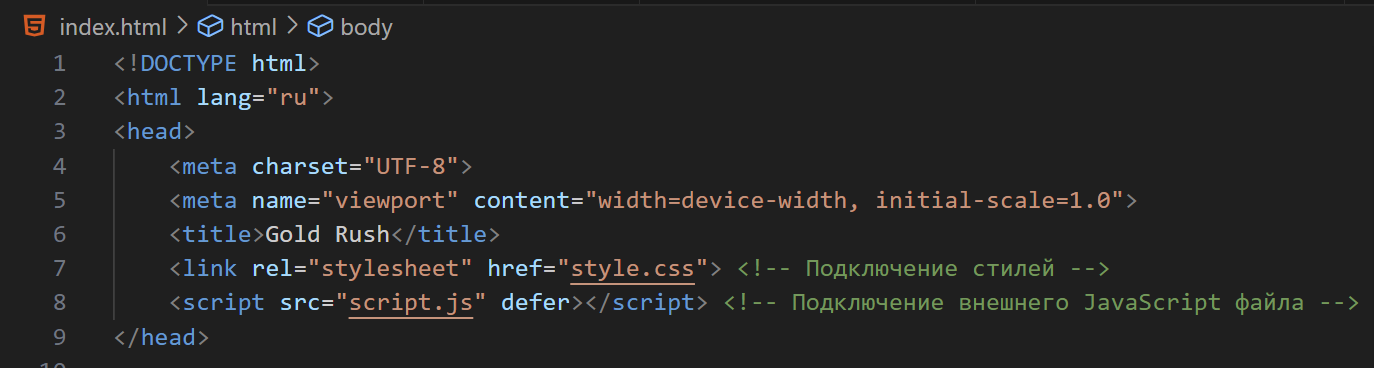


Рис. 2.7. Заголовочная часть HTML-документа (head): мета-теги, стили и скрипты

**1. <!DOCTYPE html>**

Этот тег сообщает браузеру, что документ написан в HTML5, что важно для корректного отображения веб-страницы на современных устройствах и браузерах.

**2. <html lang="ru">**

Открывающий тег <html> обозначает начало HTML-документа. Атрибут lang="ru" указывает, что язык документа — русский. Это важно для правильной работы с поисковыми системами и доступности для пользователей.

**3. <head>**

В теге <head> находятся мета-данные и ссылки на внешние ресурсы, такие как стили и скрипты.

* **<meta charset="UTF-8">** — Устанавливает кодировку символов в UTF-8, что позволяет корректно отображать текст на русском языке.
* **<meta name="viewport" content="width=device-width, initial-scale=1.0">** — Этот тег отвечает за адаптивность сайта на мобильных устройствах. Устанавливает масштаб страницы в зависимости от размера экрана устройства.
* **<title>Gold Rush</title>** — Название веб-страницы, которое отображается в заголовке окна браузера.
* **<link rel="stylesheet" href="style.css">** — Подключение внешнего CSS-файла для оформления страницы.
* **<script src="script.js" defer></script>** — Подключение внешнего JavaScript-файла, который будет загружаться после рендеринга HTML-контента (атрибут defer).

**Навигационное меню**

A computer screen shot of text

AI-generated content may be incorrect.

Рис. 2.8. Навигационное меню в VS Code

**1. <header>**

Тег <header> используется для обозначения шапки страницы, в которой размещается навигационное меню. Это важная часть структуры веб-страницы, предоставляющая быстрый доступ ко всем основным разделам сайта.

**2. <nav>**

Тег <nav> служит для группировки элементов навигации. Здесь размещены ссылки на основные разделы сайта: "Главная", "Меню", "О ресторане" и "Бронирование".

**3. <ul> и <li>**

Эти теги создают ненумерованный список (<ul>) с элементами списка (<li>), которые в свою очередь содержат ссылки (<a>). Каждая ссылка ведет к соответствующему разделу страницы, что позволяет пользователю быстро перемещаться по сайту.

**Основной контент сайта**

**1. Главная страница**

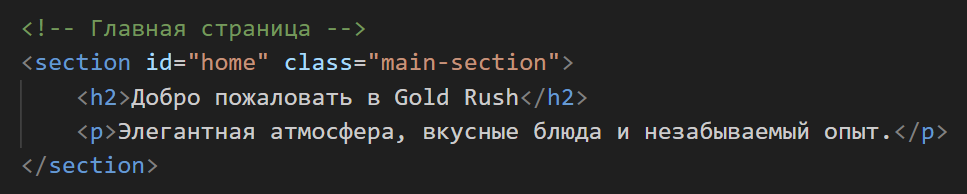
****

Рис. 2.9. Главная страница в VS Code

Тег <section> используется для логического разделения контента на странице. В этом случае он представляет раздел с главной информацией о ресторане. Атрибут id="home" необходим для привязки ссылки на навигационное меню, чтобы при клике на ссылку в меню происходил переход в этот раздел.

**2. Меню ресторана**



Рис. 2.10. Меню ресторана в VS Code

Заголовок <h2> указывает на название раздела "Наше Меню". Все блюда находятся в контейнере <div class="menu-items">.

Для каждого блюда:

* Ссылка <a> ведет на страницу с подробным описанием.
* <div class="menu-item"> отображает карточку блюда с фоновым изображением.
* <h3> и <span> показывают название блюда и цену.

**3.О ресторане**

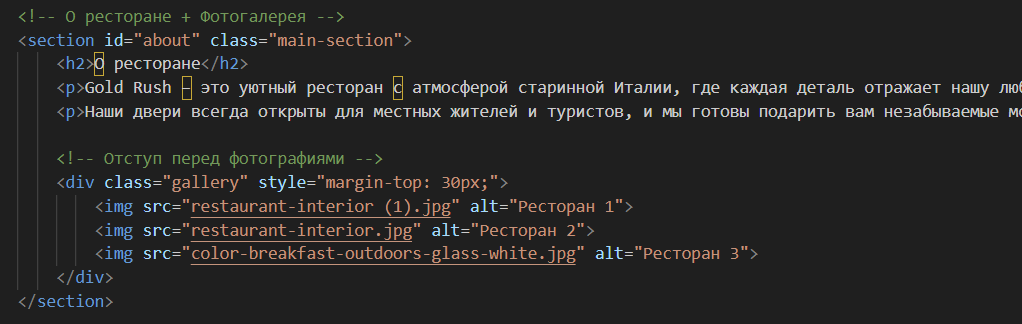


Рис. 2.11. О ресторане в VS Code

В разделе "О ресторане" представлен текст, который знакомит пользователя с концепцией и атмосферой заведения. Далее идет галерея с изображениями, оформленная через <div class="gallery">. Каждое изображение оформлено с помощью тега <img> с атрибутом alt, который используется для описания изображения.

**4. Форма бронирования**

A computer screen with text

AI-generated content may be incorrect.

Рис. 2.12. Форма бронирования в VS Code

**Заголовок** <h2>: отображает название раздела "Бронирование".

**Форма бронирования**:

* Содержит атрибут id="bookingForm", чтобы можно было легко обращаться к ней через JavaScript.

**Поля формы**:

* **Имя**: Текстовое поле для ввода имени пользователя с атрибутом required, который требует обязательного заполнения этого поля.
* **Дата бронирования**: Поле для выбора даты через календарь (в формате date), также обязательное.
* **Время**: Поле для ввода времени (в формате time), с обязательным заполнением.
* **Количество человек**: Поле для ввода числа гостей (тип number), где минимальное значение равно 1, также обязательное.

**Кнопка отправки**: Кнопка с текстом "Забронировать столик", которая отправляет форму при нажатии.

**Футер**

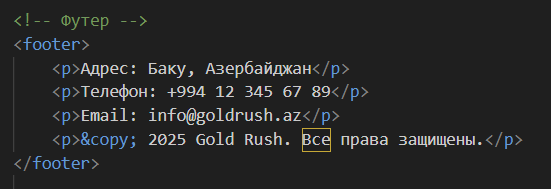
****

Рис. 2.13. Футер в VS Code

Тег <footer> представляет нижнюю часть страницы, где обычно размещается информация о контактных данных, авторских правах и другие дополнительные сведения. Здесь содержится информация о местоположении ресторана, его телефоне и электронном адресе.

**Кнопка "Наверх"**

**A black background with white text

AI-generated content may be incorrect.**

Рис. 2.14. Кнопка «Наверх» в VS Code

Этот элемент <button> представляет кнопку для возврата в верхнюю часть страницы. При помощи JavaScript она будет появляться при прокрутке страницы вниз и позволять пользователю быстро вернуться в начало.

Когда мы нажимаем на блюдо в секции "Наше меню", открывается новый HTML-документ, где представлена подробная информация о выбранном блюде, включая его описание и ингредиенты. В этом документе содержится все необходимое, чтобы познакомить посетителей с особенностями блюда, например, как в случае с пастой Карбонара, где указаны все компоненты, из которых она состоит. Этот новый HTML-документ включает изображение блюда, его цену, описание, а также список ингредиентов с иконками, что помогает пользователю лучше понять состав блюда.

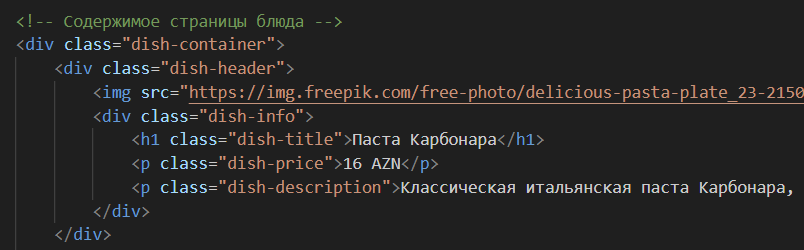


Рис. 2.15. Содержимое страницы блюда

Этот блок HTML представляет собой часть страницы, посвященную конкретному блюду — пасте Карбонара.

**Контейнер блюда**:

* <div class="dish-container"> — это основной контейнер для всего содержимого, связанного с конкретным блюдом. Он объединяет заголовок, изображение и информацию о блюде в одну структуру.

**Заголовок блюда и изображение**:

* <div class="dish-header"> — этот блок содержит изображение и основную информацию о блюде.
* <img src="..." alt="Паста Карбонара" class="dish-image"> — это изображение блюда, в данном случае пасты Карбонара. Атрибут src указывает путь к изображению, а атрибут alt предоставляет текстовое описание изображения для улучшения доступности.

**Информация о блюде**:

* <div class="dish-info"> — это контейнер для информации о блюде, которая включает в себя название, цену и описание.
* <h1 class="dish-title">Паста Карбонара</h1> — заголовок, в котором указано название блюда, оформленное как основной заголовок (h1).
* <p class="dish-price">16 AZN</p> — параграф с указанием цены блюда.
* <p class="dish-description">...</p> — параграф с описанием блюда.

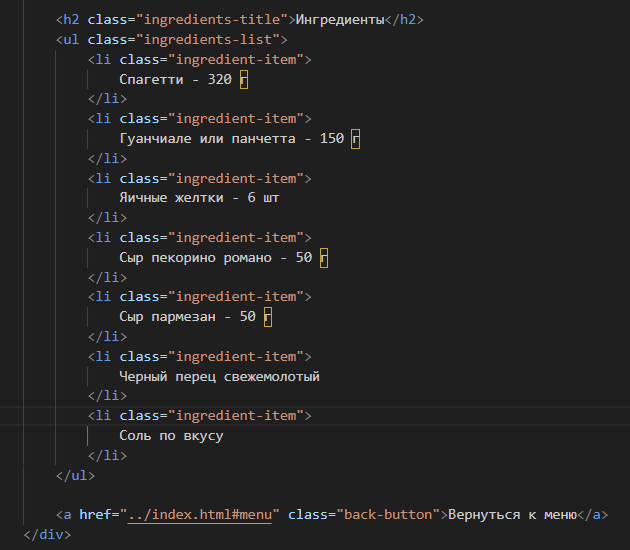


Рис. 2.16. Список ингредиентов

Этот блок HTML представляет собой список ингредиентов блюда "Паста Карбонара" и включает в себя элементы, описывающие состав блюда, а также ссылку для возврата к основному меню.

**Заголовок списка ингредиентов**:

* <h2 class="ingredients-title">Ингредиенты</h2> — заголовок второго уровня (h2), который сообщает пользователю, что далее будет перечислен список ингредиентов блюда. Заголовок имеет класс ingredients-title для стилизации.

**Список ингредиентов**:

* <ul class="ingredients-list"> — ненумерованный список (ul), в котором будут перечислены все ингредиенты блюда. Класс ingredients-list используется для стилизации всего списка.

**Элементы списка ингредиентов**:

* Каждый ингредиент представлен элементом списка <li class="ingredient-item">. Каждый элемент имеет текстовое описание ингредиента.

**Возврат к меню**:

* <a href="../index.html#menu" class="back-button">Вернуться к меню</a> — ссылка, которая позволяет пользователю вернуться к разделу "Меню" на главной странице. Атрибут href="../index.html#menu" указывает на якорь #menu на странице index.html. Класс back-button используется для стилизации этой ссылки, возможно, придавая ей вид кнопки.

Всё то же самое применяется и к другим блюдам в секции «Наше Меню».

## **2.3 CSS стили для сайта**

**Общие стили для всего сайта**

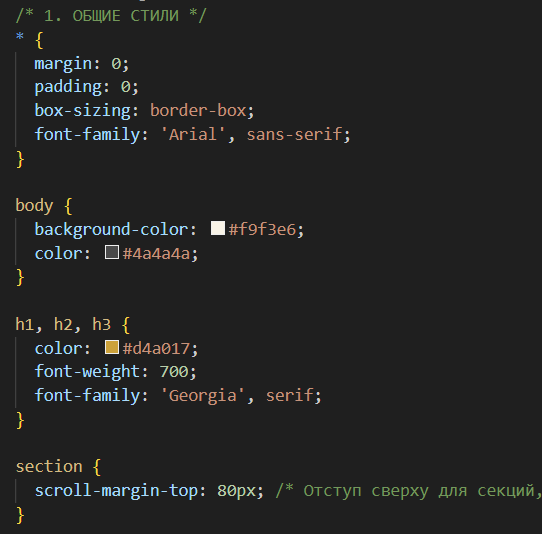


Рис. 2.14. Общие CSS стили для всего сайта

**Глобальные стили**:

* margin: 0; — удаляет внешние отступы у всех элементов на странице.
* padding: 0; — удаляет внутренние отступы у всех элементов на странице.
* box-sizing: border-box; — изменяет способ расчета размеров элементов. Вместо того, чтобы добавлять ширину и высоту элементов к их отступам и границам, эта настройка заставляет браузер учитывать их в общих размерах элемента.
* font-family: 'Arial', sans-serif; — задает шрифт для всех элементов на странице. Основной шрифт — Arial, и если он недоступен, будет использован любой другой шрифт без засечек (sans-serif).

**Стили для тела страницы (body)**:

* background-color: #f9f3e6; — задает светлый кремовый цвет фона для всей страницы.
* color: #4a4a4a; — устанавливает темный серый цвет текста для всей страницы, что создает контраст с фоном и делает текст более читаемым.

**Стили для заголовков (h1, h2, h3)**:

* color: #d4a017; — задает золотисто-желтый цвет для текста заголовков h1, h2, h3.
* font-weight: 700; — делает текст заголовков жирным.
* font-family: 'Georgia', serif; — устанавливает шрифт с засечками (serif) для заголовков, в данном случае шрифт Georgia, который придает тексту классический и элегантный вид.

**Стили для секций (section)**:

scroll-margin-top: 80px; — добавляет отступ сверху при прокрутке страницы, если элемент (секции) попадает в область видимости.

**Навигационное меню**

**A screen shot of a computer

AI-generated content may be incorrect.**

Рис. 2.15. CSS стили в навигационном меню

**Стили для header**:

* background-color: #fffaf0; — светлый кремовый фон.
* box-shadow: 0 2px 8px rgba(0, 0, 0, 0.08); — мягкая тень для отделения шапки от контента.
* position: sticky; top: 0; — фиксирует шапку в верхней части экрана при прокрутке.
* z-index: 1000; — приоритет в наложении элементов.

**Стили для nav (меню)**:

* display: flex; justify-content: space-between; — распределяет элементы по горизонтали.
* max-width: 1200px; margin: 0 auto; — центрирует меню и ограничивает его ширину.
* padding: 0 20px; — отступы внутри меню.

**Стили для логотипа**:

* font-size: 2.2rem; color: #d4a017; — увеличенный размер шрифта и золотистый цвет.
* font-family: 'Georgia', serif; — классический шрифт с засечками.
* letter-spacing: 1px; — увеличение межбуквенного интервала.

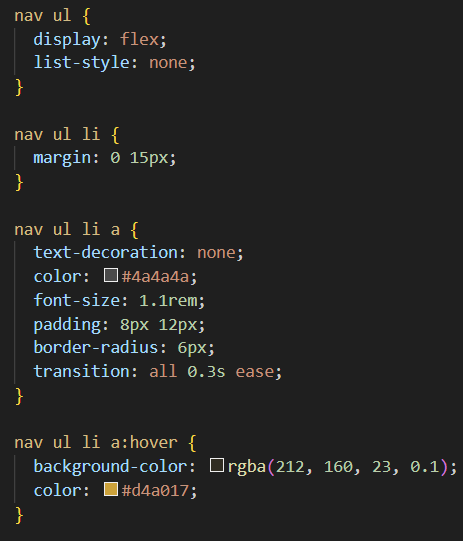
****

Рис. 2.16. CSS стили для списка навигации

**Для списка навигации (nav ul)**:

* display: flex; — горизонтальное расположение элементов.
* list-style: none; — убирает маркеры списка.

**Для элементов списка (nav ul li)**:

* margin: 0 15px; — отступы между пунктами меню.

**Для ссылок в меню (nav ul li a)**:

* text-decoration: none; — убирает подчеркивание.
* color: #4a4a4a; — серый цвет текста.
* font-size: 1.1rem; — размер шрифта.
* transition: all 0.3s ease; — плавные изменения при наведении.

**При наведении на ссылку (nav ul li a:hover)**:

* background-color: rgba(212, 160, 23, 0.1); — золотистый фон.
* color: #d4a017; — золотисто-желтый цвет текста.

**Секция "Главная страница"**

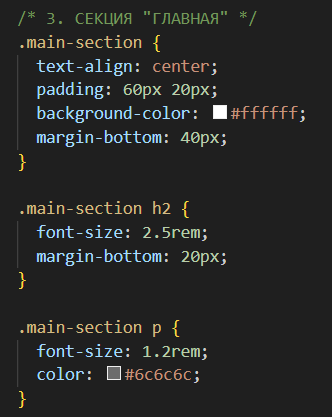


Рис. 2.17. CSS стили для секции «Главная страница»

**Для секции .main-section**:

* text-align: center; — выравнивание текста по центру.
* padding: 60px 20px; — отступы внутри секции, 60px сверху и снизу, 20px по бокам.
* background-color: #ffffff; — белый фон секции.
* margin-bottom: 40px; — отступ снизу для разделения с другими секциями.

**Для заголовка в секции (.main-section h2)**:

* font-size: 2.5rem; — увеличенный размер шрифта.
* margin-bottom: 20px; — отступ снизу.

**Для абзаца в секции (.main-section p)**:

* font-size: 1.2rem; — размер шрифта.
* color: #6c6c6c; — серый цвет текста.

**Секция "Меню"**

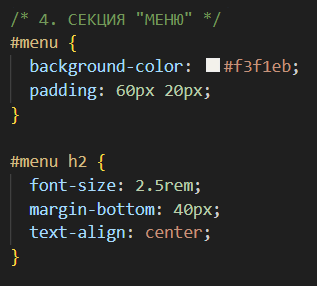


Рис. 2.17. CSS стили для секции «Меню»

**1. Для секции "Меню" (#menu):**

* background-color: #f3f1eb; — светлый фон для секции "Меню".
* padding: 60px 20px; — отступы внутри секции: 60px сверху и снизу, 20px по бокам.

**2. Для заголовка в секции "Меню" (#menu h2):**

* font-size: 2.5rem; — увеличенный размер шрифта заголовка.
* margin-bottom: 40px; — отступ снизу для разделения с другими элементами.
* text-align: center; — выравнивание заголовка по центру.

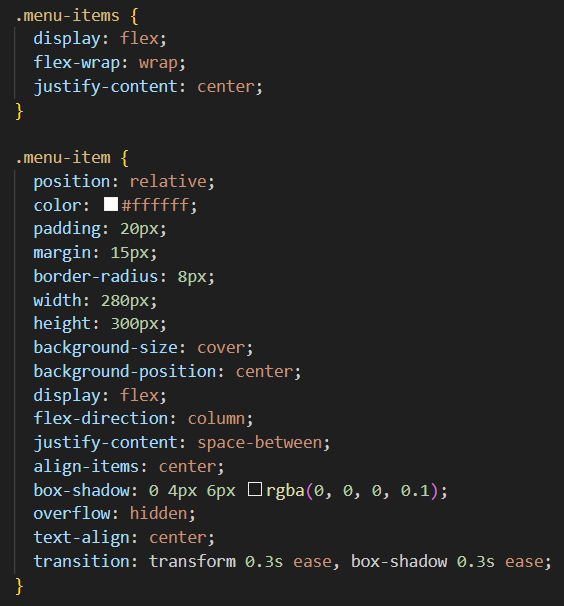


Рис. 2.18. CSS стили для контейнера с блюдами

**3. Для контейнера с блюдами (.menu-items):**

* display: flex; — делает элементы внутри контейнера горизонтальными.
* flex-wrap: wrap; — позволяет элементам переходить на новую строку, если не помещаются в одном ряду.
* justify-content: center; — выравнивает элементы по центру контейнера.

**4. Для каждого блюда в меню (.menu-item):**

* position: relative; — позволяет использовать псевдоэлемент ::before.
* padding: 20px; — внутренние отступы вокруг контента блюда.
* margin: 15px; — отступы между блюдами.
* border-radius: 8px; — скругленные углы для блюда.
* background-size: cover; — изображение фона растягивается, чтобы покрыть весь блок.
* background-position: center; — центрирует изображение фона.
* display: flex; и flex-direction: column; — вертикальное расположение элементов внутри блока.
* justify-content: space-between; — распределяет элементы по высоте.
* align-items: center; — выравнивает элементы по центру по горизонтали.
* box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1); — тень вокруг блока с блюдом.
* overflow: hidden; — скрывает части контента, выходящие за пределы блока.
* text-align: center; — выравнивание текста по центру.
* transition: transform 0.3s ease, box-shadow 0.3s ease; — плавное изменение масштаба и тени при наведении.

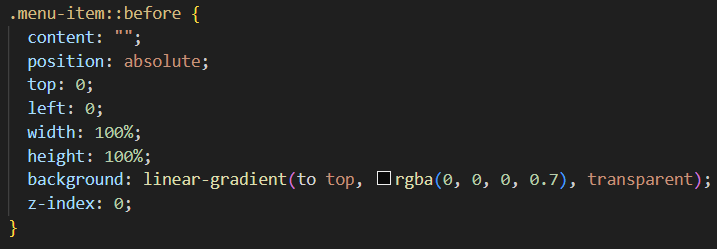


Рис. 2.19. Создание псевдоэлемента

**content: "";** — Создает пустой контент для псевдоэлемента. Это обязательное свойство для создания псевдоэлементов.

**position: absolute;** — Псевдоэлемент позиционируется относительно ближайшего предка с position: relative;. В данном случае это элемент .menu-item.

**top: 0; left: 0;** — Псевдоэлемент позиционируется в верхнем левом углу родительского элемента (например, блока с блюдом).

**width: 100%; height: 100%;** — Псевдоэлемент занимает всю ширину и высоту родительского элемента. Это позволяет покрыть весь блок, на котором находится.

**background: linear-gradient(to top, rgba(0, 0, 0, 0.7), transparent);** — Создает фон с градиентом, который идет от черного цвета с прозрачностью 0.7 (темный цвет) к прозрачному. Это дает эффект затемнения верхней части элемента.

**z-index: 0;** — Псевдоэлемент находится ниже содержимого элемента (например, текста и изображения), потому что z-index: 0 ставит его на самый низкий уровень в стеке слоев.

**Секция «О ресторане»**

A screen shot of a computer program

AI-generated content may be incorrect.

Рис. 2.20. CSS стили для секции «О ресторане»

Стили для секции "О ресторане":

* background-color: #ffffff — белый фон.
* padding: 60px 20px — отступы сверху/снизу и по бокам.
* text-align: center — выравнивание текста по центру.

**Секция «Бронирование»**

A computer screen shot of a black background

AI-generated content may be incorrect.

Рис. 2.21. CSS стили для секции «Бронирование»

Стили для секции бронирования:

* background-color: #f3f1eb — светло-бежевый фон.
* padding: 60px 20px — отступы.
* text-align: center — выравнивание контента по центру.

Стили для формы бронирования:

* max-width: 600px — ограничение максимальной ширины формы.
* margin: 0 auto — центрирование формы по горизонтали.
* background-color: #ffffff — белый фон для формы.
* box-shadow — добавляет тень.

**Секция «Фотографии»**

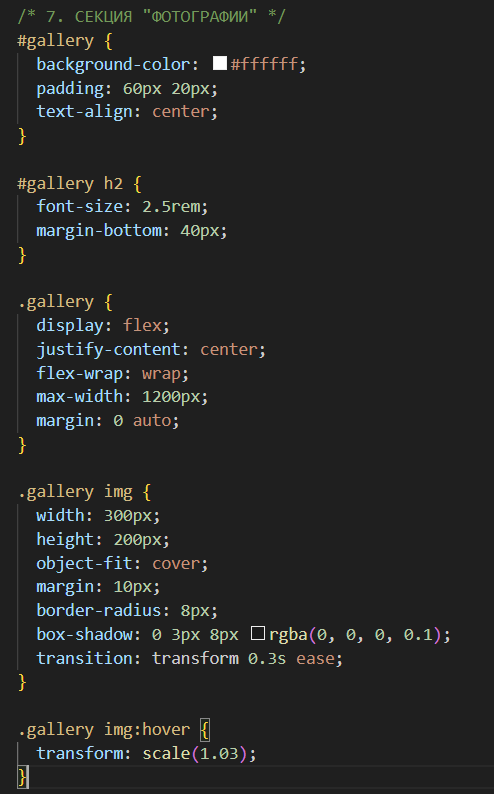


Рис. 2.22. CSS стили для секции «Фотографии»

**#gallery**: Устанавливает белый фон, отступы (60px сверху и 20px по бокам) и центрирует текст.

**#gallery h2**: Задает размер шрифта для заголовка секции и отступ снизу.

**.gallery**:

* Использует Flexbox для центрирования и распределения изображений, позволяет изображениям обтекать, если их больше, чем в одну строку.
* Ограничивает максимальную ширину секции до 1200px и выравнивает по центру.

**.gallery img**: Задает размеры для изображений (300x200px), с обрезкой содержимого (object-fit: cover), добавляет отступы, скругляет углы и добавляет тень для эффекта.

**.gallery img:hover**: При наведении на изображение оно немного увеличивается (масштабируется).

**Кнопка «Наверх»**

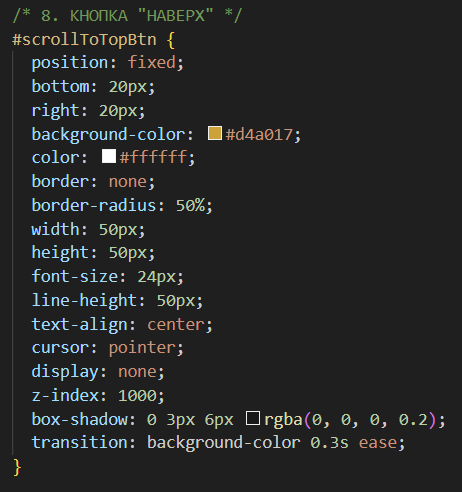


Рис. 2.23. CSS стили для кнопки «Наверх»

**#scrollToTopBtn**:

* Кнопка фиксируется в правом нижнем углу экрана.
* Имеет золотой фон, белый текст, круглую форму (ширина и высота 50px), с центровкой текста.
* Применяется курсор "рука" при наведении.
* Кнопка скрыта по умолчанию (display: none), отображается только при прокрутке страницы.
* Добавлен эффект тени и плавный переход фона при наведении.

**#scrollToTopBtn:hover**:

* При наведении на кнопку фон становится темнее (более темный золотой).

**Футер**

A computer screen shot of a black screen

AI-generated content may be incorrect.

Рис. 2.24. CSS стили для футера

Стили для футера:

* background-color: #d4a017 — золотой фон для футера.
* color: #ffffff — белый цвет текста в футере.
* text-align: center — выравнивание текста по центру.

**Страница «Блюда»**

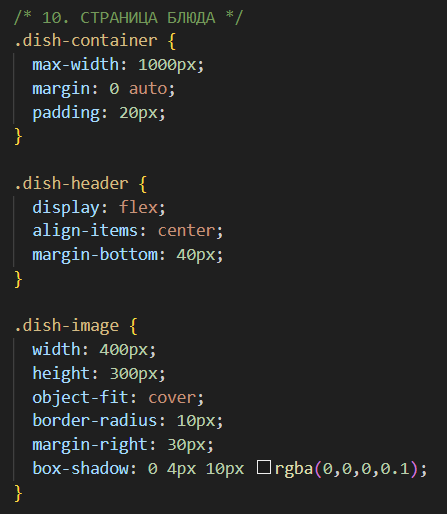
****

Рис. 2.25. CSS стили для контейнера и заголовков для страницы с блюдами

**.dish-container**:

* Контейнер с максимальной шириной 1000px, который центрируется на странице и имеет отступы по бокам и сверху/снизу (20px).

**.dish-header**:

* Заголовок блюда с использованием Flexbox для выравнивания содержимого по центру и отступом снизу (40px).

**.dish-image**:

* Изображение блюда размером 400x300px с обрезкой содержимого (object-fit: cover), скругленными углами (10px), отступом справа и тенью для создания объема.

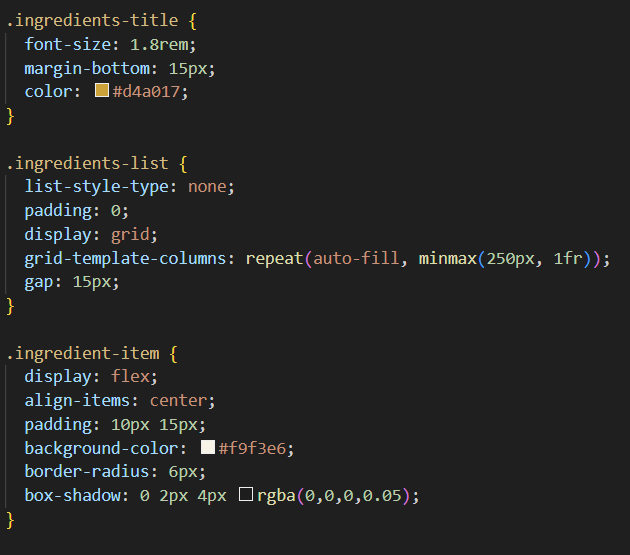


Рис. 2.26. CSS стили для раздела с ингредиентами блюд

**.ingredients-title**:

* Заголовок раздела: золотой цвет, крупный шрифт (1.8rem), отступ снизу (15px).

**.ingredients-list**:

* Список без маркеров, отображается в виде адаптивной сетки: каждая колонка не меньше 250px, с равномерными отступами между элементами (15px).

**.ingredient-item**:

* Каждый ингредиент оформлен как блок с флекс-выравниванием по центру, светлым фоном, скругленными углами и легкой тенью.



Рис. 2.27. CSS стили для кнопки «Назад»

**.back-button**:

* Кнопка отображается как строчный блок с отступом сверху (30px), внутренними отступами, золотым фоном и белым текстом.
* Без подчеркивания (text-decoration: none), скругленные углы, жирный шрифт.
* Добавлен плавный переход цвета фона при наведении.

**.back-button:hover**:

* При наведении фон становится темнее (тёмно-золотой цвет).

**Мобильная адаптивность**

A screen shot of a computer program

AI-generated content may be incorrect.

Рис. 2.28. Медиа-запросы

Медиа-запрос для адаптивности:

* При ширине экрана меньше 768px элементы навигации и меню становятся вертикальными (flex-direction: column), а карточки меню занимают всю ширину экрана.

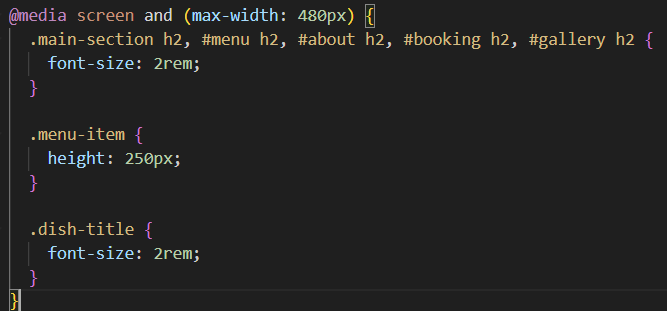


Рис. 2.29. Медиа-запрос для экрана с шириной 480 пикселей или меньше

Этот CSS-блок — **медиа-запрос**, который применяется, если ширина экрана **480 пикселей или меньше** (обычно смартфоны). Он делает сайт удобнее для маленьких экранов:

1. **.main-section h2, #menu h2, #about h2, #booking h2, #gallery h2**  
   — Заголовки в этих секциях уменьшаются до шрифта **2rem**, чтобы не занимать слишком много места.
2. **.menu-item**  
   — Высота каждого элемента меню уменьшается до **250px**, чтобы они лучше вписывались в компактный экран.
3. **.dish-title**  
   — Название блюда тоже становится **меньше (2rem)**, чтобы хорошо смотреться на мобильном устройстве.

## **2.4 JavaScript для функционала**

**Плавная прокрутка по якорным ссылкам в меню**

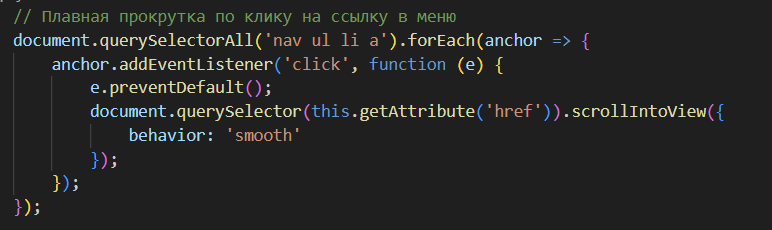


Рис. 2.30. Плавная прокрутка

**Поиск всех ссылок в меню**: document.querySelectorAll('nav ul li a') — этот код находит все элементы <a> (ссылки), которые находятся внутри <li>, которые в свою очередь находятся внутри элемента <ul> в навигационном меню <nav>.

**Добавление обработчика событий**: .forEach(anchor => { ... }) — для каждой найденной ссылки добавляется событие. Это означает, что для каждой ссылки в меню будет выполнен следующий код.

**Обработка клика по ссылке**: anchor.addEventListener('click', function (e) { ... }) — добавляется слушатель события на клик по ссылке. Когда пользователь кликает на ссылку, выполняется функция.

**Отмена стандартного поведения**: e.preventDefault() — эта строка отменяет стандартное действие браузера, то есть переход по ссылке. Вместо этого выполняется прокрутка.

**Плавная прокрутка**: document.querySelector(this.getAttribute('href')).scrollIntoView({ behavior: 'smooth' })

* this.getAttribute('href') получает значение атрибута href у ссылки, которое указывает на идентификатор (id) секции на странице.
* document.querySelector(...) находит элемент на странице, указанный в href.
* scrollIntoView({ behavior: 'smooth' }) заставляет страницу плавно прокрутиться до этого элемента, а параметр behavior: 'smooth' отвечает за плавность прокрутки.

**Проверка формы бронирования**



Рис. 2.31. Проверка формы бронирования столика

**Обработчик события на отправку формы**:

document.getElementById('bookingForm').addEventListener('submit', function(event) {...}) — добавляется обработчик события на форму с идентификатором bookingForm, который срабатывает при попытке отправить форму. Событие будет перехвачено и обработано внутри функции.

**Предотвращение стандартной отправки формы**:

event.preventDefault(); — предотвращает стандартное поведение формы, то есть отправку данных на сервер. Вместо этого форма будет обработана локально в браузере.

**Считывание значений полей формы**:

* let name = document.getElementById('name').value; — считывает значение поля с именем (по идентификатору name).
* let date = document.getElementById('date').value; — считывает дату бронирования.
* let time = document.getElementById('time').value; — считывает время бронирования.
* let people = document.getElementById('people').value; — считывает количество человек.

**Проверка заполненности всех полей**:

if (!name || !date || !time || !people) — проверка, чтобы все поля были заполнены. Если какое-либо из полей пустое, то:

* if (formMessage) formMessage.style.display = 'block'; — если элемент с идентификатором formMessage существует, то его стиль изменяется на видимый (появляется сообщение об ошибке).
* alert("Пожалуйста, заполните все поля."); — выводится предупреждение с просьбой заполнить все поля формы.
* return; — прерывает выполнение функции и не отправляет форму.

**Отправка успешного сообщения**:

* Если все поля заполнены, скрывается сообщение об ошибке (если оно есть):

if (formMessage) formMessage.style.display = 'none'; — скрывает сообщение об ошибке.

* alert("Спасибо за бронирование! Мы ждем вас " + date + " в " + time + "."); — выводит сообщение с подтверждением бронирования, где указывается выбранная дата и время.
* document.getElementById('bookingForm').reset(); — сбрасывает форму, очищая все поля после успешной отправки.

A computer screen shot of a program code

AI-generated content may be incorrect.

Рис. 2.32. Кнопка «Наверх»

**Получение элемента кнопки**:

* const mybutton = document.getElementById("scrollToTopBtn"); — находит элемент на странице с идентификатором scrollToTopBtn и сохраняет его в переменную mybutton.

**Обработчик события прокрутки страницы**:

* window.onscroll = function() {...} — добавляется обработчик события scroll на окно браузера. Этот обработчик срабатывает каждый раз, когда пользователь прокручивает страницу.

**Показ или скрытие кнопки в зависимости от прокрутки**:

if (document.body.scrollTop > 20 || document.documentElement.scrollTop > 20) — проверяется, насколько прокручена страница. Если прокрутка больше 20 пикселей (когда пользователь прокрутил страницу вниз), то кнопка становится видимой:

* mybutton.style.display = "block"; — делает кнопку видимой (устанавливает стиль display в значение block).

else { mybutton.style.display = "none"; } — если страница не прокручена более чем на 20 пикселей, то кнопка скрывается (стиль display изменяется на none).

**Добавление обработчика события для кнопки "наверх"**:

mybutton.addEventListener("click", function() {...}) — добавляется обработчик события на клик по кнопке. Когда пользователь нажимает на кнопку, выполняется следующая функция:

* window.scrollTo({ top: 0, behavior: "smooth" }); — выполняет плавную прокрутку страницы вверх (к верхней части страницы). Свойство behavior: "smooth" обеспечивает плавную анимацию.

# **3. РЕЗУЛЬТАТ**

В ходе разработки сайта для ресторанного проекта *"Gold Rush"* были успешно реализованы все ключевые функциональные элементы и страницы. Сайт соответствует элегантному итальянскому стилю, который мы выбрали для оформления. Использована цветовая палитра из бежевого, золотого, белого и оранжевого, что подчеркивает атмосферу изысканности и уюта.

**1. Главная страница**  
На главной странице сайта размещена приветственная информация о ресторане, а также навигационное меню с доступом к основным разделам. Главная страница выполнена в минималистичном дизайне с акцентом на изображения, что создает спокойную и приятную атмосферу.

**2. Страница меню**  
Меню сайта организовано в виде раздела, где посетители могут легко просматривать блюда с кратким описанием, фотографиями и ценами. Для удобства навигации меню разделено на категории, такие как «Основные блюда», «Десерты», «Напитки».

**3. Страница с описанием блюда**  
На странице с каждым блюдом подробно раскрываются его особенности. Примером такой реализации служит страница *«Паста Карбонара»*. Страница включает:

* Заголовок с названием блюда.
* Крупное и качественное изображение блюда.
* Цена и краткое описание блюда.
* Перечень основных ингредиентов.
* Кнопка навигации для возврата в меню.
* Футер с контактной информацией.

Такая реализация страниц блюд позволяет посетителям узнать подробности о каждом блюде, что способствует повышению интереса и вовлеченности.

**4. Страница о ресторане**  
Этот раздел посвящен истории ресторана, его концепции и миссии. Здесь размещены фотографии интерьера и общее описание атмосферы заведения. Сайт помогает пользователям почувствовать атмосферу ресторана даже до посещения.

**5. Страница бронирования**  
Для удобства клиентов была реализована форма онлайн-бронирования столика. Посетители могут выбрать дату, время и количество человек, после чего их запрос будет отправлен в систему ресторана для подтверждения. Эта функция помогает улучшить пользовательский опыт, обеспечивая удобный процесс бронирования.

**6. Галерея**  
Страница с фотографиями ресторана представляет собой фотогалерею, в которой отображаются изображения интерьера, атмосферы заведения, а также различные события и блюда. Это помогает посетителям получить визуальное представление о месте.

**7. Футер**  
На каждом экране присутствует футер, в котором отображается контактная информация, ссылки на страницы о ресторане, меню и страницу бронирования, а также социальные сети. Это помогает пользователю легко навигировать по сайту.

**8. Адаптивность и производительность**  
Сайт выполнен с учетом всех современных стандартов веб-разработки. Он адаптивен и корректно отображается на различных устройствах, таких как смартфоны, планшеты и компьютеры. Время загрузки страниц оптимизировано, что улучшает общую производительность и пользовательский опыт.

# **4. ЛИТЕРАТУРА**

1. Бенфорд, Б. HTML и CSS. Разработка и дизайн веб-сайтов. — СПб.: Питер, 2020.
2. Дакетт, Дж. JavaScript и jQuery. Интерактивная веб-разработка. — СПб.: Питер, 2021.
3. Никсон, Р. Современный JavaScript. Руководство для веб-разработчиков. — М.: Вильямс, 2022.
4. W3Schools — https://www.w3schools.com
5. Google Fonts — https://fonts.google.com — каталог шрифтов.
6. Freepik — https://freepik.com — источник изображений для фотогалереи.

**Ссылка на GitHub:** <https://github.com/emiliyamamaliyeva/hotel-restaurant>

**Ссылка на сайт:** <https://emiliyamamaliyeva.github.io/hotel-restaurant/>