**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](#_Toc196764045)

[**1.ТЕОРЕТИЧЕСКАЯ ЧАСТЬ** 7](#_Toc196764046)

[**1.1 Web-программирование** 7](#_Toc196764047)

[**1.2 HTML** 8](#_Toc196764048)

[**1.3 CSS** 12](#_Toc196764049)

[**1.4 JavaScript** 19](#_Toc196764050)

[**2.ПРАКТИЧЕСКАЯ ЧАСТЬ** 24](#_Toc196764051)

[**2.1 Описание предметной области** 24](#_Toc196764052)

[**2.2 Основные компоненты HTML-документа** 25](#_Toc196764053)

[**2.3 CSS стили для сайта** 29](#_Toc196764054)

[**2.4 JavaScript для функционала** 34](#_Toc196764055)

[**3. РЕЗУЛЬТАТ** 36](#_Toc196764056)

[**4. ЛИТЕРАТУРА** 38](#_Toc196764057)

# **Введение**

Веб-программирование — это процесс разработки веб-страниц и веб-приложений, которые доступны через интернет с использованием различных технологий, таких как HTML, CSS и JavaScript. Современный веб является неотъемлемой частью жизни людей, бизнес-процессов и образовательных систем. Веб-сайты и веб-приложения предоставляют платформы для взаимодействия пользователей с различными сервисами, товарами и услугами. Программирование для интернета предполагает создание динамичных и функциональных интерфейсов, которые могут адаптироваться под различные устройства и платформы.

Развитие веб-программирования тесно связано с историей интернета и технологий, которые использовались для его создания. За последние десятилетия веб-технологии значительно эволюционировали, и на сегодняшний день они предоставляют разработчикам широкие возможности для создания как простых статических страниц, так и сложных многослойных веб-приложений. Для того чтобы стать успешным веб-разработчиком, необходимо овладеть основами ключевых технологий, таких как HTML, CSS, JavaScript, а также освоить серверные технологии и базы данных.

Основные технологии веб-программирования

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

HTML является основным строительным блоком любого веб-сайта. Это язык разметки, который используется для создания структуры веб-страниц. HTML позволяет задавать заголовки, абзацы, списки, изображения, ссылки и другие элементы, которые составляют контент веб-страницы. Структура документа HTML организована в виде иерархической структуры, где каждый элемент имеет открывающий и закрывающий тег, а также может содержать атрибуты, которые уточняют его свойства.

С помощью HTML разработчики могут создавать каркас страницы, на который впоследствии можно наложить стили и добавить интерактивность. HTML не предназначен для создания сложных визуальных эффектов или обработки данных, однако без него невозможна базовая структура веб-страницы.

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

CSS используется для стилизации веб-страниц. Он позволяет задавать внешний вид элементов, созданных с помощью HTML, таких как цвет, шрифт, размер, отступы, расположение и другие параметры. С помощью CSS можно легко адаптировать веб-страницу под различные устройства, изменяя стиль в зависимости от разрешения экрана или других факторов, что делает сайт удобным для пользователей на мобильных телефонах, планшетах и компьютерах.

CSS предоставляет гибкие возможности для реализации дизайна веб-страниц, включая такие технологии как Flexbox и Grid для создания сложных макетов. Благодаря применению CSS можно существенно улучшить пользовательский интерфейс, повысить визуальное восприятие сайта и сделать его более удобным для взаимодействия.

1. **JavaScript**

JavaScript — это язык программирования, который используется для создания интерактивных элементов на веб-страницах. Он позволяет реагировать на действия пользователя, такие как клики, прокрутка или ввод данных, и изменять содержимое страницы без необходимости перезагружать её. Это позволяет создавать динамичные и отзывчивые веб-приложения, такие как формы, галереи, анимации и многое другое.

JavaScript играет важную роль в создании одностраничных приложений (SPA), где контент обновляется без перезагрузки страницы. С помощью JavaScript можно взаимодействовать с сервером через API, обрабатывать данные на клиенте и обновлять интерфейс. Современные библиотеки и фреймворки, такие как React, Angular и Vue.js, значительно упрощают разработку сложных веб-приложений, предоставляя разработчикам мощные инструменты для создания интерактивных пользовательских интерфейсов.

1. **Серверные технологии и базы данных**

Веб-программирование не ограничивается только клиентской частью. Для создания полноценных веб-приложений необходимы серверные технологии и базы данных. Серверная часть отвечает за обработку запросов от клиентов, хранение и извлечение данных, а также за взаимодействие с внешними системами. Среди популярных серверных языков программирования можно выделить PHP, Python, Ruby, Node.js и Java.

Для хранения данных на сервере используются базы данных, которые позволяют эффективно управлять информацией. Наиболее распространёнными типами баз данных являются реляционные базы данных (например, MySQL, PostgreSQL) и NoSQL базы данных (например, MongoDB). Базы данных являются важной частью большинства веб-приложений, так как они обеспечивают хранение данных пользователей, информации о товарах, заказах и других аспектов работы сайта.

Основные этапы разработки веб-приложения

Процесс создания веб-сайта или веб-приложения обычно состоит из нескольких ключевых этапов:

1. **Планирование и проектирование**. На этом этапе определяются цели и задачи веб-сайта, его целевая аудитория, структура и основные функции. Создаются каркасные наброски страниц (wireframes), которые показывают расположение элементов на странице. Важно на этом этапе обсудить требования к функционалу и дизайну с заказчиком или командой.
2. **Разработка структуры (HTML)**. Создаётся основная разметка сайта с помощью HTML. Это включает в себя создание всех необходимых страниц, добавление текстового контента, изображений и ссылок. Разработчики работают над тем, чтобы каждая страница была правильно структурирована и соответствовала требованиям к функционалу.
3. **Стилизация (CSS)**. После того как структура сайта готова, наступает этап стилизации. Используя CSS, разработчик настраивает визуальные элементы сайта: шрифты, цвета, отступы, размеры и расположение блоков. Важной задачей на этом этапе является создание адаптивного дизайна, который будет корректно отображаться на разных устройствах.
4. **Интерактивность (JavaScript)**. На этом этапе добавляется функциональность, такая как формы обратной связи, динамическое обновление контента, обработка пользовательских действий и взаимодействие с сервером. JavaScript используется для создания взаимодействий на странице без её перезагрузки, что делает сайт более удобным и быстрым.
5. **Тестирование и оптимизация**. Когда сайт готов, его нужно протестировать на разных устройствах и браузерах, чтобы убедиться, что он работает корректно. Также проводится оптимизация кода для повышения скорости загрузки страниц и улучшения производительности. Это может включать в себя сжатие изображений, уменьшение размера файлов CSS и JavaScript, а также использование кэширования.
6. **Размещение и поддержка**. После того как сайт прошёл тестирование, его размещают на сервере, и он становится доступным для пользователей. Важно помнить, что разработка сайта не заканчивается на этом этапе. Необходимо регулярно обновлять и поддерживать сайт, добавлять новые функции и исправлять ошибки.

# **1.ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

## **1.1 Web-программирование**

Веб-программирование — это создание веб-сайтов и веб-приложений, доступных через интернет с помощью различных технологий. Современные веб-страницы и приложения не ограничиваются только текстом и изображениями, они становятся динамичными, интерактивными и адаптивными. Основные технологии веб-разработки включают HTML, CSS и JavaScript, а также серверные языки и базы данных.

**Основные компоненты веб-разработки**

1. **Frontend (клиентская часть)**

Frontend-разработка охватывает создание внешнего вида и интерфейса сайта. Используя HTML, CSS и JavaScript, фронтенд-разработчики создают структуру, стили и динамичные элементы, такие как анимации, формы и обновление контента без перезагрузки страницы. Важным аспектом является создание адаптивных сайтов, которые корректно отображаются на разных устройствах.

1. **Backend (серверная часть)**

Серверная часть веб-разработки управляет запросами от пользователей, выполняет обработку данных и взаимодействует с базами данных. Для серверной логики используются такие языки, как Python, PHP, Java и Node.js. Сервер отвечает за отправку данных на клиентскую часть и обработку бизнес-логики.

1. **Работа с базами данных**

Базы данных являются основой для хранения информации в веб-приложениях, таких как данные пользователей или товары в интернет-магазине. Разработчики используют реляционные (SQL) и нереляционные (NoSQL) базы данных, в зависимости от типа данных и требований приложения. SQL-запросы помогают извлекать, обновлять и удалять данные.

Современные веб-приложения часто используют технологии, такие как AJAX и фреймворки вроде React или Angular, для создания динамичных интерфейсов. Это позволяет обновлять контент на странице без её перезагрузки, что повышает производительность и удобство взаимодействия с пользователем. Одностраничные приложения (SPA) становятся всё более популярными, обеспечивая более плавный опыт.

С развитием мобильных устройств важным аспектом веб-разработки стало создание адаптивных сайтов. Сайты должны подстраиваться под различные экраны и устройства, обеспечивая комфортное взаимодействие для пользователей. Важно также учитывать доступность, создавая сайты, которые могут использовать люди с ограниченными возможностями, например, с помощью специальных технологий для управления интерфейсом.

## **1.2 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.

Пример базовой структуры HTML-страницы:

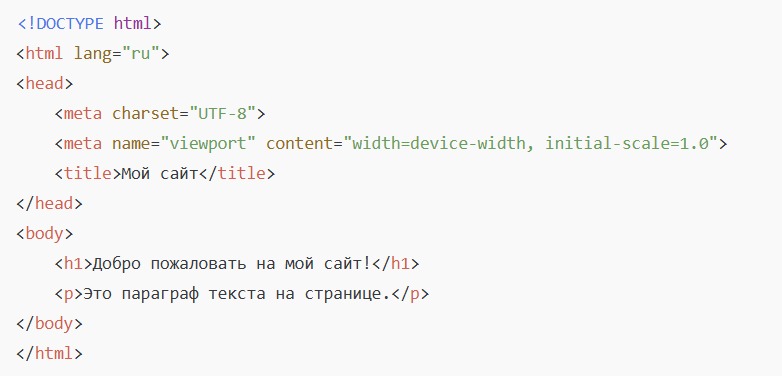


Рис. 1.2.1

**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.
* **<blockquote>**: Цитата, которая обычно выделяется отступами с обеих сторон.
* **<a>**: Тег для создания гиперссылок. Содержит атрибут href, который указывает URL.

Пример использования тегов для текста:

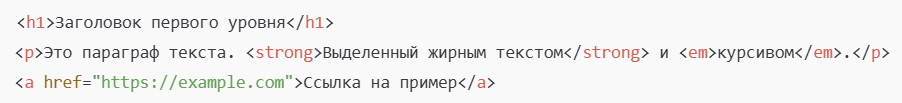


Рис. 1.2.2

**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.

Пример вставки изображения и видео:

A close-up of text

AI-generated content may be incorrect.

Рис. 1.2.3

**Списки**

HTML позволяет создавать два типа списков:

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

Пример создания списка:

A screenshot of a computer program

AI-generated content may be incorrect.

Рис. 1.2.4

**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.

Пример таблицы:

A screenshot of a computer code

AI-generated content may be incorrect.

Рис. 1.2.5

**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.

Пример формы:

A computer code with text

AI-generated content may be incorrect.

Рис. 1.2.6

**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.

Пример с атрибутами:

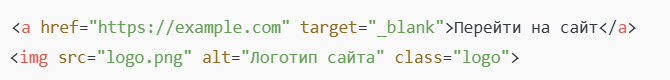


Рис. 1.2.7

**Важные моменты при работе с HTML**

1. **Валидация HTML**: Важно использовать правильный синтаксис HTML, чтобы гарантировать, что страницы отображаются корректно в разных браузерах. Для этого можно использовать валидаторы HTML, такие как W3C HTML Validator.
2. **Семантический HTML**: Семантические элементы (такие как <header>, <footer>, <article>, <section>) делают код более понятным и доступным для людей и поисковых систем. Они улучшают SEO (поисковую оптимизацию) и помогают структурировать контент.
3. **Мобильная адаптивность**: Важно, чтобы сайты были адаптированы под различные устройства. Это можно достичь с помощью адаптивного дизайна, используя медиа-запросы в CSS.
4. **SEO (поисковая оптимизация)**: Использование правильных тегов и атрибутов в HTML помогает улучшить видимость сайта в поисковых системах. Например, правильное использование заголовков <h1>, <h2>, мета-тегов и атрибутов alt для изображений повышает SEO.

## **1.3 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.

Пример CSS-кода:



Рис. 1.3.1

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**: Применение стилей непосредственно в атрибуте элемента HTML.



Рис. 1.3.2

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

**2.Встроенные стили**: Вставка CSS в секцию <head> HTML-документа с использованием тега <style>.

A computer code with text

AI-generated content may be incorrect.

Рис. 1.3.3

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. 1.3.4

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. 1.3.5

**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. 1.3.6

Применение к элементу:



Рис. 1.3.7

**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. 1.3.8

Применение к элементу:



Рис. 1.3.9

**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. 1.3.10

**5.Селекторы потомков и детей**: 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. 1.3.11

**Свойства CSS**

CSS предоставляет огромное количество свойств, которые позволяют контролировать визуальное отображение элементов. Рассмотрим основные из них:

**1.Цвет и фон**:

**color**: Определяет цвет текста.

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

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

Пример:

A close-up of a computer code

AI-generated content may be incorrect.

Рис. 1.3.12

**2.Шрифт**:

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

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

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

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

Пример:

A screenshot of a computer code

AI-generated content may be incorrect.

Рис. 1.3.13

**3.Отступы и поля**:

**margin**: Определяет внешние отступы элемента.

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

Пример:

A close-up of a computer code

AI-generated content may be incorrect.

Рис. 1.3.14

**4.Границы**:

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

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

Пример:

A close-up of a computer code

AI-generated content may be incorrect.

Рис. 1.3.15

**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.

Пример:

A screen shot of a computer code

AI-generated content may be incorrect.

Рис. 1.3.16

**6.Тени и трансформации**:

* + **box-shadow**: Добавляет тень вокруг элемента.
  + **text-shadow**: Добавляет тень к тексту.
  + **transform**: Позволяет изменять положение, размер и другие характеристики элемента.

Пример:

A screenshot of a computer code

AI-generated content may be incorrect.

Рис. 1.3.17

**Адаптивный дизайн и медиа-запросы**

Одной из главных задач современного веб-дизайна является создание сайтов, которые корректно отображаются на различных устройствах (ПК, планшетах, мобильных телефонах). Для этого используется подход адаптивного дизайна (responsive design), 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.

Пример медиа-запроса:

A screenshot of a computer code

AI-generated content may be incorrect.

Рис. 1.3.18

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

**Преимущества использования CSS**

1. **Отделение структуры от дизайна**: CSS позволяет разделить структуру HTML-документа от его визуального оформления. Это облегчает работу с кодом и его поддержку.
2. **Управление стилями для нескольких страниц**: С помощью одного внешнего файла CSS можно применить стили ко всем страницам сайта, что упрощает изменения и обновления.
3. **Гибкость и мощность**: CSS предоставляет множество свойств для детальной настройки внешнего вида элементов, включая анимации, эффекты и адаптивный дизайн.
4. **Оптимизация скорости**: Использование CSS позволяет снизить объем HTML-кода и уменьшить загрузочные файлы.

## **1.4 JavaScript**

**JavaScript** — это один из самых популярных языков программирования, используемый для создания динамичных и интерактивных веб-страниц. Он был разработан в 1995 году Бренданом Айхом для браузера Netscape Navigator и с тех пор стал неотъемлемой частью веб-разработки. 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. Он может работать с элементами DOM (Document Object Model) для изменения содержания и структуры страницы в реальном времени. JavaScript также используется для обработки событий, например, кликов, ввода текста и других действий пользователя.

**Пример:**



Рис. 1.4.1

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.

**Пример:**

A computer screen shot of a computer code

AI-generated content may be incorrect.

Рис. 1.4.2

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.

**Пример:**

A computer code with different colored text

AI-generated content may be incorrect.

Рис. 1.4.3

Этот код проверяет возраст пользователя и выводит сообщение в зависимости от результата проверки.

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

**Пример:**

A screenshot of a computer code

AI-generated content may be incorrect.

Рис. 1.4.4

Здесь цикл перебирает все числа в массиве numbers и выводит их в консоль.

**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.

**Пример:**

A close-up of a computer code

AI-generated content may be incorrect.

Рис. 1.4.5

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

**Асинхронность в JavaScript**

JavaScript поддерживает асинхронное выполнение кода, что позволяет не блокировать интерфейс пользователя. Для работы с асинхронными операциями в JavaScript используются **callback-функции**, **Promises** и **async/await**.

**Пример с callback:**

A screen shot of a computer code

AI-generated content may be incorrect.

Рис. 1.4.6

Здесь функция fetchData имитирует задержку в 2 секунды, а затем вызывает переданную ей функцию-обработчик, чтобы передать результат.

**Пример с Promise:**

A computer code with colorful text

AI-generated content may be incorrect.

Рис. 1.4.7

В этом примере создается объект Promise, который в зависимости от состояния выполняет одну из двух функций: resolve или reject.

**Пример с async/await:**

A screen shot of a computer code

AI-generated content may be incorrect.

Рис. 1.4.8

С помощью async/await можно писать асинхронный код в синхронном стиле, что делает его более читаемым и удобным для работы с промисами.

**Работа с DOM**

DOM (Document Object Model) — это интерфейс для работы с HTML-документами, который позволяет манипулировать содержимым и структурой веб-страницы. JavaScript использует DOM для добавления, удаления и изменения элементов на странице в реальном времени.

Для поиска элементов на странице и работы с ними можно использовать методы getElementById, querySelector и другие.

**Пример:**

A close-up of a computer code

AI-generated content may be incorrect.

Рис. 1.4.9

В этом примере при нажатии на кнопку с id myButton изменяется текст элемента с id myText.

**Модули и ООП в JavaScript**

JavaScript поддерживает объектно-ориентированное программирование (ООП). Это позволяет создавать сложные структуры, использовать наследование и инкапсуляцию, а также улучшать читаемость и повторное использование кода.

**Пример с классами:**

A screen shot of a computer code

AI-generated content may be incorrect.

Рис. 1.4.10

Здесь создается класс Car, который имеет метод displayInfo, выводящий информацию о машине.

# **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. В современном цифровом мире наличие качественного веб-сайта становится необходимым условием для успешного взаимодействия с клиентами, повышения узнаваемости и уровня доверия.

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.
* Упрощает навигацию и взаимодействие с потенциальными клиентами за счёт удобного интерфейса и адаптивного дизайна.

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**, что позволяет реализовать как статические, так и динамические элементы: анимации, формы и пользовательские сценарии (например, плавная прокрутка, интерактивная кнопка возврата вверх, форма валидации). При этом особое внимание уделяется адаптивности сайта и эстетике интерфейса, соответствующей визуальному стилю ресторана.

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

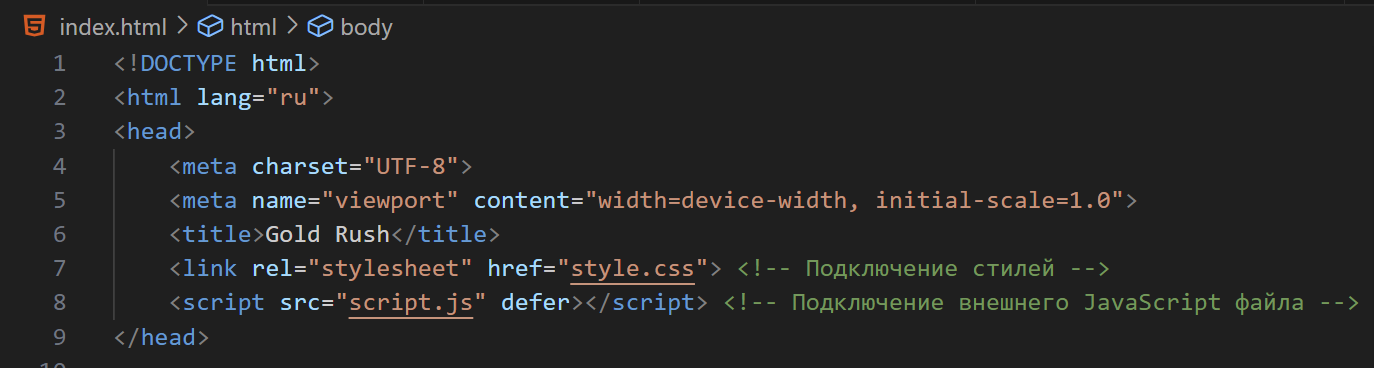


Рис. 2.2.1

**1.1. <!DOCTYPE html>**

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

**1.2. <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.

**1.3. <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. 2.2.2

**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**

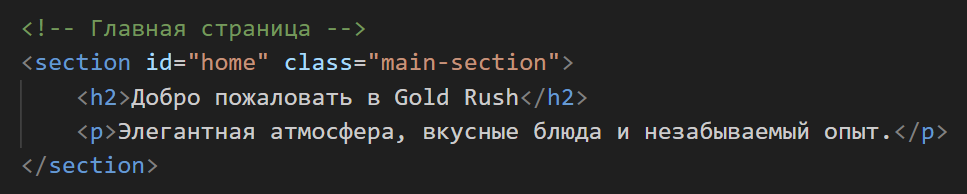
****

Рис. 2.2.3

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. 2.2.4

Раздел меню включает в себя список блюд, оформленных как карточки. Каждая карточка блюда (<div class="menu-item">) содержит название блюда и его стоимость. Также используется inline-стиль для установки фона, чтобы визуально представить каждый элемент меню через изображения.

**3.О ресторане**

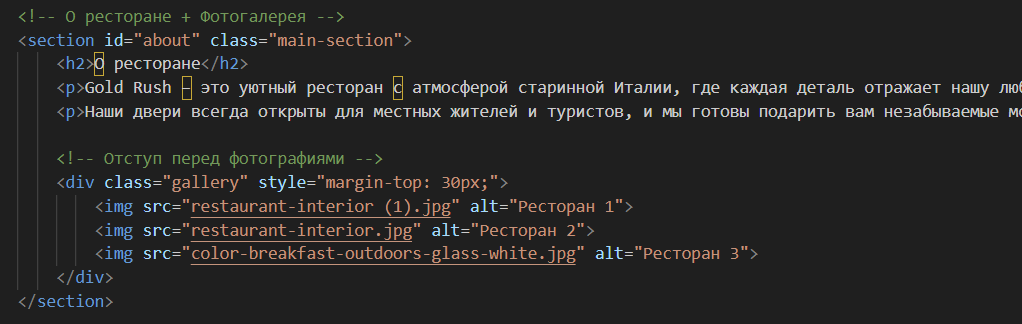


Рис. 2.2.5

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 (важно для SEO и доступности).

**4. Форма бронирования**

A computer screen with text

AI-generated content may be incorrect.

Рис. 2.2.6

Раздел с формой бронирования включает поля для ввода имени, даты, времени и количества человек. Использование тега <form> позволяет собирать информацию от пользователя и отправлять её на сервер. Атрибут required на элементах формы указывает на обязательность ввода этих данных.

**Футер**

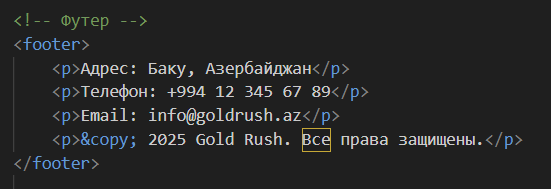
****

Рис. 2.2.7

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. 2.2.8

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.

## **2.3 CSS стили для сайта**

**Общие стили для всего сайта**

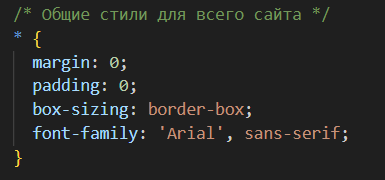


Рис. 2.3.1

Этот блок задает глобальные стили для всех элементов на странице:

* margin: 0 и padding: 0 сбрасывают отступы и внутренние отступы у всех элементов, что помогает контролировать их внешний вид.
* box-sizing: border-box изменяет способ расчета ширины и высоты элементов, включая padding и border в общую ширину/высоту.
* font-family: 'Arial', sans-serif; задает шрифт для всех элементов, использующих стандартный шрифт Arial, если его нет — будет использоваться любой другой шрифт без засечек.

**Основной фон страницы**

A black background with white and green text

AI-generated content may be incorrect.

Рис. 2.3.2

Этот стиль определяет фон и текст для всей страницы:

* background-color: #f9f3e6 — задает бежевый фон для всей страницы.
* color: #4a4a4a — устанавливает темно-серый цвет текста.

**Заголовки**

A computer screen shot of a black background with white text

AI-generated content may be incorrect.

Рис. 2.3.3

Здесь стилизуются заголовки:

* Все заголовки (h1, h2, h3) будут золотыми (color: #d4a017).
* Устанавливается жирный шрифт (font-weight: 700).
* Используется шрифт с засечками (font-family: 'Georgia', serif), чтобы создать более элегантный вид заголовков.

**Навигационное меню**

**A computer screen with text

AI-generated content may be incorrect.**

Рис. 2.3.4

Здесь стилизуется шапка с навигационным меню:

* background-color: #fffaf0 — светло-бежевый фон для шапки.
* padding: 20px 0 — отступы сверху и снизу.
* box-shadow — добавляет мягкую тень, чтобы выделить шапку.
* position: sticky — фиксирует шапку в верхней части экрана при прокрутке страницы.
* top: 0 — закрепляет шапку на верхней границе страницы.
* z-index: 1000 — обеспечивает, чтобы шапка всегда была поверх других элементов.

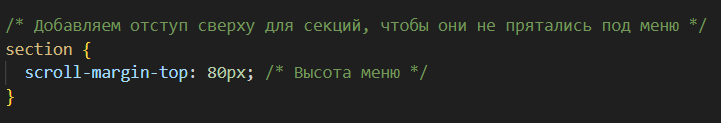


Рис. 2.3.5

Добавление отступа сверху для всех секций, чтобы они не прятались под фиксированным меню при переходе по ссылкам в меню.

A screen shot of a computer code

AI-generated content may be incorrect.

Рис. 2.3.6

Стили для контейнера навигации:

* display: flex — применяет Flexbox для размещения элементов по горизонтали.
* justify-content: space-between — распределяет элементы так, чтобы они занимали максимально возможное пространство.
* align-items: center — выравнивает элементы по вертикали по центру.
* max-width: 1200px — ограничивает ширину контейнера.
* margin: 0 auto — центрирует контейнер на странице.
* padding: 0 20px — добавляет отступы слева и справа.

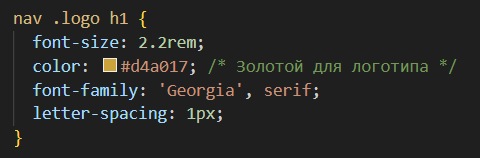


Рис. 2.3.7

Стили для логотипа:

* Устанавливается размер шрифта, золотой цвет, шрифт с засечками и немного увеличено межбуквенное расстояние (letter-spacing).

**Секция "Меню"**

A screen shot of a computer program

AI-generated content may be incorrect.

Рис. 2.3.8

Стили для элементов меню:

* position: relative — позволяет добавлять псевдоэлементы и управлять позиционированием.
* background-size: cover и background-position: center — заставляют фоновое изображение покрывать элемент и быть центрированным.
* display: flex и flex-direction: column — устанавливают вертикальное расположение содержимого.
* box-shadow — добавляет тень для элементов меню.
* text-align: center — выравнивает текст по центру.

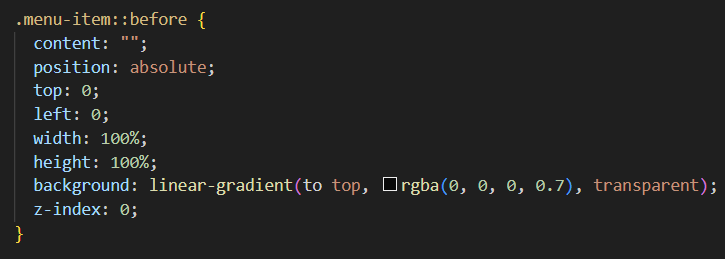


Рис. 2.3.9

Этот псевдоэлемент добавляет темное затемнение на фон элементов меню с помощью линейного градиента.

**Секция "О ресторане"**

A screen shot of a computer program

AI-generated content may be incorrect.

Рис. 2.3.10

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. 2.3.11

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.

**Футер**

A computer screen shot of a black screen

AI-generated content may be incorrect.

Рис. 2.3.12

Стили для футера:

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

**Мобильная адаптивность**

A screen shot of a computer program

AI-generated content may be incorrect.

Рис. 2.3.13

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.

## **2.4 JavaScript для функционала**

**Плавная прокрутка по якорным ссылкам в меню**

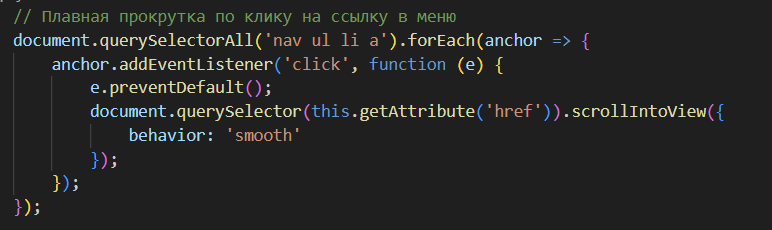


Рис. 2.4.1

* Сначала выбираются все ссылки в навигации (nav ul li a).
* Каждой ссылке добавляется слушатель событий.
* При клике прокрутка происходит плавно к соответствующему разделу (href), а не мгновенно.

**Проверка формы бронирования (валидация)**



Рис. 2.4.2

* Обработчик submit отменяет стандартное поведение отправки.
* Проверяет, чтобы все поля были заполнены.
* Если да — выводится сообщение об успешной брони.
* После этого форма очищается.

A screen shot of a computer code

AI-generated content may be incorrect.

Рис. 2.4.3

* При прокрутке вниз более чем на 20px — кнопка становится видимой.
* При возврате вверх — скрывается.
* При нажатии на кнопку — страница плавно прокручивается в начало.

# **3. РЕЗУЛЬТАТ**

**Секция «Главная страница»**

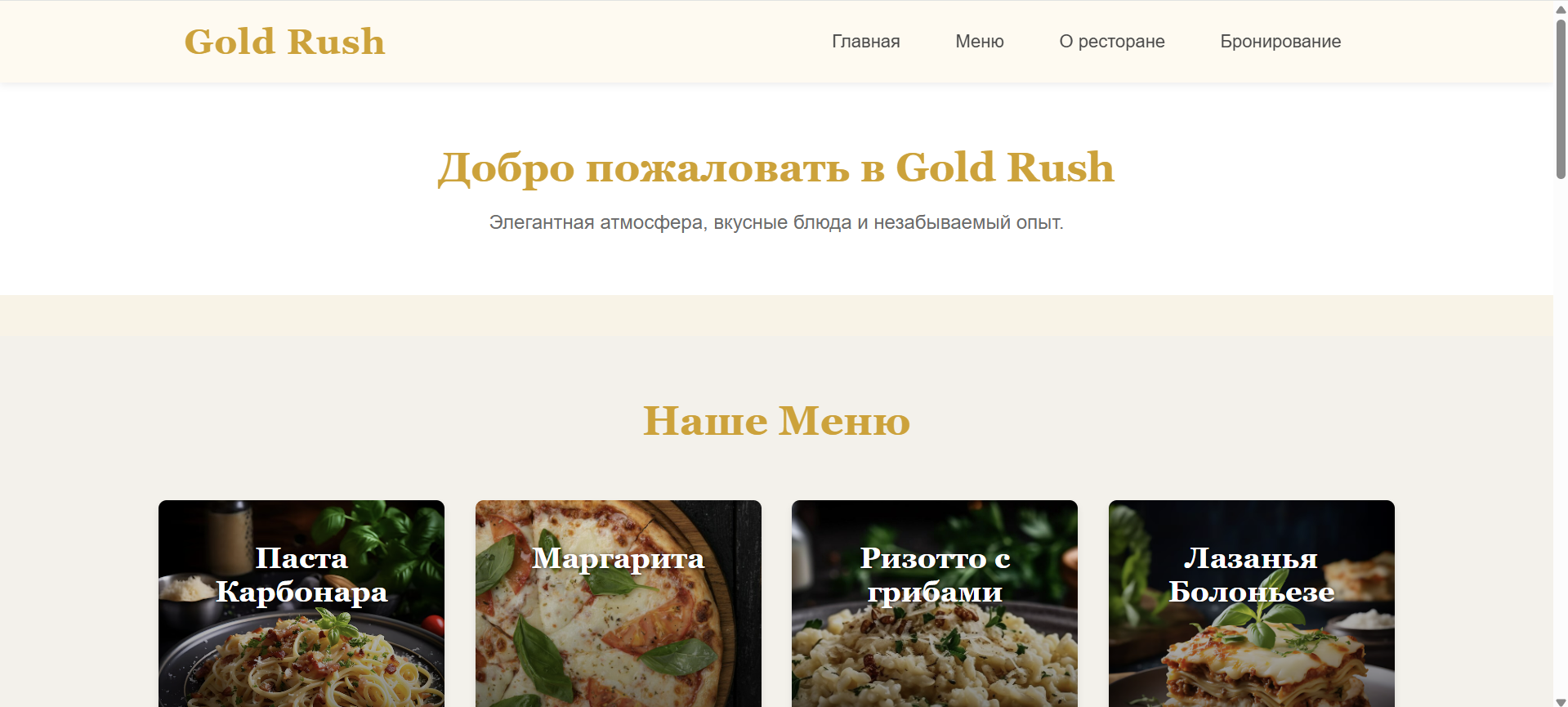


Рис. 3.1

**Секция «Наше Меню»**

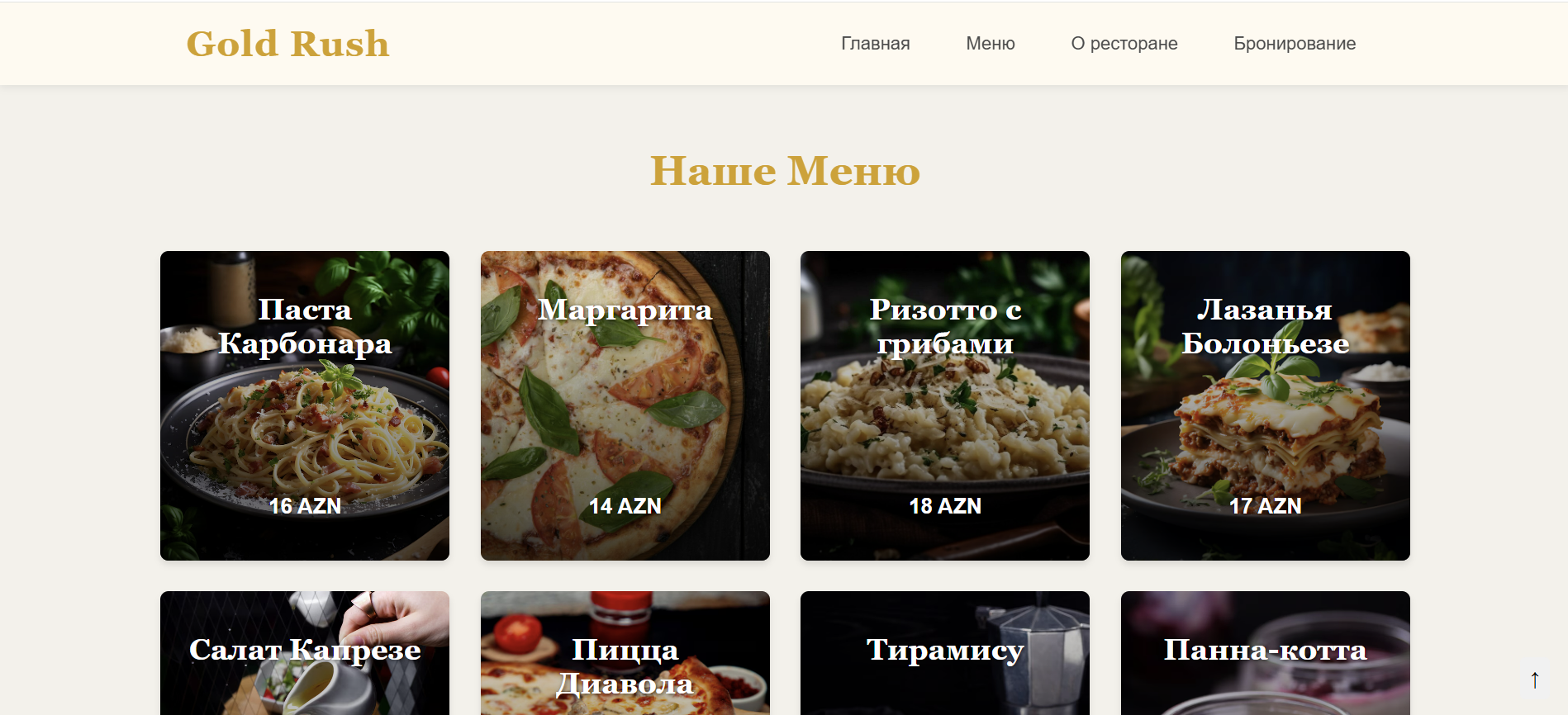


Рис. 3.2

**Секция «О ресторане»**

A screenshot of a restaurant

AI-generated content may be incorrect.

Рис. 3.3

**Секция «Бронирование**

A screenshot of a computer

AI-generated content may be incorrect.

Рис. 3.4

**Секция «Футер»**

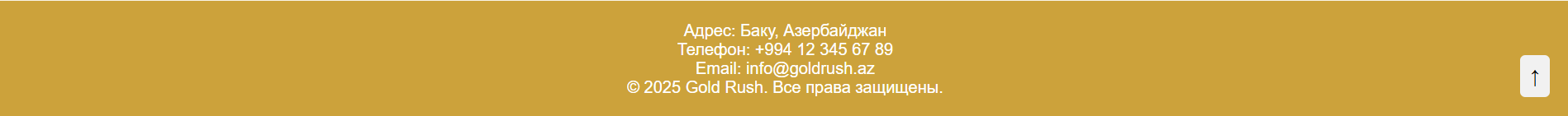


Рис. 3.5

# **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. Мэйер, Э. CSS. Каскадные таблицы стилей. — СПб.: Диалектика, 2019.
4. Никсон, Р. Современный JavaScript. Руководство для веб-разработчиков. — М.: Вильямс, 2022.
5. Коуэн, Дж. Адаптивный веб-дизайн. Принципы и практика. — СПб.: БХВ-Петербург, 2020.
6. W3Schools — https://www.w3schools.com
7. Google Fonts — https://fonts.google.com — каталог шрифтов.
8. 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/>