Geekbrains

**Разработка современного одностраничного веб-приложения для малого бизнеса на JavaScript с использованием React.**

Программа: Разработчик

Специализация: Frontend-разработка (React)

ФИО: Захарова Ольга Александровна

Санкт-Петербург

2024

**Содержание**

Введение …………………………………………………………….……………... 4

Глава 1. Основы современных веб-сайтов……………………………….………. 6

1.1. История и эволюция веб-сайтов: от первых макетов до современных решений ……………………………………………………………………….…… 6

1.2. Веб-сайт: определение, виды и особенности в условиях современности... 10

1.3. Основные элементы современного веб-сайта ………………………….….. 13

1.4. Основные сетевые компоненты и протоколы для функционирования веб-сайтов ……………………………………………………………...……...…….… 16

Глава 2. Современные инструменты для разработки веб-сайтов …...……....... 23

2.1. Обзор актуальных инструментов разработки ……………………….….…. 23

2.2. HTML и его место в разработке веб-сайтов ……………………………..… 27

2.3. CSS как основа оформления в современных проектах ………………....… 32

2.4. Введение в JavaScript: возможности и роль в веб-разработке ………….…39

2.5. Сравнение популярных JavaScript-фреймворков: Angular, Vue, React ..….45

Глава 3. Разработка одностраничного коммерческого веб-сайта для малого бизнеса …………………………………………………………………….……....51

3.1. Проектирование и создание HTML-разметки и ее стилизация с помощью CSS и препроцессоров ……………………………………………….………...…51

3.2. Добавление интерактивности с помощью JavaScript ………….………..…55

3.3. Перевод проекта на React: архитектура и компоненты ….………….…… 57

3.4. Реализация функционала переключения светлой/темной темы ……….... 60

Заключение ……………………………………………………………….………62

Список используемой литературы …………………………………….………..64

Приложение 1 ………………………………………………………………...…..66

Приложение 2 …………………………………………………………….……… 72

Приложение 3 ……………………………………………………………………. 91

Приложение 4 ………………………………………………………………..…. 105

Приложение 5 ……………………………………………………………….….. 115

Приложение 6 ……………………………………………………………….….. 116

Введение.

В современном мире веб-сайты являются неотъемлемой и немаловажной частью бизнеса. Компании, стремящиеся повысить свою конкурентоспособность, узнаваемость и эффективность, а также улучшить пользовательский опыт, активно внедряют информационные системы, которые способствуют автоматизации бизнес-процессов и улучшению взаимодействия с клиентами.

Одним из таких решений является разработка сайта, который может стать для предприятия простым и удобным инструментом для управления отношениями с клиентами. При этом на сегодняшний день на рынке представлен широкий перечень современных инструментов, используемых в веб-разработке, что способствует удовлетворению потребностей любых компаний - как крупных, так и мелких.

Одностраничные веб-приложения (Single Page Applications, SPA) занимают особое место среди современных решений, предоставляя пользователям более быстрый и интерактивный опыт за счет минимизации перезагрузок страниц и оптимизации работы с данными.

Целью данной дипломной работы является разработка современного одностраничного веб-приложения для малого бизнеса, а именно его frontend-части, с использованием JavaScript и React. Для достижения этой цели необходимо решить следующие задачи:

1. Изучить основы разработки современных веб-сайтов, их виды и особенности.
2. Провести анализ актуальных инструментов и технологий для веб-разработки.
3. Исследовать возможности JavaScript и сравнить популярные JavaScript-фреймворки.
4. Разработать HTML-разметку и стилизацию с помощью CSS для веб-приложения.
5. Реализовать интерактивный функционал с использованием JavaScript.
6. Перевести проект на фреймворк React и добавить функционал переключения светлой/темной темы.

Для достижения поставленных целей в работе будет использоваться ряд методов, включая анализ литературы и существующих решений, разработку прототипов и программирование. Работа состоит из трех глав, каждая из которых посвящена различным аспектам разработки одностраничного веб-приложения для малого бизнеса:

1. В Главе 1 рассматриваются понятие, виды и особенности веб-сайтов, их эволюция и основные элементы.
2. В Главе 2 анализируются актуальные инструменты и технологии, возможности JavaScript и сравнение популярных фреймворков, с акцентом на React.
3. В Главе 3 описываются этапы разработки веб-приложения, начиная с проектирования HTML-разметки и стилизации, и заканчивая реализацией интерактивности, переводом на React и добавлением функционала переключения тем.

В условиях высокой конкуренции, малые предприятия действительно нуждаются в доступных и эффективных решениях для максимизации прибыли, которые не потребуют большое количество денежных и временных ресурсов, а также будут просты в использовании и поддержании. Не менее важным является и расширяемость кода, что позволит подстраиваться сайту под растущие потребности и возможности компании, а не наоборот. Именно это и делает тему данной дипломной работы особенно актуальной, поскольку рассматриваемые технологии отвечают всем данным требованиям.

Таким образом, данная дипломная работа представляет собой комплексное исследование, направленное на исследование, анализ и разработку эффективного и современного веб-приложения, которое может стать полезным инструментом для малого бизнеса в условиях современных реалий.

Глава 1. Основы современных веб-сайтов.

1.1. История и эволюция веб-сайтов: от первых макетов до современных решений.

Пользовательский опыт и формирование определенных ожиданий от веб-сайтов формировались постепенно и росли вместе с развитием веб-технологий. Эволюция веб-сайтов началась с простых текстовых страниц и привела к сложным интерактивным платформам, которые используются сегодня. История развития веб-сайтов охватывает несколько этапов, каждый из которых привносил новые технологии и подходы к созданию и использованию веб-ресурсов.

На заре своего существования веб-сайты разрабатывались с целью упрощения обмена научной информацией. Ни о каком дизайне говорить не приходилось. Более того в те времена веб-дизайн и верстка как таковые не существовали, и все сайты выглядели одинаково, что можно назвать «академическим дизайном». Первые веб-сайты были статическими, состояли только из HTML-разметки и отображали исключительно текст и изображения.

Такие сайты были просты в разработке и использовали минимальное количество технологий. Они не предусматривали интерактивности и обновления контента без участия разработчика. Примеры таких сайтов можно увидеть в архиве первых версий World Wide Web, где размещались научные статьи, новости и другая статическая информация, а также на lib.ru (Иллюстрация 1).

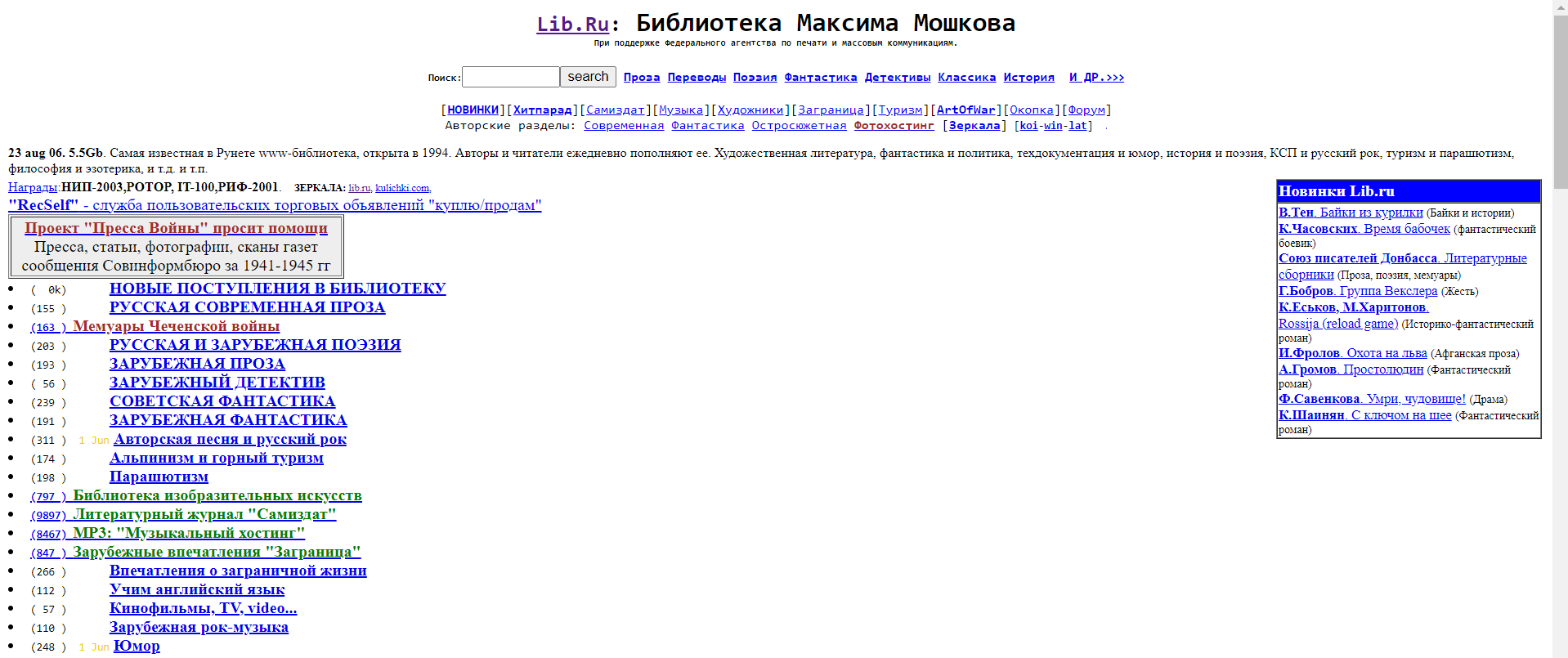


Иллюстрация 1. Пример сайта без стилизации.

Со временем пользователи, которых становилось все больше, начали отдавать предпочтение более визуально привлекательным сайтам. Браузер Netscape, доминировавший на рынке в 90-е годы, начал добавлять в HTML новые теги, позволяющие улучшить внешний вид веб-страниц. Эти теги не были стандартизированы и работали только в Netscape, но, учитывая его доминирующую долю рынка, об этом еще никто не задумывался.

Одним из ключевых моментов в эволюции веб-сайтов стало появление спецификации CSS (Cascading Style Sheets). Изначально браузеры не полностью поддерживали CSS, и разработчики использовали таблицы для макетов. Это был стандарт де-факто в то время. Однако с развитием CSS и появлением версии CSS2, а затем и CSS 2.1, стало возможным отказаться от табличной верстки в пользу блочной верстки, где расположение элементов и их стиль задаются через CSS.

Введение визуальных элементов в HTML привело к постепенному уходу от «академического дизайна» и стало толчком к развитию веб-дизайна. Разработчики сайтов поняли, что дизайн — это не просто разноцветный текст и случайные картинки, а способ эффективно представить материал, создать определенное настроение и удержать внимание посетителей. Дизайн стал неотъемлемой частью процесса разработки веб-сайтов, за которым следовала верстка. Нарисованные в графических редакторах макеты нужно было превратить в набор HTML- и CSS-файлов, которые могли бы быстро загружаться и сохранять особенности дизайна.

Эра доминирования Netscape закончилась, когда на рынок вышел браузер Internet Explorer от Microsoft, захватив значительную часть аудитории. Это создало дополнительные сложности для разработчиков из-за различий в интерпретации HTML и CSS в разных браузерах, что требовало дополнительного времени на тестирование и отладку сайтов.

С развитием технологий, таких как CGI (Common Gateway Interface), PHP (Hypertext Preprocessor) и ASP (Active Server Pages), веб-сайты стали динамическими. Это означало, что контент мог генерироваться на лету, основываясь на запросах пользователей и данных из баз данных. Динамические сайты позволяли создавать более сложные и интерактивные интерфейсы, такие как форумы, блоги и интернет-магазины. Примеры динамических сайтов включают ранние версии Amazon и eBay, которые использовали серверные скрипты для управления товарами и заказами.

Следующим витком развития разработки сайтов стала эпоха Web 2.0, которая принесла социальные сети, блоги и другие платформы, позволяющие пользователям взаимодействовать и создавать контент. В этот период начали широко использоваться JavaScript и AJAX (Asynchronous JavaScript and XML) для улучшения интерактивности и пользовательского опыта. Web 2.0 также привнес концепцию пользовательского контента, где пользователи могли создавать и редактировать информацию на сайтах, что способствовало росту социальных сетей и блогов. Примеры таких сайтов включают YouTube и VK.

Современные веб-сайты все чаще разрабатываются как одностраничные приложения (SPA), используя JavaScript-фреймворки, такие как React, Angular и Vue. SPA загружают контент динамически по мере необходимости без полной перезагрузки страницы, что значительно улучшает пользовательский опыт. Одностраничные приложения позволяют создавать более быстрые и интерактивные веб-сайты, которые реагируют на действия пользователя мгновенно. Ярким примером таких сайтов является Яндекс Почта.

Исходя из вышесказанного, можно сделать вывод о том, что развитие технологий и стандартов тесно связанно с развитием веб-сайтов. Важные этапы эволюции веб-сайтов таковы:

1. HTML5 и CSS3: Введение новых элементов и возможностей для создания более сложных и интерактивных веб-страниц.
2. JavaScript и фреймворки: Развитие JavaScript и появление популярных фреймворков, таких как React, Angular и Vue, которые облегчают создание динамических и интерактивных веб-приложений.
3. API и микросервисы: Использование API (Application Programming Interface) и микросервисной архитектуры для интеграции различных сервисов и улучшения функциональности веб-сайтов.
4. Облачные технологии: Применение облачных технологий для хостинга и масштабирования веб-приложений, что позволяет улучшить производительность и доступность.

Таким образом, эволюция веб-сайтов прошла долгий путь от простых однотипных страниц до сложных, визуально насыщенных и функциональных веб-приложений. Этот процесс включал в себя переход от акцента на структуре информации к акценту на её визуальном представлении и пользовательском опыте. Современные веб-сайты используют передовые технологии и подходы для обеспечения высокой интерактивности, производительности и удобства использования.

1.2. Веб-сайт: определение, виды и особенности в условиях современности.

В современном мире веб-сайты являются неотъемлемой частью бизнеса, предоставляя компании отличную платформу для различных целей, таких как представление продуктов и услуг, взаимодействие с клиентами, маркетинг и продажи. Развитие технологий и изменение пользовательских ожиданий диктуют новые требования к веб-сайтам, которые должны быть функциональными, удобными, быстрыми и визуально привлекательными.

Существует большое количество разнообразных интерпретация понятия «веб-сайт», но все они сводятся к одному основному смыслу. На мой взгляд, наиболее полным можно считать определение, представленное на сайте Википедия, согласно которому сайт, или веб-сайт (от англ. website: web — «паутина, сеть» и site — букв. «место, сегмент, часть в сети»), также веб-узел, — одна или несколько логически связанных между собой веб-страниц; также место расположения контента сервера. Основной фокус данного определения сделан на связанных некоторой логикой между собой веб-страниц, содержащих определенный контент. И это действительно так, ведь для того, чтобы сайт был полезным его пользователям, он должен восприниматься ими как единое целое, что совершенно невозможно без общей смысловой концепции.

При этом данная логика имеет большое значение. Именно она определяет весь вектор разработки сайта, а также то, какой контент и в каком формате будет представлен на этом сайте. В зависимости от типа контента можно выделить следующие виды веб-сайтов:

1. Корпоративные сайты. Предназначены для представления компании, ее услуг и продукции. Такие сайты обычно включают разделы "О компании", "Услуги", "Новости", "Контакты" и другие. Они могут содержать элементы интерактивности, такие как формы обратной связи, чат-боты и динамические элементы.
2. Интернет-магазины: Специализированные платформы для онлайн-продаж товаров и услуг. Включают каталог товаров, систему управления заказами, корзину покупок, интеграцию с платежными системами и другие функции. Интернет-магазины требуют высокой степени безопасности для защиты данных пользователей и обеспечения конфиденциальности транзакций.
3. Блоги и новостные порталы: Платформы для публикации статей, новостей и информации. Блоги позволяют авторам делиться своими мыслями, а новостные порталы предоставляют актуальные новости и аналитику. Такие сайты часто включают системы управления контентом (CMS) для удобства публикации и администрирования материалов.
4. Лендинги: Одностраничные сайты, созданные для маркетинговых кампаний и привлечения внимания к конкретному продукту или услуге. Лендинги обычно включают яркий дизайн, призывы к действию (CTA) и формы для сбора контактных данных посетителей. Они оптимизированы для высокой конверсии и используются в рекламных кампаниях.
5. Одностраничные сайты (Single Page Applications, SPA): Сайты, вся информация на которых доступна на одной странице. SPA обеспечивает высокую скорость загрузки и удобство использования, загружая данные динамически по мере необходимости без полной перезагрузки страницы. Это особенно важно для улучшения пользовательского опыта и повышения производительности сайта.

Помимо ожиданий относительно вида контента современные пользователи предъявляют высокий ряд требований к функционированию веб-сайтов. В связи с этим можно выделить следующие особенности современных веб-сайтов:

1. Адаптивный дизайн: Современные веб-сайты должны корректно отображаться на различных устройствах, включая компьютеры, планшеты и смартфоны. Адаптивный дизайн (responsive design) позволяет сайту автоматически подстраиваться под размер экрана и разрешение устройства, обеспечивая удобство использования на любом устройстве.
2. Интерактивность: Высокая степень взаимодействия с пользователем через формы, анимации, всплывающие окна и другие элементы. Интерактивные элементы улучшают пользовательский опыт, делая сайт более привлекательным и удобным для посетителей.
3. Быстрая загрузка: Оптимизация скорости загрузки страниц для улучшения пользовательского опыта и повышения рейтинга сайта в поисковых системах. Быстрая загрузка страниц важна для удержания посетителей и предотвращения их ухода на сайты конкурентов.
4. SEO-оптимизация: Подстройка контента и структуры сайта для улучшения видимости в поисковых системах. SEO-оптимизация включает использование ключевых слов, мета-тегов, создание качественного контента и улучшение внутренней структуры сайта.
5. Безопасность: Защита данных пользователей и самого сайта от различных угроз, таких как хакерские атаки, вредоносное ПО и утечки данных. Современные веб-сайты должны использовать SSL-сертификаты, системы аутентификации и другие меры безопасности для защиты информации.

Исходя из вышесказанного, можно сделать вывод о том, что для небольших компаний веб-сайт должен быть особенно тщательно продуманным, чтобы эффективно представлять бизнес в онлайн-среде и привлекать целевую аудиторию. Пользовательский интерфейс должен быть простым и легко воспринимаемым, чтобы посетители могли быстро найти нужную информацию. Адаптивный дизайн должен быть минималистичным, но привлекательным, отражая бренд и специфику бизнеса. Контент должен быть актуальным, лаконичным и быстро загружаемым. Разработка сайта не должна занимать слишком много времени и денежных средств. Именно такой сайт может стать не просто онлайн-визиткой, а мощным средством для привлечения новых клиентов и удержания существующих.

1.3. Основные элементы современного веб-сайта.

Современные веб-сайты состоят из множества элементов, каждый из которых играет важную роль в обеспечении функциональности, удобства использования и визуальной привлекательности. Основные элементы современного веб-сайта включают в себя структуру, интерактивные компоненты и визуальные элементы.

Структура веб-сайта (или же его устройство) определяет, как контент организован и представлен пользователю. Она включает в себя следующие ключевые компоненты:

1. Шапка сайта (Header): Включает логотип, навигационное меню, контактную информацию и другие важные элементы. Шапка часто остается фиксированной на странице, обеспечивая легкий доступ к навигации и важной информации.
2. Главная часть (Main Content): Основной контент страницы, включающий текст, изображения, видео и другие медиафайлы. Главная часть может быть разделена на несколько секций, каждая из которых отвечает за определенную информацию или функциональность.
3. Боковые панели (Sidebars): Дополнительные необязательные области, которые могут содержать навигационные ссылки, рекламу, виджеты и другие элементы. Боковые панели часто используются для размещения второстепенной информации, которая дополняет основной контент.
4. Меню: Может быть как горизонтальным, так и вертикальным. Обычно содержит в себе группу навигационных ссылок на другие важные страницы для удобства поиска по сайту пользователей.
5. Подвал (Footer): Содержит дополнительную информацию, такую как контакты, ссылки на социальные сети, политика конфиденциальности и другие. Подвал обычно находится в нижней части страницы и используется для размещения общедоступной информации.

Базовая структура сайта представлена на Иллюстрации 2:

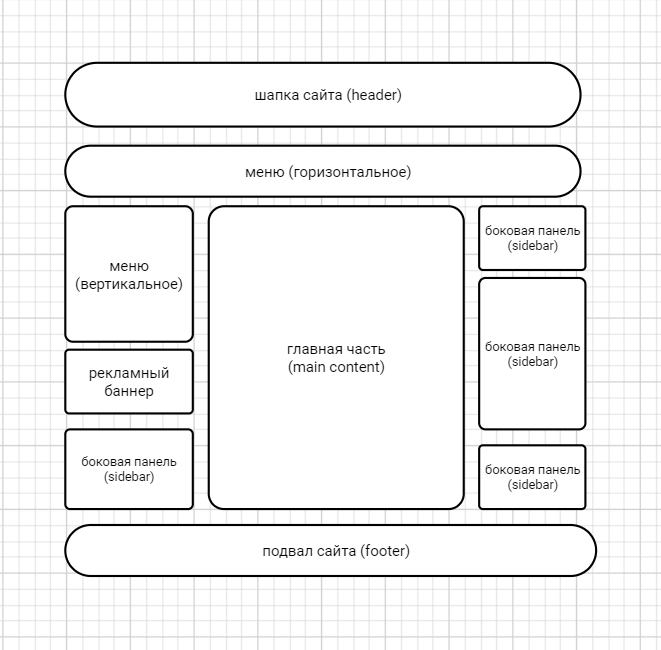


Иллюстрация 2. Базовая структура сайта.

Итак, Иллюстрация 2 наглядно демонстрирует базовое устройство веб-страницы, которое встречается на большинстве современных сайтов.

Еще одной важнейшей составляющей для сайта является его интерактивность. Интерактивные элементы улучшают пользовательский опыт и делают веб-сайты более привлекательными и удобными для посетителей. Современные веб-сайты используют множество интерактивных элементов, чтобы обеспечить высокую степень взаимодействия с пользователями.

Основными интерактивными элементами, которые чаще всего встречаются на современных сайтах, являются следующие:

1. Формы для сбора данных пользователей, такие как контактные формы, формы регистрации и подписки. Они позволяют пользователям взаимодействовать с сайтом, отправлять запросы, оставлять комментарии и подписываться на рассылки. Хорошо спроектированные формы включают в себя поля с валидацией данных, чтобы обеспечить корректность вводимой информации.
2. Использование CSS и JavaScript для создания плавных анимаций и переходов между элементами. Анимации могут привлекать внимание пользователей, делая сайт более динамичным и живым. Переходы между страницами и элементами помогают пользователям лучше ориентироваться на сайте и улучшать их восприятие информации.
3. Модальные окна и всплывающие сообщения, которые также используются для привлечения внимания к важной информации или для взаимодействия с пользователем. Модальные окна могут содержать уведомления, формы, инструкции и другую важную информацию, которая требует немедленного внимания пользователя.
4. Кнопки, вкладки, слайдеры и другие элементы, улучшающие пользовательский опыт. К примеру, слайдеры позволяют пользователям просматривать изображения или контент в ограниченном пространстве, а вкладки помогают организовать информацию и сделать её более доступной.
5. Чат-боты и онлайн-консультанты для общения с пользователями в реальном времени. Они могут отвечать на вопросы, помогать с навигацией по сайту и предлагать поддержку, что особенно важно для малого бизнеса, стремящегося улучшить обслуживание клиентов.

Таким образом, интерактивные элементы стали привычной для большинства пользователей частью веб-сайта благодаря удобству использования, привлечению внимания к важной информации и возможностью взаимодействовать с компанией в режиме реального времени.

Подводя итоги, можно сделать вывод о том, что современный сайт может включать в себя такие элементы как шапка, основной контент, дополнительные горизонтальные и вертикальные меню, боковые панели, рекламные баннеры и подвал. Актуальный внешний вид веб-странице придают такие интерактивные элементы как формы, анимации, модальные и всплывающие окна, онлайн-консультанты и другие.

1.4. Основные сетевые компоненты и протоколы для функционирования веб-сайтов.

Для более глубокого понимания устройства веб-сайтов следует изучить основы функционирования современных сайтов и дополнительно разобрать значение таких понятий как домен, IP-адрес, порт и протокол.

Домен или доменное имя представляет собой определенную последовательность, состоящую из букв латинского алфавита и/или цифр, которая обозначает имя или название сайта. Доменные имена давно вошли в обиход, стали привычными для обычных пользователей в силу удобства для понимания и запоминания и используются как имя сайта для быстрого поиска.

К примеру, доменное имя example.ru имеет две составные части:

1. ru – это доменное имя первого уровня,

2. example – это доменное имя второго уровня.

Следует отметить то, что домен может содержать более двух уровней и их количество не ограничено, например, proglive.tiu.ru. Однако обычно большое количество уровней не используется, опять-таки по причине усложнения запоминания для пользователей и росте вероятности допустить ошибку в длинном имени.

Как уже было сказано ранее, использование доменных имен обусловлено удобством для пользователей, при этом обмен данными между пользователем и сайтом происходит посредством уникального IP-адреса.

Сетевой или IP-адрес сайта – это уникальный сетевой идентификатор компьютера в виде маршрутизируемого протокола сетевого уровня, который выстраивается согласно сетевой модели передачи данных TCP/IP («от источника информации к получателю»). Именно благодаря IP-адресу существует возможность обращаться к сайту.

На текущий момент существует 2 версии IP-адреса:

1. IPv4 – к примеру, 23.255.30.11. Каждое из чисел в адресе — это восьмизначное двоичное число, или октет. Оно может принимать значения от 0000 0000 до 1111 1111 или же от 0 до 255 в десятичной системе счисления — то есть 256 разных значений. Поэтому диапазон IP-адресов стартует с 0.0.0.0 и заканчивается 255.255.255.255, а значит, если посчитать количество всех возможных адресов в этом диапазоне, то получится 4 294 967 296. По этой причине появилась следующая версия IP-адреса IPv6.

2. IPv6 – к примеру, 2001:0db8:11a4:09d7:0000: 0000: 7a1e: 1460: 5334. Адреса этой версии состоят уже из 128 битов (в то время как в IPv4 — 32 бита). Таким образом, IPv6 позволяет пронумеровать 2128 устройств (по 300 миллионов на каждого жителя Земли) и проблема заканчивающихся IP-адресов на ближайшее время решена.

То есть каждый сайт в сети Интернет имеет уникальный IP-адрес и доменное имя, а для того, чтобы пользователь мог открыть некоторый веб-сайт, ему нужно с помощью браузера обратиться к сайту по IP-адресу. Подобные запросы обрабатывает сервер — специальная программа, которая запущена в целях обработки запросов, выполнения некоторой логики и возвращения искомой пользователем страницы. Именно сейчас следует ввести следующее важное понятие «порт».

Порт - это целое неотрицательное число, представляющее собой идентификатор программы или процесса, который обслуживает сетевые соединения на заданном IP-адресе. Номера портов могут быть от 0 до 65 535. Порт существует для разделения программ. Именно с помощью порта любая программа (и браузер, и сервер) понимает, нужно ли ему обрабатывать присланные данные или нет, и обрабатывает только то, что нужно. Так происходит разделение запросов.

Существуют зарезервированные порты для сайтов:

1. порт 80 — для веб-сервера, который работает по HTTP-протоколу передачи данных,

2. порт 443 - для веб-сервера, который работает по HTTPS-протоколу передачи данных.

Кроме того, для веб-сайтов допустимо использование нестандартных портов (то есть тех, которые не зарезервированы под иные нужны), однако в этом случае их необходимо явно прописывать, что обычно не делают для удобства пользователей.

И наконец, протокол передачи данных — это набор правил и соглашений, которые определяют способ обмена данными между устройствами и программами в сети. Это правила «общения» между программами, которые определяют, как данные упаковываются, отправляются и как устройства должны взаимодействовать друг с другом.

Основные протоколы обмена данными на текущий момент:

1. HTTP - Hypertext Transfer Protocol - это протокол передачи гипертекста, который используется при пересылке веб-страниц. К примеру, http://example.ru.

2.HTTPS - Hypertext Transfer Protocol Secure – это расширение протокола HTTP для поддержки шифрования в целях повышения безопасности. Данные в протоколе HTTPS передаются в зашифрованном виде. К примеру, https://example.ru.

Таким образом, все необходимые для понимания процесса функционирования веб-сайта понятия разобраны, а значит, можно перейти к разбору данного процесса.

Обмен данными между пользователем, который с помощью браузера запрашивает некоторую HTML-страницу, и сервером, который может ее отдать, представлено на Иллюстрации 3:

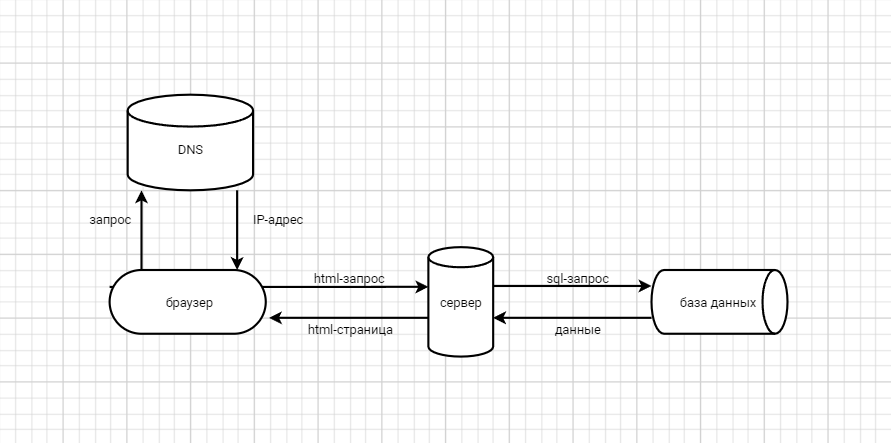


Иллюстрация 3. Схема http-запроса страницы.

Иллюстрация 3 наглядно изображает упрощенную схему http-запроса страницы. Когда пользователь вводит адрес сайта, его браузер в своей памяти проверяет, открывал ли он этот сайт ранее. Если нет, тогда он не знает, по какому IP-адресу необходимо обратиться. В этом случае браузер обращается к одной из DNS-служб (сервера, созданные по всему миру) своего провайдера. Провайдеры создают подобные службы для того, чтобы не перегружать свои сети. Однако может произойти так, что DNS-служба провайдера пользователя также не будет знать IP-адрес искомого сайта. В этом случае браузер пойдет дальше и рано или поздно моет дойти до одного из корневых DNS, в котором хранятся все существующие IP-адреса. Однажды получив IP-адрес, браузер хранит их для последующих обращений.

После выяснения IP-адреса, браузер обращается на 443 порт, сервер выдает сертификат, в котором указано его доменное имя. По сути, этот сертификат является цифровой подписью от удостоверяющего центра, выдавшего сертификат, подтверждая подлинность сайта и устанавливая зашифрованное соединение с браузером. После проверки сертификата и установки соединения, браузер отправляет HTTP(S)-запрос на сервер по соответствующему IP-адресу и порту.

Процесс передачи данных между пользователем и сервером включает следующие этапы:

1. Введение адреса сайта: Пользователь вводит URL в адресной строке браузера (например, https://example.ru).
2. DNS-запрос: Браузер проверяет свой кэш на наличие IP-адреса для данного домена. Если в кэше нет нужной записи, браузер отправляет запрос к DNS-серверу провайдера. Если провайдер не знает IP-адрес, запрос передается дальше, пока не будет найден соответствующий DNS-запись (включая запросы к корневым DNS-серверам).
3. Получение IP-адреса: DNS-сервер возвращает IP-адрес, соответствующий введенному домену. Браузер сохраняет этот адрес в своем кэше для последующих обращений.
4. Установка соединения: Браузер устанавливает соединение с сервером по найденному IP-адресу, используя порт 443 для HTTPS (или 80 для HTTP). В случае HTTPS-соединения сервер отправляет свой SSL-сертификат браузеру.
5. Проверка сертификата: Браузер проверяет подлинность сертификата, убеждаясь, что он выдан доверенным удостоверяющим центром и что доменное имя сертификата соответствует адресу сайта.
6. Шифрование: Если сертификат подтвержден, устанавливается зашифрованное соединение с использованием протокола TLS (Transport Layer Security). Это гарантирует, что данные передаются в зашифрованном виде и защищены от перехвата.
7. Отправка запроса: Браузер отправляет HTTP(S)-запрос на сервер. Запрос может быть GET (запрос данных) или POST (отправка данных).
8. Обработка запроса: Веб-сервер принимает запрос, обрабатывает его (может включать запрос к базе данных или выполнение бизнес-логики), и формирует ответ, который затем отправляется обратно клиенту.
9. Получение и отображение ответа: Браузер получает ответ от сервера, который обычно содержит HTML-документ, стили CSS, JavaScript-код и другие ресурсы. Браузер интерпретирует эти данные и отображает их пользователю в виде веб-страницы.
10. Кэширование данных: Браузер может кэшировать полученные данные, чтобы ускорить доступ при следующем посещении того же сайта.

Таким образом, процесс работы веб-сайта включает в себя множество этапов взаимодействия между пользователем, браузером, DNS-серверами и веб-сервером, обеспечивая быструю и безопасную передачу данных.

Подводя общие итоги к Главе 1 данной дипломной работы, можно выделить следующие основные тезисы.

Вебсайт – это связанные некоторой логикой между собой одна или несколько веб-страниц, содержащих определенный контент. Данный контент разбивается на такие основные элементы сайта как шапка, основной контент, меню, дополнительные модули и подвал. Прошли те дни, когда пользователям было достаточно статичных данных, сейчас для сайтов очень важен адаптивный и интерактивный дизайн хорошо продуманной структуры.

Браузер делает http-запрос на сервер. Он, в свою очередь, этот запрос принимает, выполняет определенную работу (sql-запрос) при необходимости и возвращает ему html-страницу. Затем браузер эту страницу отрисовывает для пользователя.

Глава 2. Современные инструменты для разработки веб-сайтов.

2.1. Обзор актуальных инструментов разработки.

В современном мире веб-разработка стремительно развивается, предоставляя разработчикам широкий спектр инструментов и технологий для создания эффективных, интерактивных и эстетически привлекательных веб-сайтов.

Основные технологии, такие как HTML, CSS и JavaScript, являются фундаментом для построения веб-страниц, обеспечивая структуру, стильный внешний вид и функциональность.

Кроме этих базовых технологий, разработчики активно используют различные фреймворки и библиотеки, которые значительно ускоряют процесс разработки и обеспечивают более высокое качество кода. Среди них выделяются такие популярные фреймворки, как React, Angular и Vue.js. Эти инструменты предоставляют мощные возможности для создания сложных и масштабируемых веб-приложений.

Помимо фреймворков, современная веб-разработка включает в себя использование вспомогательных инструментов, которые облегчают и ускоряют работу программистов. К таким инструментам относятся редакторы кода, системы контроля версий, сборщики, средства автоматизации и отладки, а также платформы для развертывания.

Редакторы кода являются основным инструментом любого разработчика при создании кода. Они обеспечивают удобную среду для написания, редактирования и отладки кода. На сегодняшний день наиболее популярными редакторами кода являются следующие:

1. Visual Studio Code (VS Code) - этот редактор от Microsoft быстро завоевал популярность благодаря своей легкости, расширяемости и поддержке множества языков программирования. VS Code обладает встроенными средствами для отладки, управления версиями и интеграцией с Git. Именно этот редактор кода будет использован при создании данной дипломной работы.
2. Sublime Text - известен своей скоростью и минималистичным интерфейсом. Sublime Text поддерживает множество плагинов, которые добавляют функциональность для различных языков и фреймворков.
3. Atom - разработанный GitHub, этот редактор также ориентирован на расширяемость и поддерживает множество пакетов, которые можно установить для улучшения функциональности.

Системы контроля версий являются неотъемлемой частью современного процесса разработки. Они позволяют отслеживать изменения в коде, работать в команде и управлять различными версиями проекта. Основными системами контроля версий являются следующие:

1. Git - самая популярная система контроля версий. Git позволяет разработчикам работать над проектами параллельно, создавая и сливая ветки. GitHub, GitLab и Bitbucket являются популярными платформами для хостинга Git-репозиториев. При создании данной дипломной работы будет использована платформа GitHub.
2. Subversion (SVN) - несмотря на то, что SVN значительно уступает по популярности Git, он все еще используется в ряде крупных проектов. SVN обеспечивает централизованное управление версиями и хорошо подходит для проектов, где необходим строгий контроль доступа.

Помимо редакторов кода и системы контроля версий для упрощения процесса сборки и развертывания проектов используются различные сборщики и средства автоматизации, такие как Webpack, Gulp и Parcel:

1. Webpack - мощный инструмент для сборки JavaScript-приложений. Webpack позволяет обрабатывать файлы различных типов (JavaScript, CSS, изображения) и создает оптимизированные сборки для развертывания.
2. Gulp - средство автоматизации повторяющихся задач, к примеру, минификация файлов, компиляция SCSS в CSS и перезагрузка браузера.
3. Parcel - сборщик, который отличается простотой настройки и высокой скоростью сборки. Parcel автоматически настраивает зависимости и обеспечивает оптимизацию выходных файлов.

Для обеспечения качества и стабильности кода используются различные средства отладки и тестирования, такие как:

1. Chrome DevTools - встроенные в браузер Google Chrome инструменты для отладки веб-приложений. DevTools позволяют отслеживать выполнение кода, анализировать производительность и просматривать сетевые запросы. Будет использован при работе над данной дипломной работой.
2. Jest - фреймворк для тестирования JavaScript-кода, разработанный Facebook. Jest поддерживает тестирование компонентов React и позволяет писать модульные и интеграционные тесты.
3. Cypress - инструмент для тестирования веб-приложений. Cypress позволяет писать тесты, которые проверяют поведение приложения с точки зрения пользователя.

Для развертывания веб-приложений используются различные облачные платформы, к примеру:

1. Netlify - платформа для развертывания статических сайтов и серверных функций. Netlify обеспечивает простую настройку CI/CD и интеграцию с Git.
2. Vercel - платформа, оптимизированная для развертывания Next.js приложений, но поддерживающая также другие фреймворки. Vercel предоставляет масштабируемую инфраструктуру и простые инструменты для развертывания.
3. Heroku - платформа как услуга (PaaS), которая позволяет развертывать и управлять веб-приложениями на различных языках программирования. Heroku обеспечивает автоматическое масштабирование и поддержку множества аддонов для расширения функциональности.

Таким образом, современные вспомогательные инструменты разработки веб-сайтов значительно упрощают процесс создания, тестирования и развертывания приложений. Они позволяют разработчикам работать более эффективно, обеспечивая высокое качество и производительность конечных продуктов. При работе над практической частью данной дипломной работы будут использованы редактор кода Visual Studio Code и система контроля версий Git на платформе GitHub. В качестве средства отладки и тестирования будет использован Chrome DevTools, встроенный в браузер Google Chrome.

2.2. HTML и его место в разработке веб-сайтов.

Как уже было проанализировано в Главе 1 данной дипломной работы, HTML является основой для создания веб-страниц. Именно HTML используется для структурирования содержимого страницы, тогда как остальные инструменты, такие как CSS, JavaScript и фреймворки, только улучшают его, делая визуально привлекательнее и интерактивнее.

HTML (HyperText Markup Language) – это стандартный язык разметки документов в Интернете. Данный язык позволяет добавлять такие элементы на веб-страницу, как текст, изображения, ссылки, кнопки и так далее.

Базовая структура HTML-документа представлена на Иллюстрации 4:



Иллюстрация 4. Базовая структура HTML-документа.

Согласно Иллюстрации 4, представленной выше, HTML-документ начинается с объявления элемента DOCTYPE (строка 1), который представляет из себя тип текущего документа — DTD (document type definition, описание типа документа). DTD обозначает, какой именно диалект, основанный на HTML, используется. В современной версии достаточно указать <!DOCTYPE html>, в то время как ранее были несколько других вариантов, в том числе специальная версия — XHTML. Все они обозначались своими DOCTYPE и их вариациями, которые можно встретить исключительно в старом коде. В любом случае, DTD – это информация, которая нужна для того, чтобы браузер максимально правильно отрисовать разметку.

Следом на строке 3 идет тег <html>. Вся информация, которая содержится в html-документе, находится внутри этого тега. Тег <html> состоит из двух тегов: <head> и <body>.

Тег <head> представляет из себя шапку документа. Он содержит служебную информацию, которая не отображается визуально на странице, но тем не менее содержит важнейшие настройки для того, чтобы страница отображалась корректно. Здесь обычно подключаются различные файлы, библиотеки, а также указывается заголовок вкладки:

1. <meta charset='UTF-8'> - указание кодировки документа. Без этого можно увидеть символы вместо предполагаемого текста при открытии страницы в браузере.
2. <meta name='viewport' …> - важный мета-тег для указания браузеру информации, что сайт будет адаптивным и сможет поддерживать медиа-запросы, а также будет правильно масштабироваться на маленьких экранах.
3. title — заголовок вкладки.

И, наконец, тег <body> необходим для размещения всего контента, который должен быть отображен на странице.

При создании веб-страниц используется множество HTML-тегов. Все они служат для определения структуры и содержимого документа. Основными тегами, используемыми для оформления контента, являются <div>, <span>, <p>, <a>, <img>, <form>, <input>, <label>, и <button>.

Тег <div> используется для создания контейнеров или блоков. Это универсальный контейнер, который можно использовать для группировки других элементов. Данный тег имеет следующий синтаксис:

* <div> Контент внутри </div>.

Особенности поведения тега <div>, а также всех прочих блочных элементов таковы:

1. Содержимое тега занимает всю возможную ширину.
2. Его высота определяется наполнением.
3. Контент всегда начинается с новой строки и после себя добавляет перенос строки.

К прочим основным блочным элементам, имеющим аналогичное поведение, относятся <p>, <h1> - <h6>, <ul>, <ol>, <li>, <table>, <tr>, <th>, <td>, <form>, и так далее.

Тег <span> в отличие от <div>, <span> является строчным элементом. Он используется для выделения части текста или других элементов внутри строки. Данный тег имеет следующий синтаксис:

* <span> Текст внутри </span>.

К особенностям тега <span> и прочих строчных элементов относят следующие:

1. Являются частью строки. По сути, все строчные элементы являются частью окружающего текста.
2. Ширина определяется содержимым и отступами слева и справа. Поэтому свойства, связанные с размерами, не применяются, даже если их прописать в CSS.
3. Строчные элементы, идущие подряд, не переносятся на новую строку и располагаются друг за другом как обычный текст.

К прочим основным строчным элементам, имеющим аналогичное поведение, относятся <a>, <img>, <input>, <label>.

Тег <p> предназначен для создания абзацев текста. Как уже было сказано ранее, является блочным элементом. Синтаксис тега имеет следующий вид:

* <p>Текст</p>

Тег <a> используется для создания гиперссылок на другие страницы или ресурсы. В зависимости от выбранного атрибута name или href тег устанавливает ссылку или якорь. Якорь представляет собой ссылку на часть той же страницы, на которой находится тег <a>. Возможный синтаксис представлен ниже:

* <a href="URL">...</a>
* <a name="идентификатор">...</a>

Следующий тег <img> служит для отображения на сайте изображения в формате gif, jpeg или png. Адрес файла с картинкой задается через обязательный атрибут src, при этом путь может быть как относительным, так и абсолютным.

Синтаксис тега таков:

* <img src="URL" alt="альтернативный текст">

Тег <form> используется для создания формы ввода данных для пользователя. Область применения форм не ограничена отправкой данных на сервер, с помощью клиентских скриптов можно получить доступ к любому элементу формы, изменять его и применять по своему усмотрению. Тег имеет следующий синтаксис:

* <form action="/submit" method="post">

<label for="name">Имя:</label>

<input type="text" id="name" name="name">

<input type="submit"> // или <button>Отправить</button>

</form>

Неотделимыми от формы являются теги <label> и <input>. Если <label> используется для связывания метки с элементом формы, то <input> используется для создания интерактивных элементов управления в формах, таких как текстовые поля, кнопки, переключатели и флажки.

На текущий момент отправка формы возможна двумя способами:

1. С помощью элемента <input type="submit">, который указан перед закрывающим тегом </form>.
2. С помощью следующего элемента <button>, который используется намного чаще:

* <button>Отправить</button>

Кроме того, HTML5, последняя версия HTML, которая была выпущена в 2014 году, принесла с собой множество новых возможностей и улучшений, которые значительно упростили разработку веб-страниц. Некоторые из ключевых нововведений HTML5 таковы:

1. Семантические элементы: HTML5 ввел новые теги, такие как <header>, <footer>, <article>, <section>, <nav>, <main> и так далее, которые улучшают семантическое значение структуры страницы. Это не только облегчает понимание кода, но и улучшает SEO (поисковую оптимизацию) и доступность для людей с ограниченными возможностями. При использовании семантических тегов облегчается понимание целей данного контента как для самих программистов, так и для поисковых роботов.
2. Мультимедиа: HTML5 добавил встроенную поддержку для видео и аудио через теги <video> и <audio>, что позволяет воспроизводить мультимедийный контент без использования сторонних плагинов.
3. Формы: Новые атрибуты и элементы для форм, такие как <input type="email"> и <input type="date">, упрощают валидацию и ввод данных, улучшая пользовательский опыт.

Таким образом, HTML является языком разметки документа, предоставляя большое количество различных тегов для отображения необходимого контента на странице. Все теги можно разделить на блочные и строчные. Их основное отличие заключается в различном поведении и в занимаемом пространстве. Благодаря этому рассмотренные теги позволяют размещать на веб-сайтах разнообразный контент и даже разделять его по смыслу с помощью семантических тегов.

2.3. CSS как основа оформления в современных проектах

Как уже было отмечено ранее, пользователи начали предъявлять требования к оформлению сайтов, созданных с помощью HTML-разметки. В ответ на эти растущие ожидания появились каскадные таблицы стилей. Если HTML используется для структурирования контента на веб-странице, то CSS (Cascading Style Sheets) отвечает за его визуальное оформление.

CSS - Cascading Style Sheets - каскадные таблицы стилей - это формальный язык, который используют, чтобы задать визуальное отображение веб-страницам. Он является дополнением к HTML, так как оформляет HTML-разметку в более привлекательный для пользователя вид.

Синтаксис CSS выглядит следующим образом:

* селектор {

cвойство1: значение 1;

cвойство2: значение 2;

}

Селектор в CSS – это особые правила, которые определяют, к каким элементам необходимо применить стили. Существует 5 видов следующих селекторов:

1. Селектор тегов – применение ко всем указанным тегам в документе:

* div {background: green; } – применить зеленый цвет фона ко всем элементам <div>

1. Селектор идентификатора (id) – применение к уникальному элементу с id, который может быть только один на странице:

* #name { background: red; } - применить красный цвет фона к единственному элементу с id “name”

1. Селектор класса – применение, когда нужно использовать данные стили чаще чем один раз, а именно ко всем элементам с данным классом:

* .sendButton {background: orange; } - применить оранжевый цвет фона ко всем элементам с классом “ sendButton”

1. Селектор атрибутов – связаны с атрибутами и их значениями:

* img[alt=”new”] { background: yellow; } - применить желтый цвет фона ко всем изображениям, атрибуты которых равны значению “ dress ”

1. Селектор регулярных выражений – селекторы, которые содержат специальные символы, к примеру, ^, $, \*.

* [attr^=" item"] { background: green; }- применить зеленый цвет фона ко всем элементам, атрибут которых начинается сo значения “item”
* [attr$=" item"] { background: green; } - применить зеленый цвет фона ко всем элементам, атрибут которых заканчивается на значение “item”
* [attr=" item"]\* { background: green; } - применить зеленый цвет фона ко всем элементам, атрибут которых содержит значение “ item”

Чаще всего в современных компаниях используются селекторы классов. Селекторы идентификатора ограничены в использовании из-за ограничений в переиспользовании, а использование селекторов тегов иногда считается дурным тоном.

Основными свойствами стилей, без которых не обходится ни одна веб-страница, являются шрифт и единицы измерения, высота и ширина, цвета, а также фон.

Для обозначения размеров, расстояний, шрифтов и других параметров в CSS существует большое количество различных единиц измерения. Основными из них являются следующие:

1. px – пиксели – используются для точного определения размера относительно экрана. Один пиксель равен одному пикселю на экране пользователя.
2. % - проценты - обычно используются для задания ширины и высоты относительно родительского элемента
3. em – относительная единица, которая основывается на размере шрифта родительского элемента. Если размер шрифта родительского элемента 16px, то 1em будет равен 16px.
4. rem - похож на em, но относится к размеру шрифта корневого элемента (обычно <html>). Если размер шрифта корневого элемента 16px, то 1rem будет равен 16px.
5. vw - viewport width - 1vw равен 1% от ширины окна браузера.
6. vh - viewport height - 1vh равен 1% от высоты окна браузера.

За отображение шрифта текста в CSS отвечает свойство font (с английского шрифт). Данный селектор отвечает за группу следующих свойств:

1. font-family - семейство шрифтов, которым будет отображаться текст. К примеру, div { font-family: ‘Verdana’, serif; }. Данное свойство работает таким образом: когда пользователь открывает страницу, браузер пытается найти шрифт ‘Verdana’ в операционной системе компьютера пользователя, и, если он его найдет, то применит, а если нет, то начнет работать так называемое специальное обобщающее для шрифтов с засечками значение serif (также существуют sans-serif для обобщения рубленного шрифта, cursive – для курсивных шрифтов, fantasy – для декоративных шрифтов и monospace – для моноширинных шрифтов), а значит, браузер подставит любой имеющийся в его операционной системе шрифт с засечками. При этом важно знать, что современные возможности предлагают широкий спектр простых инструментов, позволяющих не надеяться на то, что необходимый шрифт найдется у пользователя на компьютере, и подключать их к веб-сайту.
2. font-size - размер шрифта в выбранных единицах измерения, к примеру,

h1 { font-size: 12px; }.

1. font-weight - используется для вывода жирного шрифта. Значения свойства font-weight: normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900. К примеру, p { font-weight: lighter; }
2. font – объединенное свойство, позволяющее записывать несколько свойств шрифта в одну строку, к примеру, span { font: bold 24px ‘Verdana’; }.

За обозначение ширины элемента, обычно блочного, так как на строчные элементы данное свойство не распространяется, отвечает свойство width, к примеру, button {width: 150px; }.

Пример свойства height, отвечающее за высоту элемента: div { height: 5%; }.

Цвета в CSS представлены следующим образом:

1. Именованные цвета – к примеру, orange, green и так далее. При этом важно отметить, что названия существуют далеко не для всех цветов, и поэтому использование этой группы весьма ограничено. Однако их использование обуславливается удобством, а потом широко распространено на многих сайтах, в особенности для таких базовых цветов как черный (black) и белый (white).
2. Функциональные цвета – к примеру, RGB(255, 130, 0).
3. Шестнадцатиричные цвета – к примеру, #FA96CF.

И, наконец, фон, последний из основных разбираемых свойств, в CSS представлен свойством background. Аналогично свойству font, данное свойство background является объединяющим для следующих свойств фона в CSS:

1. background-color - цвет фона, к примеру, div { background-color: red;}.
2. background-image - одно или несколько фоновых изображений для элемента, к примеру, button { background-image: url(img/foto.jpg);}.
3. background-position – расположение фоновой картинки относительно экрана, к примеру, div { background- position: center;}.
4. background-repeat – свойство, регулирующее, как будет повторяться фоновое изображение, к примеру,

button { background-repeat: no-repeat;}.

1. background – как уже было сказано ранее, объединяющее свойство, к примеру, div { background: #ff0 url(img/foto.jpg) top repeat-x;}.

CSS3, текущая версия CSS, также принесла множество новых возможностей, которые значительно расширили возможности стилизации веб-страниц, а именно Flexbox, Grid, анимации и переходы, препроцессоры CSS и адаптивный дизайн.

Флексбокс (Flexbox) представляет собой модель компоновки, которая упрощает создание гибких и адаптивных макетов. Данный инструмент позволяет легко выравнивать и распределять пространство между элементами контейнера. К примеру, div {display: flex; justify-content: center; align-items: center;}. Данная запись позволяет распределить контент, содержащийся внутри элемента div четко в центре вне зависимости от размеров самого div, его содержимого и окружающих элементов.

Грид (Grid) – это одна мощная модель компоновки, которая позволяет создавать сложные макеты с использованием сетки. CSS Grid предоставляет гибкость в размещении элементов в двух измерениях, что особенно полезно для построения адаптивных дизайнов.

Структура грида напоминает обычный Excel-файл: есть горизонтальные и вертикальные линии, которые вместе образуют много разных ячеек. В отличие от файла в Excel, внутри ячеек находятся элементы сайта: текст, картинки, кнопки, блоки HTML-кода и так далее. Формально все эти части грид-разметки называются так:

1. grid-контейнер — самый главный элемент во всей разметке, в нём хранится всё содержимое сетки;
2. grid-ячейка — единица грид-сетки, сюда можно положить один или несколько блоков кода;
3. grid-линия — горизонтальная или вертикальная линия, разделяющая столбцы и колонки;
4. grid-строка (row) — ряд ячеек;
5. grid-столбец (column) — колонка ячеек;
6. grid-элемент — какой-либо элемент сайта;
7. grid-область (area) — пространство из ячеек, в CSS можно объединить несколько ячеек в одну и работать с ними как с единым целым.

Пример ниже позволяет задать размеры и количество всех колонок и рядов:

div {

display: grid;

grid-template-columns: 15px auto 20%;

grid-template-rows: 200px 10vw 10rem;

}

Если говорить об анимациях и переходах, то CSS3 ввел сильнейшую поддержку, что позволяет создавать плавные визуальные эффекты без использования JavaScript. Это улучшает пользовательский опыт и делает интерфейсы более интерактивными.

Анимация представляет собой несколько изображений, показанных последовательно, что создает иллюзию движения. Анимации привлекают внимание пользователей и помогают сделать интерфейсы понятными. Есть два способа анимировать элементы в CSS: свойства animation и transition.

Свойство animation позволяет изменять свойства элемента в течение определённого периода, а transition определяет, как элемент меняется за определённый период. Для animation нужны @keyframes, то есть требуется определить точки начала и конца изменений. Ключевые кадры используют для пошаговых анимаций. Более простые анимации создают при помощи transition. В этом случае движение запускается по определённому сигналу, например, по клику или наведению курсора. Свойства animation и transition управляют продолжительностью, задержкой, итерациями движения.

Синтаксис для стилизации переходов в CSS выглядит следующим образом: a { transition: property duration timing-function delay;}, где property - свойство, которое будет изменяться, duration - продолжительность перехода (например, 2s для двух секунд), timing-function - функция временной зависимости (например, ease, linear, ease-in, ease-out), delay - задержка перед началом перехода (например, 1s для одной секунды).

Препроцессоры в CSS стали отличным вспомогательным инструментом, который значительно облегчил работу над дизайном сайтов. По сути, препроцессор является ‘программистским’ подходом к CSS, так как он позволяет при написании стилей использовать свойственные языкам программирования приемы и конструкции, а именно переменные, вложенность, наследуемость, циклы, функции и математические операции. Код, написанный на препроцессоре, не используется сразу, сначала он компилируется в обычный CSS, а затем используется браузерами. На текущий момент существует несколько препроцессоров, наиболее популярным из которых является SASS (также он будет использован в третьей Главе данной дипломной работы для облегчения написания стилей).

Адаптивный дизайн в CSS предполагает использование медиазапросов (media queries) для изменения макета страницы в зависимости от размера экрана. Это обеспечивает оптимальное отображение и удобство использования на различных устройствах. Синтаксис медиазапроса выглядит следующим образом: @media (width: 320px) { селектор {cвойство1: значение1;} }. Наиболее популярным способом использования медиазапроса является изменение размера шрифта в зависимости от размеров экрана, а также изменение представления контента с помощью технологии flexbox или grid, к примеру, изменение количества колонок и так далее.

Таким образом, в CSS современные инструменты и подходы к веб-верстке позволяют создавать сложный, адаптивный и интерактивный дизайн, который обеспечивает высокий уровень пользовательского опыта. Шрифт и единицы измерения, высота и ширина, цвета, а также фон являются основой CSS. CSS3, значительно расширила возможности стилизации сайтов с помощью Flexbox, Grid, анимаций и переходов, препроцессоров и адаптивного дизайна.

2.4. Введение в JavaScript: возможности и роль в веб-разработке.

В условиях современной реальности JavaScript является неотъемлемой и важнейшей частью веб-разработки, предоставляя большие возможности для создания интерактивных и динамических веб-страниц. Изначально JavaScript был создан, чтобы «сделать веб-страницы живыми», то есть только для браузеров. Пользователи уже отвыкли от статических сайтов, написанных исключительно на HTML и CSS. По этой причине актуальный на протяжении многих лет JavaScript эволюционировал и стал мощным и универсальным языком программирования, который используется уже не только на клиентской, но и на серверной стороне. Его использование сегодня распространяется на веб-страницы в браузерах, что является его основным направлением, на серверное программирование с помощью Node.js, на разработку мобильных и настольных приложений, а также на многие другие цели, при которых используется интерпретатор или компилятор JavaScript-кода.

В веб-разработке JavaScript выполняет несколько ключевых функций:

1. Манипуляция DOM: JavaScript позволяет изменять структуру, стили и содержимое HTML-документа в реальном времени. Это позволяет создавать динамические элементы интерфейса, такие как модальные окна, выпадающие меню и слайдеры.
2. Обработка событий: JavaScript может реагировать на различные события, такие как клики мыши, ввод данных или изменение размера окна. Это позволяет создавать интерактивные элементы и улучшать пользовательский опыт.
3. Валидация форм: JavaScript используется для проверки данных, введенных в формы, до их отправки на сервер. Это позволяет предотвратить ошибки и обеспечить корректность введенных данных.
4. AJAX: Асинхронный JavaScript и XML (AJAX) позволяет загружать данные с сервера без перезагрузки страницы. Это улучшает производительность и позволяет создавать интерактивные веб-приложения.

Все это значительно расширяет возможности современных сайтов, улучшает пользовательский опыт и повышает эффективность бизнеса.

Наиболее важными особенностями языка программирования JavaScript можно назвать полную совместимость с HTML и CSS, относительную простоту языка, а также поддержку всеми основными браузерами и работу по умолчанию (а значит, пользователям не приходится дополнительно что-либо настраивать для использования). JavaScript – это единственная браузерная технология, сочетающая в себе все эти три преимущества, а потому достойных аналогов для него не существует и именно поэтому это самый распространённый инструмент для создания интерфейсов в браузере.

Программы на этом языке называются JavaScript. Они могут встраиваться в HTML и выполняться автоматически при загрузке веб-страницы. Скрипты распространяются и выполняются, как простой текст. Им не нужна специальная подготовка или компиляция для запуска.

Самую глубокую, детальную и формализованную информацию о JavaScript содержит Спецификация ECMA-262. Она определяет сам язык и является источник самой достоверной информации о данном языке программирования. Новая версия спецификации появляется каждый год, а до ее выхода в открытом доступе находится ее черновик.

Помимо Спецификации существует также справочник MDN (Mozilla) JavaScript Reference, который содержит в себе примеры и другую полезную информацию о функциях языка, методах встроенных объектов и так далее.

JavaScript предоставляет множество возможностей для разработки веб-приложений, такие как функциональное программирование, объектно-ориентированное программирование (ООП), асинхронное программирование, а также разнообразные API.

Если говорить о функциональном программировании в JavaScript, то данный язык программирования поддерживает функции как первоклассные объекты, что позволяет использовать функциональные подходы к программированию. Замыкания и стрелочные функции упрощают работу с функциями и контекстом выполнения. К примеру, функция высшего порядка, принимающая другую функцию в качестве аргумента, представлена ниже:

* function greet(name) {

return function(message) {

console.log(`${message}, ${name}`);

};

}

const greetJohn = greet('John');

greetJohn('Hello');

const sum = (a, b) => a + b;

console.log(sum(3, 4));

В рамках ООП JavaScript поддерживает прототипное наследование, что позволяет создавать объекты и классы. ES6 (ECMAScript 2015) добавил поддержку классов и улучшил синтаксис для работы с объектно-ориентированным программированием. Базовый пример создания касса в ES6 и наследования представлен ниже:

* class Person {

constructor(name, age) {

this.name = name;

this.age = age;

}

greet() {

console.log(`Hello, I am ${this.name} and I am ${this.age} years old.`);

}

}

class Student extends Person {

constructor(name, age, grade) {

super(name, age);

this.grade = grade;

}

study() {

console.log(`${this.name} is studying in ${this.grade}.`);

}

}

const student = new Student('Olga', 29, 'GeekBrains');

student.greet(); // Вывод: Hello, I am Olga and I am 29 years old.

student.study(); // Вывод: Olga is studying in GeekBrains.

Современный JavaScript также поддерживает асинхронные операции через колбэки, промисы и async/await. Это позволяет обрабатывать отложенные по времени задачи, такие как загрузка данных, без блокировки основного потока выполнения. К примеру:

* function fetchData() {

return new Promise((resolve, reject) => {

setTimeout(() => {

const data = { id: 1, name: 'John Doe' };

resolve(data);

}, 2000);

});

}

async function displayData() {

try {

const data = await fetchData();

console.log(data); // Вывод: { id: 1, name: 'John Doe' }

} catch (error) {

console.error('Error fetching data:', error);

}

}

displayData();

JavaScript имеет доступ к различным API, предоставляемым браузерами:

1. API для работы с графикой: Canvas API и WebGL позволяют создавать и отображать 2D и 3D графику в браузере. Это открывает возможности для создания игр, анимаций и визуализаций.
2. API для работы с мультимедиа: JavaScript может управлять воспроизведением аудио и видео, используя встроенные теги HTML5 и соответствующие API. Это позволяет создавать мультимедийные приложения и интерактивный контент.
3. API для работы с данными: Fetch API и XMLHttpRequest позволяют загружать данные с сервера и отправлять запросы. Эти API используются для реализации AJAX и интеграции с внешними сервисами.

Для упрощения разработки на JavaScript используются различные инструменты и фреймворки, к примеру, такие как Node.js, NPM, Babel.

Node.js - это платформа, которая позволяет выполнять JavaScript на серверной стороне. Node.js предоставляет возможности для создания высокопроизводительных серверных приложений и микросервисов.

NPM - это менеджер пакетов для JavaScript, который позволяет устанавливать и управлять зависимостями. NPM является стандартным инструментом для работы с библиотеками и фреймворками.

Babel - транспайлер, который позволяет использовать современные возможности JavaScript, даже если они еще не поддерживаются браузерами. Babel преобразует код в совместимый с более старыми версиями JavaScript.

Таким образом, JavaScript регламентируется Спецификацией ECMA-262 и является мощным и универсальным инструментом для разработки веб-приложений. Его возможности позволяют создавать интерактивные и динамические интерфейсы, улучшая пользовательский опыт и функциональность веб-страниц. JavaScript используется в браузерах, в серверном программировании, в разработке мобильных и настольных приложений, а также для многих других целей. В данной части были проанализированы манипуляция DOM, обработка событий и валидация форм AJAX, а также приведены примеры функционального программирования на JavaScript, объектно-ориентированного программирования на JavaScript, асинхронного программирования на JavaScript. Полезными инструментами для работы с JavaScript являются Node.js, NPM, Babel, назначение которых было также изучено в данной работе.

2.5. Сравнение популярных JavaScript-фреймворков: Angular, Vue, React.

Веб-разработка значительно изменилась с появлением JavaScript-фреймворков, которые предоставляют структурированные подходы и инструменты для создания сложных приложений. Среди множества фреймворков наиболее популярными являются Angular, Vue и React. Эти фреймворки обладают своими уникальными особенностями и подходами, что делает их подходящими для различных типов проектов.

Angular — это мощный и комплексный фреймворк, разработанный Google. Он используется для создания одностраничных приложений (SPA) и отличается своей архитектурой и функциональностью.

К его особенностям можно отнести следующие:

1. Модульная архитектура - Angular использует модульный подход, который облегчает организацию кода и его повторное использование.
2. Двунаправленная привязка данных - Angular поддерживает двунаправленную привязку данных, что позволяет автоматически синхронизировать модель и представление.
3. Dependency Injection (DI) - встроенная система DI упрощает управление зависимостями и улучшает тестируемость кода.
4. TypeScript - Angular написан на TypeScript, что добавляет типизацию и улучшает читаемость и поддержку кода.
5. RxJS - Angular интегрирован с библиотекой RxJS для работы с асинхронными операциями, что улучшает управление потоками данных.

В связи с этими особенностями преимуществами Angular можно назвать такие:

1. Высокая производительность и оптимизация.
2. Поддержка от Google и активное сообщество.
3. Множество встроенных инструментов и функций.

Вместе с тем данный фреймворк обладает и рядом недостатков, а именно:

1. Высокая сложность и крутая кривая обучения.
2. Большой размер фреймворка.

Таким образом, Angular заслуженно называется мощным фреймворком, при этом он является довольно сложным в разработке, поддержании и фиксировании ошибок.

Vue — это прогрессивный фреймворк, разработанный Эваном Ю. Vue ориентирован на простоту и гибкость, что делает его популярным среди разработчиков всех уровней.

Особенностями данного фреймворка являются следующие:

1. Реактивность: Vue использует реактивную систему, которая автоматически обновляет представление при изменении данных.
2. Компонентный подход: Vue позволяет создавать переиспользуемые компоненты, что улучшает модульность и организованность кода.
3. Легковесность: Vue обладает небольшим размером и высокой производительностью.
4. Простота интеграции: Vue можно легко интегрировать в существующие проекты и использовать частично или полностью.

К преимуществам Vue можно отнести такие характеристики этого фреймворка:

1. Простота освоения и использования.
2. Высокая гибкость и настраиваемость.
3. Активное сообщество и отличная документация.

Недостатки:

1. Меньшее количество встроенных функций по сравнению с Angular.
2. Меньшая корпоративная поддержка по сравнению с Angular.

Исходя из вышесказанного, можно сделать вывод о том, что Vue стал ответом на серьезный недостаток предыдущего фреймворка Angular, а именно его сложность, однако на текущий момент стать полноценной для него заменой все не смог.

И наконец, третьим фреймворком, на текущие дни являющимся самым популярным как среди программистов, так и работодателей, является React. Данный фреймворк представляет собой популярную библиотеку для создания пользовательских интерфейсов, разработанная Facebook (деятельность компании Facebook в Российской Федерации признана экстремистской и запрещена). React используется для создания компонентных, реактивных и высокопроизводительных веб-приложений. Именно его особенности и преимущества делают его одним из самых популярных инструментов в современной веб-разработке.

К данным особенностям React можно отнести такие, как:

1. Компонентный подход: React позволяет создавать приложения из компонентов — независимых, переиспользуемых частей интерфейса. Каждый компонент может содержать свою логику и состояние, что облегчает разработку и поддержку кода.
2. Virtual DOM: React использует виртуальный DOM для повышения производительности. Виртуальный DOM — это легковесная копия реального DOM, которая позволяет минимизировать количество операций обновления реального DOM, что значительно ускоряет работу приложения.
3. JSX: React использует JSX — синтаксис, который позволяет писать HTML-подобный код внутри JavaScript. JSX значительно упрощает создание компонентов и улучшает читаемость кода.
4. Однонаправленный поток данных: React использует однонаправленный поток данных, что означает, что данные передаются от родительских компонентов к дочерним через свойства (props).

Преимущества React таковы:

1. Высокая производительность благодаря Virtual DOM.
2. Гибкость и мощность компонентного подхода.
3. Широкое распространение и поддержка от Facebook.
4. Большая экосистема и множество вспомогательных библиотек.

К недостаткам React можно отнести следующие:

1. Потребность в дополнительных библиотеках для решения некоторых задач (например, маршрутизация, управление состоянием).
2. Некоторая сложность при первоначальном освоении.

Таким образом, React смог стать той самой ‘золотой серединой’ между сложным, но функциональным Angular и простым, но не покрывающим весь объем задач Vue. Именно React позволяет осуществить переход от проекта на JavaScript к более сложному веб-приложению при необходимости.

Для более подробного сравнения трех названных фреймворков можно рассмотреть следующую таблицу:

Таблица 1.

Сравнение Angular, Vue и React.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Angular | Vue | React |
| Сложность | Самый сложный | Самый простой | Средняя сложность |
| Производи-тельность | Высокая | Высокая | Высокая |
| Гибкость | Ограниченная гибкость | Очень гибкий, легко интегрируется в существующие проекты | Гибкий, но требует настройку и дополнительные библиотеки |
| Сообщество и поддержка | Сильная поддержка от Google и активное сообщество | Активное сообщество, но меньшая корпоративная поддержка | Широкое распространение, поддержка от Facebook |

Таким образом, Таблица 1 наглядно демонстрирует разницу сравниваемых фреймворков, а также может помочь в выборе между Angular, Vue и React в зависимости от конкретных требований проекта и предпочтений разработчика. Angular подходит для больших и сложных приложений, требующих мощных инструментов и строгой архитектуры. Vue лучше подходит для проектов, требующих гибкости и простоты. React является отличным выбором для создания высокопроизводительных компонентных интерфейсов и обладает широкой поддержкой и экосистемой. Немаловажным является и то, что React позволяет с относительно небольшими усилиями перейти на работу с ним от проекта, написанном на JavaScript, что и будет показано и доказано в практической части данного дипломного проекта.

Подводя итоги Главе 2, был проведен обзор актуальных инструментов для веб-разработки и для дальнейшей работы над практической частью данной дипломной работы будут использованы редактор кода Visual Studio Code, система контроля версий на платформе GitHub, Chrome DevTools для тестирования.

После этого был проанализирован на базовом уровне HTML и его место в разработке веб-сайтов. Были рассмотрены основные теги HTML, в том числе семантические, а также их особенности в зависимости от принадлежности к блочными или строчным элементам.

Немаловажными для современных веб-сайтов являются каскадные таблицы стилей CSS, которые отвечают за ее визуальное оформление HTML-элементов. В данной работе были разобраны обозначения размеров, расстояний, шрифтов и других параметров в CSS, а также такие возможности CSS, как Flexbox, Grid, анимации и переходы, препроцессоры CSS и адаптивный дизайн.

После этого данная работа отразила важнейшее значение JavaScript для разработки веб-приложений. Были изучены манипуляция DOM, обработка событий и валидация форм AJAX, а также приведены примеры функционального программирования на JavaScript, объектно-ориентированного программирования на JavaScript, асинхронного программирования на JavaScript.

И наконец, в последнем пункте данной Главы 2 было приведено сравнение популярных фреймворков, значительно расширяющих возможности JavaScript, именно Angular, Vue и React. Благодаря этому сравнению для дальнейшей практической части был сделан выбор в пользу React. Таким образом, была сформирована цель на следующую Главу 3 данной дипломной работы, а именно создать такой проект, который будет простым, малозатратным по ресурсам и при этом расширяемым по мере необходимости.

Глава 3. Разработка одностраничного коммерческого веб-сайта для малого бизнеса.

3.1. Проектирование и создание HTML-разметки и ее стилизация с помощью CSS и препроцессоров.

В целях дальнейшей практической работы над разработкой коммерческого веб-сайта будет взят за основу вымышленный ресторан фастфуда под названием BURGUR CLUB. Данная компания является показательным примером малого бизнеса, который уже проверил свою эффективность на рынке и планирует развиваться. Ресторан снискал популярность среди местных жителей и сотрудников близлежащий офисов, а потому для продвижения данного сайта и качественного информирования своих клиентов его владельцы хотят быстро и недорого разработать веб-сайт, который можно будет развивать и усложнять по мере развития самого бизнеса. Подобные потребности часто встречаются среди малого бизнеса, который стремится увеличить свою онлайн-присутствие и привлечь больше клиентов через Интернет. Для таких предприятий важно иметь доступный и функциональный веб-сайт, который можно быстро создать и легко адаптировать под изменяющиеся условия и потребности рынка. В этой связи создание адаптивного одностраничного коммерческого веб-сайта для ресторана BURGUR CLUB является актуальной задачей, которая может служить примером для других малых предприятий, стремящихся к цифровой трансформации и росту.

Неслучайно данная Глава началась с основных целей компании BURGUR CLUB. Именно это определяет план работы над созданием сайта. Учитывая небольшие размеры компании и ограниченность ее бюджета на разработку, план работы будет выстроен следующим образом:

1. Разработать необходимую структуру для создаваемого сайта.
2. Создать адаптивную HTML-разметку с учетом согласованной структуры.
3. Стилизовать HTML-разметку с помощью CSS и препроцессоров с учетом согласованной структуры.
4. Добавить интерактивность посредством JavaScript с учетом согласованной структуры.
5. Перевести проект на React и добавить функциональность с учетом согласованной структуры.

В целях разработки необходимой структуры для сайта, прежде всего, необходимо отталкиваться от целей сайта. Как уже было сказано ранее, цели сайта таковы: информирование клиентов (о меню компании, интерьере и сотрудниках) и продвижение данного сайта в сети Интернет для привлечения новых клиентов. Исходя из этого, структура сайта будет содержать следующие блоки:

1. Шапка (header), в которой будет размещен логотип компании BURGUR CLUB и навигационное меню.
2. Приветственное сообщение и визуальное изображение, которое будет ассоциироваться с компанией и вызывать желание посетить данный ресторан.
3. Меню, в котором все желающие смогут ознакомиться с ассортиментом компании. Удобство использования будет достигнуто за счет сортировок и фильтров, реализованных с помощью JavaScript.
4. Слайдер с изображениями интерьера компании BURGUR CLUB, который будет реализован на JavaScript.
5. Блок с фотографиями сотрудников компании для повышения лояльности клиентов к персоналу, который также будет реализован на JavaScript.
6. Подвал (footer), в котором будут размещены ссылки на социальные сети и уведомление об авторских правах.

При разработке также важно использовать правильные HTML-теги, чтобы обеспечить семантическую разметку и улучшить SEO-оптимизацию сайта.

Таким образом, структура HTML-разметки будет выглядеть следующим образом:

1. Тег <head>:

<head>

<meta charset="UTF-8">

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

<link rel="stylesheet" href=<https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.1/css/all.min.css> integrity="sha512-DTOQO9RWCH3ppGqcWaEA1BIZOC6xxalwEsw9c2QQeAIftl+Vegovlnee1c9QX4TctnWMn13TZye+giMm8e2LwA=="

crossorigin="anonymous" referrerpolicy="no-referrer" />

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

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

<title>Burgur Club </title>

</head>

1. Тег <body>:

<body>

<header class="header">…</header>

<main class="main">…</main>

<footer class="footer">…</footer>

</body>

Полная HTML-разметка по данному проекту представлена в Приложении 1, а также размещена на Github по адресу <https://github.com/oazakharova/burgur-delivery>.

Стилизовать адаптивную HTML-разметку – следующий этап в разработке веб-сайта, который позволяет придать странице привлекательный и функциональный внешний вид. Для этого используется CSS (Cascading Style Sheets), который определяет, как элементы HTML должны отображаться на экране.

Для стилизации сайта BURGUR CLUB были выбраны следующие ключевые элементы и подходы:

1. Базовые стили для всего сайта:

* Определение основных стилей для шрифтов, отступов и цвета фона.
* Создание глобальных классов и переменных, которые можно повторно использовать в различных частях сайта.

1. Стили для шапки (header):

* Логотип и навигационное меню.
* Адаптивные стили для корректного отображения на мобильных устройствах

1. Стили для основного контента (main):

* Приветственное сообщение и визуальные элементы.
* Гибкая сетка для размещения контента

1. Стили для меню и слайдера:

* Адаптивные карточки товаров.
* Стили для слайдера с изображениями.

1. Стили для подвала (footer):

* Ссылки на социальные сети и уведомление об авторских правах.

Таким образом, стилизация HTML-разметки для сайта BURGUR CLUB с помощью CSS и препроцессоров обеспечивает современный и привлекательный внешний вид, который соответствует потребностям бизнеса и ожиданиям клиентов. Все стили для данного проекта представлены в Приложении 2, а также размещены на Github по адресу <https://github.com/oazakharova/burgur-delivery>.

Итак, на текущий момент полноценная адаптивная, но в то же время все еще статичная версия веб-сайта готова.

3.2. Добавление интерактивности с помощью JavaScript.

К моменту, когда статичная веб-версия страницы готова, а иногда и в процессе разработки, можно переходить к следующему этапу разработки сайта для компании BURGUR CLUB, а именно добавление интерактивности посредством JavaScript с учетом согласованной структуры. Это позволит улучшить пользовательский опыт, сделав сайт более динамичным и отзывчивым.

JavaScript является мощным инструментом для создания интерактивных элементов на веб-страницах. Он позволяет реагировать на действия пользователей, изменять содержимое страницы без перезагрузки и улучшать общую функциональность сайта. Для сайта BURGUR CLUB будут следующие реализованы следующие ключевые аспекты интерактивности:

1. Слайдер изображений:

* Добавление слайдера для изображений интерьера ресторана - создание слайдера для отображения интерьера ресторана позволит пользователям ознакомиться с обстановкой и дизайном заведения. Это визуальное представление играет важнейшую роль в формировании первого впечатления и повышении интереса к посещению ресторана.
* Отображение карточек сотрудников ресторана - будет использоваться аналогичный слайдер не только для показа интерьера, но и для отображения карточек сотрудников ресторана. Это поможет посетителям узнать больше о команде, создающей атмосферу BURGUR CLUB, что будет способствовать повышению доверия и лояльности клиентов.
* Реализация возможности переключения следующего и предыдущего слайда - для удобства навигации пользователям будет предоставлена возможность переключения слайдов вперед и назад. Это можно реализовать с помощью кнопок управления слайдером, что обеспечит интуитивно понятный интерфейс

2. Фильтрация и сортировка меню:

* Интерактивная фильтрация блюд по категориям - пользователи смогут фильтровать меню по различным категориям, таким как "Бургеры", "Напитки", "Закуски" и т.д. Это сделает процесс поиска нужного блюда более удобным и быстрым. JavaScript позволит динамически изменять отображение элементов меню в зависимости от выбранной категории, без перезагрузки страницы
* Возможность сортировки меню по цене или популярности - функция сортировки меню позволит пользователям упорядочить блюда по цене (от низкой к высокой и наоборот). Это особенно полезно для тех, кто ищет бюджетные варианты. Реализация этой функции также будет выполнена с помощью JavaScript, что обеспечит мгновенный отклик и удобство использования.

Добавление интерактивности с помощью JavaScript значительно повышает функциональность и привлекательность сайта BURGUR CLUB. Все реализованные элементы помогут создать удобный и современный сайт, который будет способствовать привлечению новых клиентов и поддержке лояльности существующих. Полная реализация интерактивных элементов представлена в Приложении 3, а также размещена на Github по адресу <https://github.com/oazakharova/burgur-delivery>.

Таким образом, на данном этапе уже можно говорить о наличии полноценной страницы, которую можно показывать клиентам компании.

3.3. Перевод проекта на React: архитектура и компоненты.

Перевод проекта на React представляет собой значимый шаг в развитии веб-приложения для компании BURGUR CLUB. React позволяет разрабатывать масштабируемые и эффективные пользовательские интерфейсы, основанные на компонентах. В контексте создания веб-сайта для ресторана фастфуда, React помогает управлять динамическим содержимым, обеспечивать переиспользование компонентов и улучшать общую структуру проекта.

Основные шаги перевода проекта на React:

1. Подготовка окружения:

* Установка Node.js и npm (Node Package Manager), если они ещё не установлены.
* Создание нового проекта React или интеграция React в существующий проект.

2. Интеграция React в проект:

* Создание структуры каталогов для React-приложения, если она отличается от текущей структуры проекта.
* Добавление файловых настроек (например, .babelrc, webpack.config.js), если требуется.

3. Компоненты и их структура:

* Выделение частей интерфейса в компоненты React, такие как Header, Menu, Slider и др.

4. Использование состояний и эффектов:

* Применение useState и useEffect для управления состоянием компонентов и выполнения побочных эффектов.

5. Управление данными и пропсами:

* Передача данных между компонентами с использованием пропсов.

6. Интеграция стилей и CSS:

* Интеграция существующих стилей CSS.

7. Тестирование и отладка:

* Проверка функциональности переведенного приложения, включая тестирование пользовательского интерфейса и логики компонентов.
* Использование инструментов разработчика для отладки (например, React Developer Tools).

8. Оптимизация производительности:

* Оценка и улучшение производительности приложения, если это необходимо, с помощью инструментов анализа производительности.

Перевод проекта на React требует систематического подхода и понимания основ React-разработки. Этот процесс способствует улучшению масштабируемости, поддержки и производительности веб-приложения, делая его более современным и гибким для будущих изменений и расширений.

Перевод проекта на React предполагает следующую архитектурную организацию:

1. Структура каталогов и файлов:

src/: Основная директория проекта.

* components/: Компоненты React, включая Header, Footer, Menu, OurRestaurant, OurTeam, Slogan.
* styles/: Файлы стилей, включая CSS или SCSS для компонентов.
* App.js: Основной файл приложения, где происходит рендеринг основных компонентов.
* index.js: Точка входа в приложение, где происходит инициализация React и рендеринг корневого компонента.

2. Использование компонентов:

* Header: включает логотип и навигационное меню.
* Menu: отображает меню с возможностью фильтрации и сортировки.
* OurRestaurant: показывает изображения интерьера ресторана.
* OurTeam: показывает карточки сотрудников.
* Footer: включает ссылки на социальные сети и уведомление об авторских правах.

3. Управление состоянием:

* Использование useState и useEffect для управления состоянием компонентов.

На данном этапе важно перевести весь проект на React без улучшений и глобальных изменений для того, чтобы продемонстрировать, что это не составит больших проблем и неудобств, а потому к этому моменту код практически не изменился. Полная реализация компонентов и функционала представлена в Приложении 4, а также доступна на GitHub по адресу <https://github.com/oazakharova/burgur-delivery> в ветке react.

Таким образом, в этой части дипломной работы удалось доказать на практике, что перевод проекта с JavaScript на React прост и не требует больших усилий со стороны программиста. Сайт остался таким же функциональным, однако вместе с тем теперь он имеет гораздо больший потенциал для трансформаций и изменений.

3.4. Реализация функционала переключения светлой/темной темы.

К моменту, когда проект переведен на React и для пользователя функционирует без ошибок как ранее, добавление нового функционала становится простой задачей.

В качестве примера расширения функционала будет внедрено переключение светлой/темной темы. Это позволит пользователю сайта менять цвета и стили всех компонентов в зависимости от выбранной темы одним кликом.

Реализация данного функционала в React подразумевает создание нового компонента ThemeToggle и включает в себя следующие этапы:

1. Реализация хранения текущей темы:

* Состояние текущей темы (темная или светлая) хранится в useState.

2. Переключение темы:

* Создание функции, которая изменяет состояние текущей темы от темной к светлой и наоборот.

3. Применение темы к компонентам:

* Использование CSS-классов для применения различных стилей в зависимости от текущей темы.

Реализация компонента ThemeToggle представлена в Приложении 5, а также а также размещена на Github по адресу <https://github.com/oazakharova/burgur-delivery>. Итоговый вариант App.js можно найти в Приложении 6, а также на Github по адресу <https://github.com/oazakharova/burgur-delivery> в ветке react.

Итак, на странице была добавлена новая возможность по переключению темы, что наглядно продемонстрировало широкие возможности React по масштабированию сайта. При этом у сайта остался большой потенциал для развития, к примеру, можно добавить использование React Router для управления навигацией по различным страницам и отображению соответствующих компонентов без перезагрузки страницы, глобальнее поработать со слайдером и вынести его в отдельный компонент, реализовать модальные окна для подтверждения действий, отображения мощных и валидируемых форм или дополнительной информации, а также всплывающие подсказки для предоставления контекстной информации или помощи пользователям и многое другое.

Таким образом, подводя итоги к Главе 3, можно сделать вывод о том, что для компании BURGUR CLUB был создан сайт, с которого можно начать продвижение в сети Интернет, а также которого будет достаточно для информирования клиентов об ассортименте, сотрудниках и интерьере. Были использованы такие инструменты как HTML, CSS, в том числе препроцессоры, JavaScript и React.

Заключение.

В ходе выполнения данной дипломной работы была достигнута основная цель – разработка современного одностраничного веб-приложения для малого бизнеса на примере ресторана фастфуда BURGUR CLUB. Этот проект позволил не только глубже понять текущие тенденции в области веб-разработки, но и наглядно продемонстрировал практическое применение современных инструментов и технологий. Рассмотренные методы и этапы разработки показали, как эффективно использовать существующие ресурсы для создания функционального и эстетически привлекательного веб-приложения.

В первой главе были изучены основы современных веб-сайтов, их эволюция, виды и основные элементы. Исторический обзор позволил лучше понять, как изменялись технологии и подходы к разработке веб-сайтов, а также какие современные практики являются наиболее эффективными. Были рассмотрены ключевые сетевые компоненты и протоколы, обеспечивающие функционирование веб-сайтов на сегодня.

Вторая глава была посвящена анализу актуальных инструментов для разработки веб-сайтов. Были изучены возможности HTML, CSS и JavaScript, а также проведено сравнение популярных JavaScript-фреймворков: Angular, Vue и React. Особое внимание было уделено React как наиболее подходящему инструменту для создания интерактивных и динамичных веб-приложений, по мнению автора данной дипломной работы.

Третья глава описывала практическую часть работы – разработку одностраничного веб-приложения для BURGUR CLUB. Процесс начался с проектирования и создания адаптивной HTML-разметки и ее стилизации с помощью CSS и препроцессоров. Далее было добавлено интерактивное поведение с использованием JavaScript, что позволило улучшить пользовательский опыт и сделать сайт более отзывчивым. Затем проект был переведен на React, что значительно упростило архитектуру и позволило внедрить дополнительные функциональные возможности, такие как переключение светлой и темной темы. Данный сайт однозначно имеет большой потенциал на развитие. В будущем данное приложение можно развивать и усложнять, добавляя новые функциональные возможности, такие как управление навигацией по различным страницам с использованием React Router без перезагрузки страницы, улучшение работы слайдера, реализация модельных окон, форм, всплывающих подсказок и многое другое.

Основные выводы, сделанные в ходе выполнения работы, заключаются в следующем:

1. Современные технологии веб-разработки позволяют создавать мощные и удобные инструменты для малого бизнеса, которые не требуют больших затрат.
2. Использование одностраничных веб-приложений (SPA) позволяет значительно улучшить пользовательский опыт за счет минимизации перезагрузок страниц и оптимизации работы с данными.
3. React является мощным и гибким инструментом, подходящим для разработки динамичных и масштабируемых веб-приложений. Его компонетный подход позволяет легко расширять и поддерживать код.
4. Интеграция интерактивных элементов, таких как слайдеры и фильтрация меню, делает сайт более привлекательным и удобным для пользователей.

Разработанное веб-приложение BURGUR CLUB демонстрирует, как можно эффективно использовать современные технологии для создания полезных и доступных решений для малого бизнеса. Данный проект может служить примером для других компаний, стремящихся улучшить свою онлайн-присутствие и привлечь больше клиентов через Интернет.

Таким образом, данная дипломная работа представляет собой комплексное исследование и практическую реализацию современного веб-приложения, которое может значительно улучшить бизнес-процессы и взаимодействие с клиентами для малого бизнеса.

**Список используемой литературы.**

1. Книга: Бибер, Б. "React и Redux. Функциональная веб-разработка". СПб.: Питер, 2018. - 288 с.
2. Книга: Бос, Б. и Лие, Х.В. "Каскадные таблицы стилей: дизайн для профессионалов". СПб.: БХВ-Петербург, 2009. - 264 с.
3. Книга: Дакетт, Дж. "HTML и CSS. Разработка и дизайн веб-сайтов". М.: Эксмо, 2014. - 480 с.
4. Книга: Дакетт, Дж. "JavaScript и jQuery. Разработка интерактивных веб-сайтов". М.: Эксмо, 2014. - 640 с.
5. Книга: Дюк, Д., Линдерс, Б. и Блуд, Э. "HTML5 и CSS3. Разработка современных веб-сайтов". М.: Диалектика, 2017. - 320 с.
6. Книга: Кантей, Б. и Клементи, Л. "HTML5. Недостающее руководство". М.: Символ-Плюс, 2013. - 600 с.
7. Книга: Кинни, Д. "React и Redux: функциональная разработка". М.: ДМК Пресс, 2018. - 352 с.
8. Книга: Крокфорд Дуглас. Как устроен JavaScript. — СПб.: Питер, 2019. — 304 с. — (Серия «Для профессионалов»).
9. Книга: Макфарланд Д. Большая книга CSS3. 3-е изд. — СПб.: Питер, 2014. — 608 с.: ил. — (Серия «Бестселлеры O’Reilly»).
10. Книга: Мерник, С. "Изучаем React". СПб.: Питер, 2019. - 512 с.
11. Книга: Прохоренок Н. А. Bootstrap и CSS-препроцессор Sass. Самое необходимое. — СПб.: БХВ-Петербург, 2021. — 496 с.: ил. — (Самое необходимое)
12. Книга: Резиг, Дж. и Джоселин, А. "JavaScript. Карманный справочник". М.: Диалектика, 2018. - 162 с.
13. Книга: Роббинс, Дж. "Изучаем CSS3". М.: Вильямс, 2017. - 320 с.
14. Книга: Симкин, Т. и Матвейчук, Е. "JavaScript и jQuery. Исчерпывающее руководство". СПб.: БХВ-Петербург, 2019. - 880 с.
15. Книга: Томас Марк Тиленс. React в действии. — СПб.: Питер, 2019. — 368 с.: ил. — (Серия «Для профессионалов»).
16. Книга: Флэнаган, Д. "JavaScript: Подробное руководство". М.: Символ-Плюс, 2020. - 1184 с.
17. Книга: Шмидт, С. и Манн, Э. "Node.js и JavaScript. Руководство веб-разработчика". СПб.: БХВ-Петербург, 2017. - 352 с.
18. Книга: Яковлев, В.В. и Глухова, Т.А. "Основы HTML5 и CSS3". М.: Юрайт, 2017. - 244 с.
19. Сайт // <https://angularjs.org/> [2024]. Дата обновления: 15.06.2024. URL: https://angularjs.org / (дата обращения: 15.06.2024).
20. Сайт // <https://ecma-international.org/publications-and-standards/standards/ecma-262/> [2024]. Дата обновления: 15.06.2024. URL: https://ecma-international.org / (дата обращения: 15.06.2024).
21. Сайт // <https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference> [2024]. Дата обновления: 15.06.2024. URL: https://developer.mozilla.org/ (дата обращения: 15.06.2024).
22. Сайт // htmlbook.ru. [2024]. Дата обновления: 14.06.2024. URL: https://htmlbook.ru/ (дата обращения: 14.06.2024).
23. Сайт // <https://ru.react.js.org/> [2024]. Дата обновления: 15.06.2024. URL: https://ru.react.js.org/ (дата обращения: 15.06.2024).
24. Сайт // <https://ru.vuejs.org/> [2024]. Дата обновления: 15.06.2024. URL: https://ru.vuejs.org/ (дата обращения: 15.06.2024).
25. Сайт // Библиотека Максима Мошкова. [2024]. Дата обновления: 14.06.2024. URL: http://lib.ru/ (дата обращения: 14.06.2024).
26. Сайт // Википедия. [2024]. Дата обновления: 25.05.2024. URL: https://ru.wikipedia.org/?curid=42154&oldid=138002275/(дата обращения: 25.05.2024).

Приложение 1.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.1/css/all.min.css"

integrity="sha512-DTOQO9RWCH3ppGqcWaEA1BIZOC6xxalwEsw9c2QQeAIftl+Vegovlnee1c9QX4TctnWMn13TZye+giMm8e2LwA=="

crossorigin="anonymous" referrerpolicy="no-referrer" />

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

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

<title>Burgur</title>

</head>

<body>

<header class="header">

<div class="container">

<a class="logo" href="index.html">burgur club</a>

<nav class="nav">

<a class="anchor" href="#menu">menu</a>

<a class="anchor" href="#ourRestaurant">our restaurant</a>

<a class="anchor" href="#ourTeam">our team</a>

</nav>

</div>

</header>

<main class="main">

<div class="container">

<div class="sloganWrap">

<div class="slogan">

<div>THE ULTIMATE</div>

<div>Burger Club </div>

<div>Savor the Flavor, Join the Club!</div>

</div>

<img class="burgerImg" src="images/mainBurger.png" alt="burger">

</div>

<section class="section menuSection">

<h2><a name="menu">menu</a></h2>

<div class="underline"></div>

<div class="buttonsWrap">

</div>

<div class="sortWrap">

<button class="filterButton lowToHighPriceButton" type="button" data-id="sortLowToHigh">

<span>low to high</span>

<i class="fa-solid fa-arrow-up-wide-short"></i>

</button>

<button class="filterButton highToLowPriceButton" type="button" data-id="sortHighToLow">

<span>high to low</span>

<i class="fa-solid fa-arrow-down-wide-short"></i>

</button>

</div>

<div class="menuItemsWrap">

</div>

</section>

<section class="section ourRestaurantSection">

<h2><a name="ourRestaurant">our restaurant</a></h2>

<div class="underline"></div>

<div class="slider-container ourRestaurantSliderWrap">

<div class="slide">

<img class="slideImg" src="images/restaurant-1-pexels-pixabay-460537.jpg" alt="restaurant photo">

<h3 class="title">1</h3>

</div>

<div class="slide">

<img class="slideImg" src="images/restaurant-pexels-pixabay-262047.jpg" alt="restaurant photo">

<h3 class="title">2</h3>

</div>

<div class="slide">

<img class="slideImg" src="images/restaurant-pexels-chan-walrus-941861.jpg" alt="restaurant photo">

<h3 class="title">3</h3>

</div>

<div class="slide">

<img class="slideImg" src="images/restaurant-pexels-quang-nguyen-vinh-2159065.jpg" alt="restaurant photo">

<h3 class="title">4</h3>

</div>

<div class="slide">

<img class="slideImg" src="images/restaurant-pexels-elle-hughes-2696064.jpg" alt="restaurant photo">

<h3 class="title">5</h3>

</div>

</div>

<div class="btn-container ourRestaurantButtonContainer">

<button class="prevBtn" type="button">

<i class="fas fa-chevron-left"></i>

</button>

<button class="nextBtn" type="button">

<i class="fas fa-chevron-right"></i>

</button>

</div>

<nav class="slider-nav">

<i class="fa-solid fa-circle"></i>

<i class="fa-regular fa-circle"></i>

<i class="fa-regular fa-circle"></i>

<i class="fa-regular fa-circle"></i>

<i class="fa-regular fa-circle"></i>

</nav>

</section>

<section class="section ourTeamSection">

<h2><a name="ourTeam">our team</a></h2>

<div class="underline"></div>

<div class="ourTeamItemsWrap">

<article class="ourTeamItem">

<div class="ourTeamImgWrap">

<img class="ourTeamImg" src="images/expressive-redhead-guy-beige-shirt.jpg" alt="person">

</div>

<h4 class="ourTeamUserName">sam smith</h4>

<h5 class="ourTeamJob">barista</h5>

<p class="ourTeamDesc">

Lorem ipsum dolor sit amet consectetur adipisicing elit. Aperiam provident laboriosam dolorem sunt dolor

dignissimos

</p>

<div class="ourTeamButtonContainer">

<button class="prevBtn">

<i class="fas fa-chevron-left"></i>

</button>

<button class="nextBtn">

<i class="fas fa-chevron-right"></i>

</button>

</div>

</article>

</div>

</section>

</div>

</main>

<footer class="footer">

<div class="container">

<div class="copyInfo">&copy; All Rights Reserved</div>

<nav class="socialsWrap">

<a href="#"><i class="fa-brands fa-facebook-f socialLink"></i></a>

<a href="#"><i class="fa-brands fa-instagram socialLink"></i></a>

<a href="#"><i class="fa-brands fa-x-twitter socialLink"></i></a>

</nav>

</div>

</footer>

</body>

</html>

Приложение 2

// файл \_variables.scss

// fonts

@import url("https://fonts.googleapis.com/css2?family=Lexend+Zetta:wght@100..900&family=Sansita+Swashed:wght@300..900&display=swap");

$fontMainLexendZetta: "Lexend Zetta", sans-serif;

$fontSansitaSwashed: "Sansita Swashed", system-ui;

// variables

$backgroundMainColor: #12372a;

$colorLightGreen: #adbc9f;

$colorLight: #fbfada;

$colorWhite: #ffffff;

// файл style.scss

@import "\_variables";

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

display: flex;

flex-direction: column;

background-color: $backgroundMainColor;

}

.header .container,

.footer .container {

min-height: 80px;

flex-shrink: 0;

}

.main {

min-height: 500px;

flex-grow: 1;

}

.container {

width: 90%;

margin: 0 auto;

}

.header {

height: 20vh;

.container {

display: flex;

flex-direction: column;

justify-content: space-between;

align-items: center;

text-transform: uppercase;

margin-top: 20px;

margin-bottom: 200px;

@media (min-width: 1600px) {

flex-direction: row;

}

.logo {

margin-bottom: 15px;

padding: 5px;

text-decoration: none;

color: $colorLight;

font-family: $fontMainLexendZetta;

font-weight: 700;

letter-spacing: 0;

font-size: 25px;

@media (min-width: 768px) {

font-size: 50px;

}

@media (min-width: 768px) {

margin-bottom: 30px;

}

@media (min-width: 1600px) {

margin-bottom: 0;

}

}

.nav {

display: flex;

justify-content: center;

align-items: center;

}

.anchor {

padding: 5px;

font-family: $fontMainLexendZetta;

color: $colorLight;

font-weight: 400;

letter-spacing: 0;

font-size: 12px;

text-decoration: none;

@media (min-width: 768px) {

font-size: 25px;

}

}

.anchor:not(:last-child) {

margin-right: 15px;

}

}

}

.main {

.container .sloganWrap {

display: flex;

flex-direction: column;

justify-content: center;

align-items: center;

text-transform: uppercase;

@media (min-width: 1600px) {

flex-direction: row;

}

.slogan {

display: flex;

flex-direction: column;

justify-content: space-between;

align-items: center;

text-transform: uppercase;

text-align: center;

color: $colorLight;

font-family: $fontMainLexendZetta;

font-weight: 700;

letter-spacing: 0;

font-size: 45px;

margin-bottom: 40px;

@media (min-width: 1600px) {

font-size: 80px;

}

@media (min-width: 1600px) {

font-size: 100px;

}

}

.slogan div:first-child {

color: $colorLightGreen;

font-size: 20px;

@media (min-width: 768px) {

font-size: 40px;

}

@media (min-width: 1600px) {

font-size: 50px;

}

}

.slogan div:last-child {

color: $colorLightGreen;

font-size: 10px;

@media (min-width: 768px) {

font-size: 20px;

}

}

.burgerImg {

display: block;

width: 100%;

@media (min-width: 768px) {

max-width: 1000px;

}

}

}

.section {

display: flex;

flex-direction: column;

justify-content: center;

align-items: center;

color: $colorLight;

text-transform: uppercase;

font-family: $fontMainLexendZetta;

font-weight: 700;

letter-spacing: 0;

font-size: 20px;

margin-bottom: 100px;

h2 {

text-align: center;

}

}

.buttonsWrap {

display: flex;

flex-direction: row;

flex-wrap: wrap;

justify-content: center;

align-items: center;

margin-bottom: 20px;

}

.filterButton {

background: none;

font-size: 16px;

font-weight: 400;

padding: 3px 15px;

margin: 5px;

color: $colorLight;

font-family: $fontMainLexendZetta;

letter-spacing: 0;

border: 1px solid $colorLight;

border-radius: 3px;

transition-duration: 0.5s;

@media (min-width: 768px) {

margin: 10px 20px;

}

}

.filterButton:hover {

transform: scale(1.1, 1.1);

}

.selectedFilterButton {

background-color: $colorLight;

color: $backgroundMainColor;

}

.sortWrap {

margin-bottom: 50px;

i {

font-weight: 600;

}

span {

margin-right: 5px;

}

}

.menuItemsWrap {

display: grid;

gap: 30px;

@media (min-width: 768px) {

grid-template-columns: repeat(2, 1fr);

gap: 40px;

}

@media (min-width: 1600px) {

grid-template-columns: repeat(3, 1fr);

gap: 50px;

}

}

.menuItem {

display: flex;

flex-direction: column;

align-items: center;

@media (min-width: 1200px) {

flex-direction: row;

align-items: flex-start;

}

.photo {

display: block;

width: 150px;

height: 150px;

border: 5px solid $colorLightGreen;

border-radius: 3px;

object-fit: cover;

}

.itemInfo {

width: 80%;

header {

display: flex;

flex-direction: column;

align-items: center;

justify-content: space-between;

margin: 15px auto;

@media (min-width: 1200px) {

padding-left: 15px;

margin-top: 0;

flex-direction: row;

align-items: flex-start;

}

.itemTitle {

@media (max-width: 1199px) {

text-align: center;

}

}

.price {

color: $colorWhite;

}

}

.itemText {

font-size: 14px;

font-weight: 400;

text-align: center;

@media (min-width: 1200px) {

text-align: start;

padding-left: 15px;

}

}

}

}

}

.underline {

width: 20%;

height: 3px;

background-color: $colorLightGreen;

margin: 20px auto 50px;

}

.ourTeamItemsWrap {

background-color: darken($color: $colorLightGreen, $amount: 5%);

padding: 5px;

box-shadow: 0px 0px 15px 1px $colorLightGreen;

.ourTeamItem {

display: flex;

justify-content: center;

flex-direction: column;

align-items: center;

width: 280px;

height: 500px;

@media (min-width: 768px) {

width: 500px;

}

@media (min-width: 1600px) {

width: 800px;

}

}

.ourTeamImgWrap {

background: lighten($color: $backgroundMainColor, $amount: 2%);

border-radius: 100% 70% 95% 65%;

margin-top: 20px;

}

.ourTeamImg {

display: block;

width: 150px;

height: 150px;

border-radius: 50%;

object-fit: cover;

margin: 20px auto;

}

.ourTeamUserName {

color: $backgroundMainColor;

margin: 5px auto;

text-align: center;

}

.ourTeamJob {

margin-bottom: 15px;

text-align: center;

}

.ourTeamDesc {

color: $colorWhite;

font-weight: 400;

font-size: 12px;

text-align: center;

margin: 5px auto;

@media (min-width: 768px) {

font-size: 16px;

}

@media (min-width: 1600px) {

font-size: 18px;

}

}

.prevBtn,

.nextBtn {

background: none;

border: none;

color: $backgroundMainColor;

font-size: 26px;

padding: 10px;

}

}

.ourRestaurantSection {

position: relative;

.ourRestaurantSliderWrap {

width: 100%;

height: 300px;

overflow: hidden;

position: relative;

border: 5px solid $colorLightGreen;

border-radius: 3px;

@media (min-width: 768px) {

height: 500px;

}

@media (min-width: 1600px) {

height: 800px;

}

@media (min-width: 2000px) {

height: 1200px;

}

.slide {

display: grid;

place-items: center;

height: 100%;

width: 100%;

position: absolute;

display: grid;

place-items: center;

transition: all 0.25s ease-in-out;

text-align: center;

}

/\*.slide:nth-child(1) {

left: 0;

}

.slide:nth-child(2) {

left: 100%;

}

.slide:nth-child(2) {

left: 200%;

}

.slide {

transform: translateX(-300%);

}\*/

.slideImg {

display: block;

width: 100%;

height: 100%;

object-fit: cover;

object-position: center center;

}

.title {

position: absolute;

bottom: 10%;

left: 50%;

transform: translate(-50%, -50%);

}

}

.ourRestaurantButtonContainer {

width: 100%;

position: absolute;

top: 55%;

display: flex;

justify-content: space-between;

font-weight: 900;

}

.prevBtn,

.nextBtn {

background: none;

border: 2px solid $backgroundMainColor;

border-radius: 5px;

color: $backgroundMainColor;

padding: 5px 10px;

margin: 50px;

transition-duration: 0.5s;

i {

font-size: 26px;

@media (min-width: 768px) {

font-size: 36px;

}

@media (min-width: 1600px) {

font-size: 56px;

}

}

}

.prevBtn:hover,

.nextBtn:hover {

transform: scale(1.2, 1.2);

}

}

.slider-nav {

margin-top: 20px;

i {

font-size: 16px;

}

}

.footer {

.container {

display: flex;

flex-direction: column;

justify-content: space-between;

align-items: center;

text-transform: uppercase;

color: $colorLight;

padding: 5px;

@media (min-width: 1600px) {

flex-direction: row;

}

.copyInfo {

padding: 5px;

font-family: $fontMainLexendZetta;

color: $colorLight;

font-size: 10px;

}

.socialsWrap a {

color: $colorLight;

text-decoration: none;

padding: 5px;

i {

transition-duration: 0.5s;

}

i:hover {

transform: scale(1.3, 1.3);

}

}

.socialsWrap a:not(:last-child) {

margin-right: 10px;

}

}

}

Приложение 3

"use strict";

const menu = [

{

id: 1,

title: "cheeseburger",

category: "burgers",

price: 15.99,

img: "./images/cheeseburger.jpg",

desc: "Lorem ipsum dolor sit, amet consectetur adipisicing elit. Architecto nobis repellat dicta earum beatae! Nihil.",

},

{

id: 2,

title: "hamburger",

category: "burgers",

price: 15.99,

img: "./images/hamburger.jpg",

desc: "Lorem ipsum dolor sit, amet consectetur adipisicing elit. Architecto nobis repellat dicta earum beatae! Nihil.",

},

{

id: 3,

title: "italianburger",

category: "burgers",

price: 18.99,

img: "./images/italianburger.jpg",

desc: "Lorem ipsum dolor sit, amet consectetur adipisicing elit. Architecto nobis repellat dicta earum beatae! Nihil.",

},

{

id: 4,

title: "turkburger",

category: "burgers",

price: 16.99,

img: "./images/turkburger.jpg",

desc: "Lorem ipsum dolor sit, amet consectetur adipisicing elit. Architecto nobis repellat dicta earum beatae! Nihil.",

},

{

id: 5,

title: "greekburger",

category: "burgers",

price: 17.99,

img: "./images/greekburger.jpg",

desc: "Lorem ipsum dolor sit, amet consectetur adipisicing elit. Architecto nobis repellat dicta earum beatae! Nihil.",

},

{

id: 6,

title: "fries",

category: "appetizers",

price: 9.99,

img: "./images/fries.jpg",

desc: "Lorem ipsum dolor sit, amet consectetur adipisicing elit. Architecto nobis repellat dicta earum beatae! Nihil.",

},

{

id: 7,

title: "hot dog",

category: "appetizers",

price: 10.99,

img: "./images/hotdog.jpg",

desc: "Lorem ipsum dolor sit, amet consectetur adipisicing elit. Architecto nobis repellat dicta earum beatae! Nihil.",

},

{

id: 8,

title: "chicken",

category: "appetizers",

price: 12.99,

img: "./images/chicken.jpg",

desc: "Lorem ipsum dolor sit, amet consectetur adipisicing elit. Architecto nobis repellat dicta earum beatae! Nihil.",

},

{

id: 9,

title: "caprese salad",

category: "salads",

price: 12.99,

img: "./images/capreseSalad.jpg",

desc: "Lorem ipsum dolor sit, amet consectetur adipisicing elit. Architecto nobis repellat dicta earum beatae! Nihil.",

},

{

id: 10,

title: "greek salad",

category: "salads",

price: 11.99,

img: "./images/greekSalad.jpg",

desc: "Lorem ipsum dolor sit, amet consectetur adipisicing elit. Architecto nobis repellat dicta earum beatae! Nihil.",

},

{

id: 11,

title: "nonalcoholic drinks",

category: "drinks",

price: 4.99,

img: "./images/nonalcoholicDrinks.jpg",

desc: "Lorem ipsum dolor sit, amet consectetur adipisicing elit. Architecto nobis repellat dicta earum beatae! Nihil.",

},

{

id: 12,

title: "beer",

category: "drinks",

price: 6.99,

img: "./images/beer.jpg",

desc: "Lorem ipsum dolor sit, amet consectetur adipisicing elit. Architecto nobis repellat dicta earum beatae! Nihil.",

},

];

const ourTeam = [

{

id: 1,

fullName: "samantha smith",

job: "the chef",

img: "./images/portrait-beautiful-young-woman-standing-grey-wall.jpg",

text: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Aperiam provident laboriosam dolorem sunt dolor",

},

{

id: 2,

fullName: "bill anderson",

job: "sous chef",

img: "./images/portrait-beautiful-mature-blonde-bearded-guy-with-trendy-hairdo-casual-grey-shirt-smiling.jpg",

text: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Aperiam provident laboriosam dolorem sunt dolor",

},

{

id: 3,

fullName: "anna johnson",

job: "administrator",

img: "./images/pretty-smiling-joyfully-female-with-fair-hair-dressed-casually-looking-with-satisfaction.jpg",

text: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Aperiam provident laboriosam dolorem sunt dolor",

},

{

id: 4,

fullName: "susan terner",

job: "waiter",

img: "./images/indoor-portrait-stylish-teenag-girl-with-freckles-red-hair-bun-looking-camera-with-cheerful-smile.jpg",

text: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Aperiam provident laboriosam dolorem sunt dolor",

},

{

id: 5,

fullName: "alex rodridez",

job: "waiter",

img: "./images/confident-bearded-macho-man-looks-pleased-has-friendly-kind-grin-face-wears-round-spectacles-pink-jumper.jpg",

text: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Aperiam provident laboriosam dolorem sunt dolor",

},

{

id: 6,

fullName: "carmen rodridez",

job: "barista",

img: "./images/lifestyle-people-emotions-casual-concept-confident-nice-smiling-asian-woman-cross-arms-chest-confident-ready-help-listening-coworkers-taking-part-conversation.jpg",

text: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Aperiam provident laboriosam dolorem sunt dolor",

},

];

// menu variables

const menuItemsWrap = document.querySelector(".menuItemsWrap");

const buttonsWrap = document.querySelector(".buttonsWrap");

// team variables

const ourTeamSection = document.querySelector(".ourTeamSection");

const img = ourTeamSection.querySelector(".ourTeamImg");

const author = ourTeamSection.querySelector(".ourTeamUserName");

const job = ourTeamSection.querySelector(".ourTeamJob");

const info = ourTeamSection.querySelector(".ourTeamDesc");

const prevBtn = ourTeamSection.querySelector(".prevBtn");

const nextBtn = ourTeamSection.querySelector(".nextBtn");

let currentTeamItem = 0;

// slider variables

const slides = document.querySelectorAll(".slide");

const prevBtnInSlider = document

.querySelector(".ourRestaurantSection")

.querySelector(".prevBtn");

const nextBtnInSlider = document

.querySelector(".ourRestaurantSection")

.querySelector(".nextBtn");

const sliderNav = document.querySelector(".slider-nav");

const navPoints = sliderNav.querySelectorAll("i");

let counterInSlider = 0;

// DOMContentLoaded

window.addEventListener("DOMContentLoaded", () => {

// menu

displayMenu(menu);

displayMenuButtons();

// team

showEmployee();

// slider

slides.forEach(function (slide, index) {

slide.style.left = `${index \* 100}%`;

});

nextBtnInSlider.addEventListener("click", () => {

counterInSlider++;

carousel();

});

prevBtnInSlider.addEventListener("click", () => {

counterInSlider--;

carousel();

});

navPoints.forEach((navPoint, index) => {

navPoint.addEventListener("click", () => {

counterInSlider = index;

if (!navPoint.classList.contains("fa-solid")) {

navPoint.classList.add("fa-solid");

if (navPoint.classList.contains("fa-regular")) {

navPoint.classList.remove("fa-regular");

}

}

navPoints.forEach((btn) => {

if (btn !== navPoint) {

btn.className = "";

btn.classList.add("fa-circle");

btn.classList.add("fa-regular");

}

});

carousel();

});

});

});

// menu functions

const displayMenu = (menuItems) => {

let displayMenu = menuItems.map((item) => {

return `<article class="menuItem">

<img class="photo" src=${item.img} alt=${item.title}>

<div class="itemInfo">

<header>

<h4 class="itemTitle">${item.title}</h4>

<h4 class="price">$${item.price}</h4>

</header>

<p class="itemText">${item.desc}</p>

</div>

</article>`;

});

displayMenu = displayMenu.join(""); // because of "" do not get "," between articles

menuItemsWrap.innerHTML = displayMenu;

};

const displayMenuButtons = () => {

// const categories = menu.map((item) => item.category); // to get all categories

// to get unique categories:

const categories = menu.reduce(

(values, item) => {

if (!values.includes(item.category)) {

values.push(item.category);

}

return values;

},

["all"]

);

const categoryButtons = categories

.map((category) => {

return `<button class="filterButton" type="button" data-id=${category}>

${category}

</button>`;

})

.join("");

buttonsWrap.innerHTML = categoryButtons;

const filterButtons = document.querySelectorAll(".filterButton");

filterButtons.forEach((button) => {

if (button.dataset.id === "all") {

button.classList.add("selectedFilterButton");

}

button.addEventListener("click", (e) => {

const category = e.currentTarget.dataset.id;

const menuCategory = menu.filter((menuItem) => {

if (menuItem.category === category) {

return menuItem;

}

});

if (category === "all") {

displayMenu(menu);

} else {

displayMenu(menuCategory);

filterButtons.forEach((btn) => {

if (

btn !== button &&

btn.classList.contains("selectedFilterButton")

) {

btn.classList.remove("selectedFilterButton");

}

});

}

e.currentTarget.classList.add("selectedFilterButton");

return menu;

});

});

filterButtons.forEach((button) => {

button.addEventListener("click", (e) => {

e.currentTarget.classList.add("selectedFilterButton");

filterButtons.forEach((element) => {

if (element !== e.currentTarget) {

element.classList.remove("selectedFilterButton");

}

});

});

});

};

//sort filter buttons

const lowToHighPriceButtonEl = document.querySelector(".lowToHighPriceButton");

lowToHighPriceButtonEl.addEventListener("click", (e) => {

const unsortedMenu = 1; // тут нужно получать неотсортированное, но отфильтрованное меню

const sortedMenu = menu.sort((a, b) => a.price - b.price);

displayMenu(sortedMenu);

e.currentTarget.classList.add("selectedFilterButton");

highToLowPriceButtonEl.classList.remove("selectedFilterButton");

});

const highToLowPriceButtonEl = document.querySelector(".highToLowPriceButton");

highToLowPriceButtonEl.addEventListener("click", (e) => {

const sortedMenu = menu.sort((a, b) => b.price - a.price);

displayMenu(sortedMenu);

e.currentTarget.classList.add("selectedFilterButton");

lowToHighPriceButtonEl.classList.remove("selectedFilterButton");

});

// team functions

const showEmployee = () => {

const itemTeam = ourTeam[currentTeamItem];

img.src = itemTeam.img;

author.textContent = itemTeam.fullName;

job.textContent = itemTeam.job;

info.textContent = itemTeam.text;

};

nextBtn.addEventListener("click", () => {

currentTeamItem++;

if (currentTeamItem > ourTeam.length - 1) {

currentTeamItem = 0;

}

showEmployee();

});

prevBtn.addEventListener("click", () => {

currentTeamItem--;

if (currentTeamItem < 0) {

currentTeamItem = ourTeam.length - 1;

}

showEmployee();

});

// slider functions

// buttons prev and next in slider

function carousel() {

if (counterInSlider === slides.length) counterInSlider = 0;

if (counterInSlider < 0) counterInSlider = slides.length - 1;

slides.forEach((slide) => {

slide.style.transform = `translateX(-${counterInSlider \* 100}%)`;

});

}

Приложение 4

// Footer.js

import React from "react";

const Footer = () => {

return (

<footer className="footer">

<div className="container">

<div className="copyInfo">&copy; All Rights Reserved</div>

<nav className="socialsWrap">

<a href="#">

<i className="fa-brands fa-facebook-f socialLink"></i>

</a>

<a href="#">

<i className="fa-brands fa-instagram socialLink"></i>

</a>

<a href="#">

<i className="fa-brands fa-x-twitter socialLink"></i>

</a>

</nav>

</div>

</footer>

);

};

export default Footer;

// Header.js

import React from "react";

import ThemeToggle from "./ThemeToggle";

const Header = () => {

return (

<>

<header className="header">

<div className="container">

<a className="logo" href="/">

burgur club

</a>

<nav className="nav">

<a className="anchor" href="#menu">

menu

</a>

<a className="anchor" href="#ourRestaurant">

our restaurant

</a>

<a className="anchor" href="#ourTeam">

our team

</a>

</nav>

</div>

</header>

<ThemeToggle />

</>

);

};

export default Header;

// Menu.js

import React, { useState } from "react";

import { menu } from "../data";

const Menu = () => {

const [displayMenu, setDisplayMenu] = useState(menu);

const filterMenu = (category) => {

if (category === "all") {

setDisplayMenu(menu);

} else {

setDisplayMenu(menu.filter((item) => item.category === category));

}

};

const sortMenu = (order) => {

const sortedMenu = [...displayMenu].sort((a, b) =>

order === "lowToHigh" ? a.price - b.price : b.price - a.price

);

setDisplayMenu(sortedMenu);

};

return (

<section className="section menuSection">

<h2>

<a name="menu">menu</a>

</h2>

<div className="underline"></div>

<div className="buttonsWrap">

<button className="filterButton" onClick={() => filterMenu("all")}>

all

</button>

<button className="filterButton" onClick={() => filterMenu("burgers")}>

burgers

</button>

<button

className="filterButton"

onClick={() => filterMenu("appetizers")}

>

appetizers

</button>

<button className="filterButton" onClick={() => filterMenu("salads")}>

salads

</button>

<button className="filterButton" onClick={() => filterMenu("drinks")}>

drinks

</button>

</div>

<div className="sortWrap">

<button

className="filterButton lowToHighPriceButton"

onClick={() => sortMenu("lowToHigh")}

>

<span>low to high</span>

<i className="fa-solid fa-arrow-up-wide-short"></i>

</button>

<button

className="filterButton highToLowPriceButton"

onClick={() => sortMenu("highToLow")}

>

<span>high to low</span>

<i className="fa-solid fa-arrow-down-wide-short"></i>

</button>

</div>

<div className="menuItemsWrap">

{displayMenu.map((item) => (

<article key={item.id} className="menuItem">

<img className="photo" src={item.img} alt={item.title} />

<div className="itemInfo">

<header>

<h4>{item.title}</h4>

<h4 className="price">${item.price}</h4>

</header>

<p>{item.desc}</p>

</div>

</article>

))}

</div>

</section>

);

};

export default Menu;

// OurRestaurant.js

import React, { useState } from "react";

import { restaurantImages } from "../data";

const OurRestaurant = () => {

const [currentSlide, setCurrentSlide] = useState(0);

const nextSlide = () => {

setCurrentSlide((currentSlide + 1) % restaurantImages.length);

};

const prevSlide = () => {

setCurrentSlide(

(currentSlide - 1 + restaurantImages.length) % restaurantImages.length

);

};

return (

<section className="section ourRestaurantSection">

<h2>

<a name="ourRestaurant">our restaurant</a>

</h2>

<div className="underline"></div>

<div className="slider-container ourRestaurantSliderWrap">

{/\* Отображаем текущее изображение на основе currentSlide \*/}

<div className="slide">

<img

className="slideImg"

src={restaurantImages[currentSlide].src}

alt="restaurant photo"

/>

<h3 className="title">{currentSlide + 1}</h3>

</div>

</div>

<div className="btn-container ourRestaurantButtonContainer">

<button className="prevBtn" onClick={prevSlide}>

<i className="fas fa-chevron-left"></i>

</button>

<button className="nextBtn" onClick={nextSlide}>

<i className="fas fa-chevron-right"></i>

</button>

</div>

<nav className="slider-nav">

{/\* Отображаем навигационные точки и добавляем обработчик клика \*/}

{restaurantImages.map((\_, index) => (

<i

key={index}

className={`fa-circle ${

index === currentSlide ? "fa-solid" : "fa-regular"

}`}

onClick={() => setCurrentSlide(index)}

></i>

))}

</nav>

</section>

);

};

export default OurRestaurant;

// OurTeam.js

import React, { useState } from "react";

import { ourTeam } from "../data";

const OurTeam = () => {

const [currentTeamMember, setCurrentTeamMember] = useState(0);

const nextMember = () => {

setCurrentTeamMember((currentTeamMember + 1) % ourTeam.length);

};

const prevMember = () => {

setCurrentTeamMember(

(currentTeamMember - 1 + ourTeam.length) % ourTeam.length

);

};

const { img, fullName, job, text } = ourTeam[currentTeamMember];

return (

<section className="section ourTeamSection">

<h2>

<a name="ourTeam">our team</a>

</h2>

<div className="underline"></div>

<div className="ourTeamItemsWrap">

<article className="ourTeamItem">

<div className="ourTeamImgWrap">

<img className="ourTeamImg" src={img} alt={fullName} />

</div>

<h4 className="ourTeamUserName">{fullName}</h4>

<h5 className="ourTeamJob">{job}</h5>

<p className="ourTeamDesc">{text}</p>

<div className="ourTeamButtonContainer">

<button className="prevBtn" onClick={prevMember}>

<i className="fas fa-chevron-left"></i>

</button>

<button className="nextBtn" onClick={nextMember}>

<i className="fas fa-chevron-right"></i>

</button>

</div>

</article>

</div>

</section>

);

};

export default OurTeam;

// Slogan.js

import React from "react";

const Slogan = () => {

return (

<div className="section sloganSection">

<div className="slogan">

<div>THE ULTIMATE</div>

<div>Burger Club </div>

<div>Savor the Flavor, Join the Club!</div>

</div>

<img className="burgerImg" src="images/mainBurger.png" alt="burger" />

</div>

);

};

export default Slogan;

Приложение 5

import React, { useState, useEffect } from "react";

const ThemeToggle = () => {

const [theme, setTheme] = useState("dark");

useEffect(() => {

document.documentElement.classList.add("darkTheme");

}, []);

const toggleTheme = () => {

const newTheme = theme === "light" ? "dark" : "light";

setTheme(newTheme);

document.documentElement.classList.toggle("darkTheme");

};

return (

<div className="themeToggleWrap">

<button className="toggleThemeBtn" onClick={toggleTheme}>

<i className="fa-regular fa-lightbulb"></i> change theme

</button>

</div>

);

};

export default ThemeToggle;

Приложение 6

import React from "react";

import "./styles/style.scss";

import { Helmet } from "react-helmet";

import Header from "./components/Header";

import Slogan from "./components/Slogan";

import Menu from "./components/Menu";

import OurRestaurant from "./components/OurRestaurant";

import OurTeam from "./components/OurTeam";

import Footer from "./components/Footer";

const App = () => {

return (

<>

<Helmet>

<link

rel="stylesheet"

href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.1/css/all.min.css"

integrity="sha512-DTOQO9RWCH3ppGqcWaEA1BIZOC6xxalwEsw9c2QQeAIftl+Vegovlnee1c9QX4TctnWMn13TZye+giMm8e2LwA=="

crossorigin="anonymous"

referrerpolicy="no-referrer"

/>

</Helmet>

<div className="App">

<Header />

<main className="main container">

<Slogan />

<Menu />

<OurRestaurant />

<OurTeam />

</main>

<Footer />

</div>

</>

);

};

export default App;