СОЗДАНИЕ ОДНОСТРАНИЧНОГО САЙТА

Оглавление

[1. Введение 3](#_Toc149265659)

[2. Основная часть 3](#_Toc149265660)

[2.1. Определение целей и аудитории 4](#_Toc149265661)

[2.2. Создание контента: 4](#_Toc149265662)

[2.3. Создание html-файла 5](#_Toc149265663)

[2.4. Создание js-файла 14](#_Toc149265664)

[2.5. Php-файлы 22](#_Toc149265665)

[2.6. CSS-файл 23](#_Toc149265666)

[2.7. Домен для сайта 32](#_Toc149265667)

[3. Результат 32](#_Toc149265668)

[4. Вывод 32](#_Toc149265669)

# **Введение**

Написание сайта - этот процесс, который сегодня является одним из наиболее значимых аспектов в сфере информационных технологий и интернет-маркетинга. В наше время сайты являются мощным инструментом для представления компаний, создания онлайн-присутствия и взаимодействия с аудиторией. Этот процесс сочетает в себе технические аспекты, дизайн и контент, чтобы создать уникальное виртуальное пространство, способное привлекать внимание и достигать разнообразных целей.

Создание сайта требует понимания того, что сайт - это не просто набор веб-страниц, а мощное средство для передачи информации, продажи товаров и услуг, общения с клиентами и многих других задач. В процессе разработки сайта необходимо учесть не только его внешний вид и интерфейс, но и его функциональность, безопасность, адаптивность к разным устройствам и уровням доступности.

Создание сайта может быть как личным творческим проектом, так и задачей, порученной вам корпорацией. Но в обоих случаях профессионализм и знание современных веб-технологий становятся ключевыми факторами успеха. В данной работе мы рассмотрим основные этапы и инструменты, необходимые для создания сайта, а также принципы, которые помогут вам разработать сайт, отвечающий вашим потребностям и целям.

# **Основная часть**

Для работы мы будем использовать Visual Studio Code. Это среда разработки, созданный Microsoft для Windows, Linux и macOS. Установит Visual Studio Code можно скачать по ссылке: <https://code.visualstudio.com/>

# **Определение целей и аудитории**

Определение целей и аудитории для сайта, посвященного экскурсиям по Санкт-Петербургу, является важным шагом для создания успешного веб-ресурса.

Цель нашего сайта – предоставление услуг, продвижение услуг, обратная связь и бронирование, публикация контента.

Аудитория для нашего сайта: туристы, местные жители, партнеры и предприниматели.

Понимание целей и аудитории поможет вам создать сайт, который будет соответствовать потребностям посетителей и достигать ваших целей. Учитывайте интересы и ожидания аудитории при разработке контента, дизайна и функциональности сайта.

# **Создание контента:**

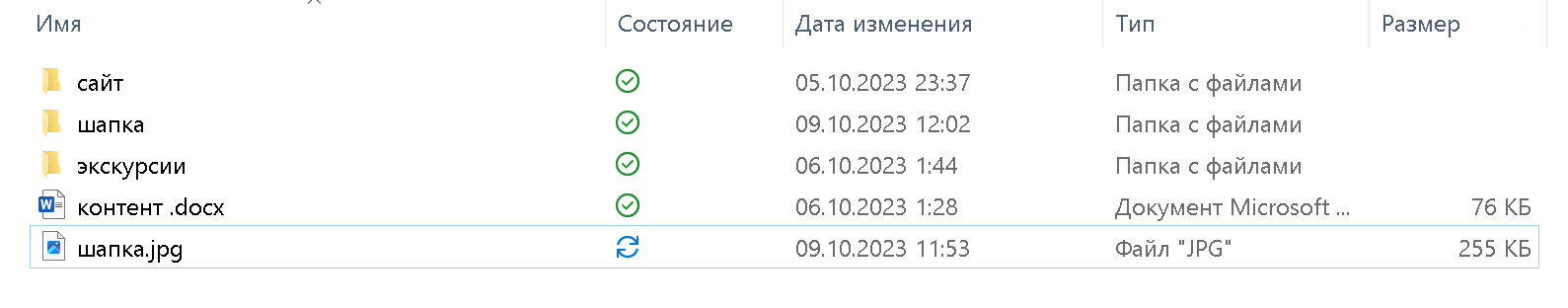
Создание контента для сайта - это важный этап процесса разработки веб-ресурса, который включает в себя создание текстов, изображений, видео и другого материала, предназначенного для привлечения и информирования аудитории.

Мы написали текст, который описывает «нашу турфирму» и придумали экскурсии и описания для них.

В сети Интернет нашли подходящие изображения и сохранили их.

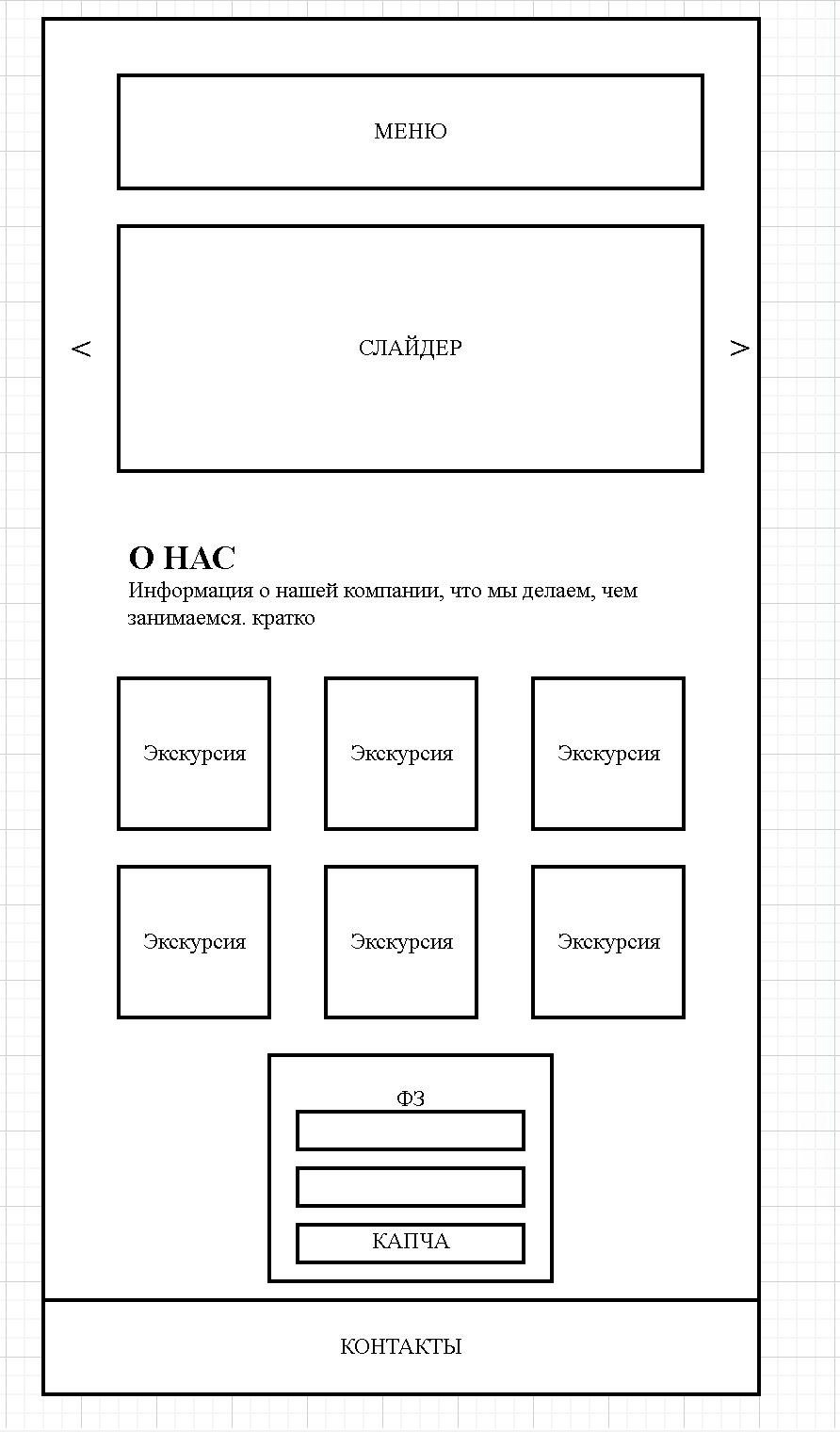
Также создали собственный логотип с помощью бесплатного сервиса Canva, но также можно использовать и другие ресурсы, например logomaster.

Для удобства все материалы сложили в одну папку для упрощения дальнейшей работы.

Рис. 1. Изображение

# **Создание макета**

Для создания собственного сайта необходимо разработать макет. Это нам поможет в дальнейшей работе, поможет визуализировать сайт. Вот что получилось у нас



Макет можно построить на онлайн-сервисе: <https://products.aspose.app/diagram/ru/mockup>

# **Создание html-файла**

Код представляет собой html-файл, который определяет структуру и содержание веб-страницы.

* Объявление документа html
* <!DOCTYPE html>
* <html lang="ru">

Рис.2. Объявление

Эти строки указывают, что это html -документ и что его язык - русский.

* Заголовок документа
* <head>
* <meta charset="UTF-8">
* <meta name="viewport" content="width=device-width, initial-scale=1.0">
* <title>Любимый город</title>
* <link type="image/x-icon" href="img/logo.ico" rel="shortcut icon">
* <link rel="stylesheet" href="style.css">
* <script type="text/javascript" src="app.js" defer=""></script>
* </head>

Рис. 3. Мета-теги

В этом блоке определены мета-теги, которые задают кодировку и масштабирование страницы. Также указан заголовок страницы, иконка для вкладки браузера, подключен файл стилей (style.css) и файл JavaScript (app.js).

* Тело документа
* <body>

Рис.4.Тело документа

Начинается основное содержание страницы

* Шапка сайта
* <header>
* <div class="logo">
* <img src="img/logo.jpg">
* </div>
* <h1 class="site-title">Экскурсии по Санкт-Петербургу</h1>
* <nav>
* <ul>
* <li><a href="#section1">О нас</a></li>
* <li><a href="#section2">Экскурсии</a></li>
* <li><a href="#section3">Заказать</a></li>
* </ul>
* </nav>
* </header>

Рис.5. Шапка сайта

Этот блок содержит шапку сайта с логотипом, названием сайта и навигацией.

Он представляет собой фрагмент html -разметки, который описывает верхнюю часть веб-страницы, обычно называемую "шапкой".

<header> - это html -тег, который обозначает начало секции шапки страницы. Шапка обычно содержит элементы, связанные с заголовком и навигацией сайта.

<div class="logo"> - элемент <div> с классом "logo" представляет собой контейнер для логотипа. Внутри этого контейнера находится изображение логотипа, которое загружается с помощью атрибута src из файла "logo.jpg" в подпапке "img".

<h1 class="site-title">Экскурсии по Санкт-Петербургу</h1> - элемент <h1> представляет заголовок страницы. У него также установлен класс "site-title", который может использоваться для применения стилей с помощью CSS. Заголовок страницы гласит "Экскурсии по Санкт-Петербургу".

Элемент <nav> представляет навигационное меню сайта. Внутри этого меню располагается неупорядоченный список.

Элемент <ul> представляет собой неупорядоченный список, который содержит пункты меню.

Каждый пункт меню представлен элементом <li>, который содержит гиперссылку (<a>). В данном коде есть три пункта меню:

* "О нас" - это ссылка, которая указывает на раздел страницы с идентификатором "section1".
* "Экскурсии" - ссылка на раздел с идентификатором "section2".
* "Заказать" - ссылка на раздел с идентификатором "section3".
* Слайдер
* <div class="slider-container">
* <div class="slider">
* <img src="img/1.jpg"  height="100px" onclick="openFullscreenImage(this)">
* <img src="img/2.jpg"  height="100px" onclick="openFullscreenImage(this)">
* <img src="img/3.jpg"  height="100px" onclick="openFullscreenImage(this)">
* <img src="img/4.jpg"  height="100px" onclick="openFullscreenImage(this)">
* </div>
* <button class="prev-button" aria-label="Посмотреть предыдущий слайд">&lt;</button>
* <button class="next-button" aria-label="Посмотреть следующий слайд">&gt</button>
* </div>

Рис.6. Слайдер

Этот блок представляет собой слайдер с изображениями, который может быть пролистан с помощью кнопок "Предыдущий" и "Следующий".

Блок представляет собой фрагмент html и JavaScript, который создает простой слайдер изображений на веб-странице.

<div class="slider-container"> - этот <div> элемент представляет собой контейнер для всего слайдера изображений.

<div class="slider"> - внутри контейнера "slider-container" находится вложенный <div> с классом "slider", который представляет контейнер для изображений слайдера.

Изображения слайдера: внутри контейнера "slider" находятся четыре изображения (<img>), каждое из которых имеет свой собственный путь к файлу (например, "img/1.jpg", "img/2.jpg" и так далее). Каждое изображение имеет установленную высоту в 100 пикселей (`height="100px"`) и привязано к функции openFullscreenImage(this), которая вызывается при клике на изображение.

onclick="openFullscreenImage(this)" - этот атрибут вызывает функцию `openFullscreenImage(this)` при клике на изображение. Параметр `this` передается функции, чтобы она могла определить, на какое изображение был совершен клик.

`<button class="prev-button" aria-label="Посмотреть предыдущий слайд">&lt;</button>` - эта кнопка "Предыдущий слайд". Она предназначена для переключения на предыдущее изображение в слайдере. Стрелка "&lt;" внутри кнопки является текстовым представлением стрелки влево.

`<button class="next-button" aria-label="Посмотреть следующий слайд">&gt;</button>` - эта кнопка "Следующий слайд". Она предназначена для переключения на следующее изображение в слайдере. Стрелка "&gt;" внутри кнопки является текстовым представлением стрелки вправо.

`openFullscreenImage(this)` - эта функция JavaScript, которая обрабатывает событие клика на изображении. Она реализована в другой части кода и отвечает за открытие изображения в полноэкранном режиме.

Все вместе, этот код создает простой слайдер изображений с кнопками для переключения между изображениями и функцией JavaScript для открытия изображения в полноэкранном режиме при клике на него.

* Секции сайта
* <section id="section1">
* <div class="text\_o\_nas">
* <h1>О нас:</h1>
* <p>Санкт-Петербург - это уникальный город, один из тех магических уголков мира, который способен покорить Ваши сердца и пробудить в вас жажду исследований.
* </p>
* <p>Мы, команда "название", предлагаем вам не просто посмотреть на этот удивительный город, но и ощутить его душу, погрузиться в его историю и культуру. Наши экскурсии - это не просто маршруты, а истории, которые оживают перед вашими глазами.
* </p>
* <p>Мы приглашаем вас в увлекательное путешествие по Санкт-Петербургу, где каждая улица, каждый мост, каждый музей - это новая страница великой истории. Мы поможем вам открыть для себя сокровища этого города, которые скрыты от глаз многих.
* </p>
* </div>
* <script src="https://static.elfsight.com/platform/platform.js" data-use-service-core defer></script>
* <div class="elfsight-app-1c5c3d38-d531-4d2e-bfb6-2fa411e8f53e"></div>
* </section>
* <section id="section2">
* <div class="column-container">
* <div class="column">
* <h2>Магия фонтанов: путешествие в Петергоф</h2>
* <img src="img/5.jpg" class="pic" onclick="openFullscreenImage(this)">
* <p>Добро пожаловать в мир Петергофа! Наши экскурсии предлагают вам уникальную возможность окунуться в великолепие этого исторического места, расположенного всего в нескольких километрах от Санкт-Петербурга. Мы приглашаем вас на захватывающее путешествие в прекрасный мир дворцов, садов и фонтанов Петергофа.</p>
* </div>
* <div class="column">
* <h2>Секретные места Васильевского острова</h2>
* <img src="img/6.jpg" class="pic" onclick="openFullscreenImage(this)">
* <p>Уникальная экскурсия, которая откроет перед вами тайны и неизведанные уголки Васильевского острова в Санкт-Петербурге! Наша экскурсия приглашают Вас отправиться в увлекательное путешествие по этому историческому району, где каждый уголок хранит свои секреты.</p>
* </div>
* <div class="column">
* <h2>Тайны Петропавловской крепости</h2>
* <img src="img/7.jpg" class="pic" onclick="openFullscreenImage(this)">
* <p>Добро пожаловать на уникальную историческую одиссею - путешествие в самое сердце Петропавловской крепости. Наша экскурсия приглашает вас отправиться в невероятное путешествие во времени и открыть перед собой богатство и тайны этого исторического символа Санкт-Петербурга.</p>
* </div>
* </div>
* <div class="column-container">
* <div class="column">
* <h2>Санкт-Петербург: Очарование Белых Ночей</h2>
* <img src="img/8.jpg" class="pic" onclick="openFullscreenImage(this)">
* <p>Обзорная экскурсия в Санкт-Петербург, где каждый камень, улица и здание хранят свои удивительные истории! Наша экскурсия - это ворота в мир этого невероятного города, полного культурного наследия и архитектурных шедевров.
* </p>
* </div>
* <div class="column">
* <h2>За Завесой Дворов и Парадных</h2>
* <img src="img/9.jpg" class="pic" onclick="openFullscreenImage(this)">
* <p>Истинная красота Петербурга скрывается за фасадами роскошных особняков, мы покажем Вам самые необычные места в центре города. Пройдясь по изящно декорированным парадным доходных домов и удивительным дворам-колодцам, Вы откроете для себя новый город Петра. </p>
* </div>
* <div class="column">
* <h2>Санкт-Петербурга: Экскурсия по Крышам</h2>
* <img src="img/10.jpg" class="pic" onclick="openFullscreenImage(this)">
* <p>Добро пожаловать на волнительное путешествие, которое откроет перед вами совершенно новый взгляд на Санкт-Петербург! Наша экскурсия по крышам города приглашает вас покорить его высоты и насладиться захватывающими видами, доступными лишь с высотных точек.</p>
* </div>
* </div>
* <div id="fullscreen-container" class="hidden">
* <span class="close-button" onclick="closeFullscreenImage()">&times;</span>
* <img src="" alt="Полноэкранное изображение" id="fullscreen-image">
* </div>
* </section>
* <section id="section3">
* <form class="obratnuj-zvonok" autocomplete="off" /\*action='email.php'\*/ method='post'>
* <div class="form-zvonok">
* <div>
* <label>Имя <span>\*</span></label>
* <input type='text' name='name' required></div>
* <div>
* <label>Номер телефона (с кодом) <span>\*</span></label>
* <input type='text' name='phone' required></div>
* <div>
* <label>Сообщение</label>
* <input type='text' name='message'>
* </div>
* <div>
* <label>Введите текст с картинки<span>\*</span></label>
* <input type='text' name='capcha' required>
* </div>
* <img src="img/capcha.jpg" class="pic">
* <input class="bot-send-mail" type='submit' value='Послать заявку'>
* </div>
* </form>
* </section>
* <button id="scrollToTopButton">Вверх</button>
* </body>

Рис.7. - секции

Эти секции включают в себя разное содержание, а именно текст, изображения и форму обратной связи.

Section1 представляет собой фрагмент html-разметки для раздела веб-страницы с идентификатором "section1"

<section id="section1"> - этот html -тег `<section>` создает раздел на веб-странице с уникальным идентификатором "section1". Этот идентификатор может использоваться для ссылок или стилей CSS.

<div class="text\_o\_nas"> - внутри раздела есть блок с классом "text\_o\_nas", который представляет собой контейнер для текстовой информации.

`<h1>О нас:</h1>` - этот заголовок первого уровня `<h1>`, который обозначает раздел текста "О нас:".

<p> - элементы `<p>` представляют собой абзацы текста, содержащие информацию о Санкт-Петербурге и предлагаемых экскурсиях. Этот текст описывает уникальность города и приглашает посетителей на увлекательное путешествие, подчеркивая, что экскурсии представляют собой не просто маршруты, а живые истории.

<script src="https://static.elfsight.com/platform/platform.js" data-use-service-core defer></script>` - этот скрипт подключает внешний JavaScript-код с https://static.elfsight.com/platform/platform.js (виджет погоды). Атрибуты `data-use-service-core` и `defer` связаны с использованием плагина для веб-страницы.

<div class="elfsight-app-1c5c3d38-d531-4d2e-bfb6-2fa411e8f53e"> - этот блок представляет собой внедренный элемент, связанный с внешним JavaScript-кодом. У него установлен класс "elfsight-app-1c5c3d38-d531-4d2e-bfb6-2fa411e8f53e", и он представляет функциональный элемент, предоставляемый сервисом "elfsight.com".

Section1 и Section3 написаны по аналогии.

Кода создают разделы веб-страницы, которые содержат текстовую информацию "О нас", "Эксурсии" "Заказать" и дополнительную функциональность через внешний JavaScript-скрипт и виджет от "elfsight.com".

* Скрипт
* </body>
* <footer style="width: 100%;">
* <div class="social-icons">
* <a href="https://web.telegram.org/" class="social-icon"><img src="img/tg.webp" alt="Tg"></a>
* <a href="https://m.vk.com/" class="social-icon"><img src="img/vk.png" alt="Vk"></a>
* <a href="https://www.instagram.com/" class="social-icon"><img src="img/ig.png" alt="Ig"></a>
* </div>
* </footer>

Рис.8. Скрипт

Этот скрипт, предназначен для встраивания сторонний информации на сайте.

Этот код представляет собой фрагмент html -разметки для нижней части веб-страницы (footer). Давайте разберем, что происходит в данном коде:

<footer style="width: 100%;"> - этот html -тег `<footer>` обозначает начало секции нижней части веб-страницы, обычно используемой для размещения информации, связанной с авторством, контактами или ссылками на социальные сети. Атрибут `style="width: 100%;"` устанавливает ширину нижней части страницы на 100%, что означает, что она будет занимать всю доступную ширину.

<div class="social-icons"> - Внутри секции `<footer>` находится контейнер с классом "social-icons", который используется для размещения и стилизации значков (иконок) социальных сетей.

Внутри контейнера "social-icons" находятся три элемента <a>, представляющих собой ссылки на различные социальные сети, такие как Telegram, ВКонтакте (VK) и Instagram. Каждая ссылка имеет атрибут `href`, указывающий на URL-адрес социальной сети, и класс "social-icon", который может использоваться для стилизации.

<img> - в каждой из ссылок <a> есть элемент <img>, представляющий собой изображение (иконку) социальной сети. Он имеет атрибут src, который указывает на путь к изображению (например, "img/tg.webp", "img/vk.png", "img/ig.png"), и атрибут `alt`, который предоставляет альтернативный текст для изображения (когда изображение недоступно, этот текст будет отображаться).

Этот код создает нижнюю часть веб-страницы (footer), содержащую иконки социальных сетей с соответствующими ссылками. Посетители могут кликнуть на эти иконки, чтобы перейти на соответствующие страницы в социальных сетях.

* Конец файла
* </html>

Рис.9. Конец файла

</html> - так завершается html-файл.

Этот код представляет собой структуру и содержание веб-страницы, а также подключает стили и скрипты для управления визуальными и интерактивными элементами страницы. В коде также реализуется форма обратной связи, кнопка "Вверх" для быстрой навигации и иконки социальных сетей в нижней части страницы.

# **Создание js-файла**

JavaScript-код предназначен для управления слайдером изображений, реализации функции открытия изображения в полноэкранном режиме, кнопки "Наверх" для плавной прокрутки страницы вверх и показа/скрытия этой кнопки в зависимости от позиции прокрутки страницы.

* Настройка переменных и выбор DOM-элементов
* const slider = document.querySelector('.slider');
* const prevButton = document.querySelector('.prev-button');
* const nextButton = document.querySelector('.next-button');
* const slides = Array.from(slider.querySelectorAll('img'));
* const slideCount = slides.length;
* let slideIndex = 0;

Рис.10. Настройка

Этот код представляет собой JavaScript-скрипт и выполняет следующие действия:

`const slider = document.querySelector('.slider');` - эта строка находит элемент на веб-странице с классом "slider" и сохраняет его в переменной `slider`. Элемент с этим классом представляет собой слайдер изображений.

`const prevButton = document.querySelector('.prev-button')` - здесь находится элемент с классом "prev-button" и сохраняется в переменной `prevButton`. Этот элемент представляет кнопку "Предыдущий слайд" в слайдере.

`const nextButton = document.querySelector('.next-button');` - эта строка находит элемент с классом "next-button" и сохраняет его в переменной `nextButton`. Этот элемент, представляет кнопку "Следующий слайд" в слайдере.

`const slides = Array.from(slider.querySelectorAll('img')); - этот код находит все изображения (элементы <img>) внутри элемента с классом "slider" и создает из них массив `slides`. Предполагается, что каждое изображение представляет один слайд в слайдере.

`const slideCount = slides.length; - здесь определяется общее количество слайдов, и это значение сохраняется в переменной `slideCount`.

let slideIndex = 0 - создается переменная `slideIndex`, которая инициализируется значением 0. Эта переменная будет использоваться для отслеживания текущего индекса активного слайда в слайдере.

Эта часть кода настраивает переменные и элементы, необходимые для управления слайдером изображений, а именно для перемещения между слайдами и отслеживания текущего активного слайда.

* Обработчики событий для кнопок
* // Устанавливаем обработчики событий для кнопок
* prevButton.addEventListener('click', showPreviousSlide);
* nextButton.addEventListener('click', showNextSlide);

Рис.11. Обработка событий

Эти строки кода добавляют обработчики событий для кнопок "Предыдущий слайд" (prevButton) и "Следующий слайд" (nextButton) в слайдере. Когда пользователь кликает на эти кнопки, соответствующие функции будут вызваны.

`prevButton.addEventListener('click', showPreviousSlide)` - эта строка добавляет обработчик события "click" (щелчок мышью) для элемента, представляющего кнопку "Предыдущий слайд" (prevButton). Когда происходит клик на этой кнопке, функция `showPreviousSlide` будет вызвана.

`nextButton.addEventListener('click', showNextSlide);` - аналогично этой строке добавляется обработчик события "click" для элемента, представляющего кнопку "Следующий слайд" (nextButton). Когда пользователь кликает на этой кнопке, функция `showNextSlide` будет вызвана.

* Функции для показа предыдущего и следующего слайда
* // Функция для показа предыдущего слайда
* function showPreviousSlide() {
* slideIndex = (slideIndex - 1 + slideCount) % slideCount;
* updateSlider();
* }
* // Функция для показа следующего слайда
* function showNextSlide() {
* slideIndex = (slideIndex + 1) % slideCount;
* updateSlider();
* }

Рис.12. Функции для показа слайдов

В этих двух функциях выполняется управление слайдером, чтобы переключать отображаемые слайды между предыдущим и следующим.

showPreviousSlide / (Показать предыдущий слайд):

`slideIndex` уменьшается на 1: `slideIndex = (slideIndex - 1 + slideCount) % slideCount;`

Эта строка кода обеспечивает корректное переключение слайдов. Если `slideIndex` становится меньше 0, то она обернет его обратно к последнему слайду (это реализовано с использованием оператора `% slideCount`). Таким образом, если вы находитесь на первом слайде и вызываете `showPreviousSlide()`, вы перейдете к последнему слайду.

showNextSlide() / (Показать следующий слайд):

`slideIndex` увеличивается на 1: `slideIndex = (slideIndex + 1) % slideCount;`

Эта строка кода обеспечивает переключение на следующий слайд. Если `slideIndex` достигает `slideCount - 1` (последний слайд), вызов `showNextSlide()` переключит вас на первый слайд, так как она обернет `slideIndex` обратно к 0.

Обе функции после обновления `slideIndex` вызывают функцию `updateSlider()`, которая обновляет отображение слайдов на странице, показывая только активный слайд, скрывая остальные. Это обеспечивает работу слайдера при нажатии на кнопки "Предыдущий" и "Следующий", а также при начальной инициализации страницы.

* Функция для обновления отображения слайдера
* // Функция для обновления отображения слайдера
* function updateSlider() {
* slides.forEach((slide, index) => {
* if (index === slideIndex) {
* slide.style.display = 'block';
* } else {
* slide.style.display = 'none';
* }
* });
* }

Рис.12. Отображение слайдера

Функция, `updateSlider()`, отвечает за обновление отображения слайдера на веб-странице. Внутри функции выполняется перебор всех слайдов и устанавливается стиль отображения для каждого из них в зависимости от `slideIndex`.

`slides` - это массив, который содержит все изображения слайдера.

`slideIndex` - эта переменная, которая указывает на индекс текущего активного слайда.

Внутри функции `updateSlider()`, используя метод `forEach`, мы перебираем все слайды в массиве `slides`. Для каждого слайда: если индекс (`index`) текущего слайда соответствует `slideIndex`, то устанавливается стиль `display` в значение `'block'`. Это делает текущий слайд видимым, так как значение `'block'` означает, что элемент отображается. В противном случае (если индекс не соответствует `slideIndex`), стиль `display` устанавливается в значение `'none'`. Это скрывает слайд, так как элементы со значением `'none'` не отображаются на странице.

Эта функция обеспечивает корректное отображение только активного слайда, скрывая остальные, когда пользователь переключает слайды, вызывая `showPreviousSlide()` и `showNextSlide()`. Это создает эффект слайд-шоу, показывая только один слайд в определенный момент времени.

* Инициализация слайдера

`updateSlider();` - эта строка кода вызывает updateSlider() для начальной настройки слайдера при загрузке страницы.

* Функции для открытия и закрытия изображения в полноэкранном режиме
* //открытие на полный экран
* function openFullscreenImage(element) {
* const fullscreenContainer = document.getElementById('fullscreen-container');
* const fullscreenImage = document.getElementById('fullscreen-image');
* fullscreenImage.src = element.src;
* fullscreenContainer.style.display = 'block';
* }
* function closeFullscreenImage() {
* const fullscreenContainer = document.getElementById('fullscreen-container');
* fullscreenContainer.style.display = 'none';
* }

Рис.13. Открытие и закрытие изображения

Эти две функции `openFullscreenImage` и `closeFullscreenImage` предназначены для открытия изображения в полноэкранном режиме (при клике на миниатюру из слайдера) и закрытия полноэкранного изображения.

openFullscreenImage(element) / Открыть изображение в полноэкранном режиме:

`element` - это параметр функции, который представляет собой элемент (изображение), на которое пользователь кликнул.

Сначала получаем ссылки на два DOM-элемента: `fullscreenContainer` и `fullscreenImage`. `fullscreenContainer` - это контейнер, который содержит изображение в полноэкранном режиме, а `fullscreenImage` - это изображение в полноэкранном режиме.

Затем устанавливаем `src` (путь к изображению) для `fullscreenImage` равным `src` элемента, на который пользователь кликнул. Это позволяет отобразить выбранное изображение в полноэкранном режиме.

Наконец, устанавливаем стиль `display` для `fullscreenContainer` в значение `'block'`, что делает контейнер видимым и открывает изображение в полноэкранном режиме.

closeFullscreenImage() / Закрыть полноэкранный режим изображения:

Получаем ссылку на DOM-элемент `fullscreenContainer`, который содержит изображение в полноэкранном режиме.

Устанавливаем стиль `display` для `fullscreenContainer` в значение `'none'`, что скрывает контейнер и закрывает полноэкранный режим изображения.

Эти функции добавляют интерактивность на сайте, позволяя пользователям открывать изображения в полноэкранном режиме и закрывать его по запросу.

* Плавная прокрутка страницы и отображение кнопки «Наверх»
* // Когда пользователь прокручивает страницу вниз 20px от верха, показать кнопку
* window.onscroll = function() {
* scrollFunction();
* };
* function scrollFunction() {
* if (document.body.scrollTop > 20 || document.documentElement.scrollTop > 20) {
* document.getElementById('scrollToTopButton').style.display = 'block';
* } else {
* document.getElementById('scrollToTopButton').style.display = 'none';
* }
* }
* // Плавный скроллинг при клике на кнопку "Наверх"
* document.getElementById('scrollToTopButton').addEventListener('click', function() {
* scrollToTop();
* });
* function scrollToTop() {
* const scrollStep = -window.scrollY / 15;
* const scrollInterval = setInterval(function() {
* if (window.scrollY !== 0) {
* window.scrollBy(0, scrollStep);
* } else {
* clearInterval(scrollInterval);
* }
* }, 15);
* }

Рис.14.Плавная прокрутка

Этот код реализует функциональность, которая позволяет пользователю прокручивать страницу вверх при помощи кнопки "Наверх".

`window.onscroll = function() { scrollFunction(); };` - эта строка добавляет обработчик события `onscroll` к объекту `window`, который срабатывает при прокрутке страницы. Когда происходит прокрутка, вызывается функция `scrollFunction()`.

`function scrollFunction() { ... }` - эта функция определена для обработки события прокрутки. Она проверяет, находится ли пользователь на странице выше 20 пикселей от верха (как в `if (document.body.scrollTop > 20 || document.documentElement.scrollTop > 20)`) и, в зависимости от результата, показывает или скрывает кнопку "Наверх" (с идентификатором "scrollToTopButton").

`document.getElementById('scrollToTopButton').addEventListener('click', function() { scrollToTop(); });` - здесь устанавливается обработчик события "click" для кнопки "Наверх". Когда пользователь кликает на эту кнопку, вызывается функция `scrollToTop()`.

`function scrollToTop() { ... }` - эта функция выполняет плавную прокрутку страницы вверх при клике на кнопку "Наверх". Она использует `setInterval` для постепенного изменения позиции прокрутки до тех пор, пока страница не достигнет верхней части. В конечном итоге, страница будет плавно прокручена вверх.

Таким образом, весь этот код обеспечивает пользователю возможность плавной прокрутки страницы вверх при помощи кнопки "Наверх", которая появляется после того, как пользователь прокрутит страницу на 20 пикселей или более вниз.

Этот код обеспечивает интерактивность веб-страницы, позволяя пользователям переключаться между слайдами, просматривать изображения в полноэкранном режиме и удобно перемещаться по странице.

# **Php-файлы**

PHP-код представляет собой обработчик формы обратной связи на веб-сайте.

* Проверяет, был ли запрос на сервер сделан методом POST:
* <?php
* if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

Рис.15. Проверка запроса

Это условие гарантирует, что код будет выполняться только при отправке данных формы методом POST.

* Извлечение данных из формы
* $name = $\_POST["name"];
* $phone = $\_POST["phone"];
* $message = $\_POST["message"];

Рис.16. Извлечение данных из формы

Name, phone, и message извлекаются из массива $\_POST, что соответствует значениям полей формы, содержащимся в отправленной форме.

* Определение адрес, на который отправлено письмо
* // Адрес, на который нужно отправить письмо
* $to = "тут ваша почта";

Рис.17. Работа с адресом

Здесь указывается адрес электронной почты, на который будет отправлено письмо

* Задает тему письма
* // Тема письма
* $subject = "Обратная связь от $name";

Рис.18.Тема письма

Тема письма формируется с использованием имени отправителя, которое было введено в форму.

* Создает текст сообщения
* // Сообщение
* $email\_message = "Имя: $name\n";
* $email\_message .= "Телефон: $phone\n";
* $email\_message .= "Сообщение:\n$message";

Рис.19. Создание текста сообщения

Здесь формируется текст сообщения, который будет включен в отправляемое письмо. Этот текст включает в себя имя отправителя, номер телефона и сообщение, которое было введено в форму.

* Отправляет письмо
* // Отправка письма
* mail($to, $subject, $email\_message);

Рис.20. Отправка письма

Эта строка кода использует функцию mail() для отправки письма на указанный адрес. В функции передаются параметры: адрес получателя ($to), тема письма ($subject) и текст сообщения ($email\_message).

* Опционально: перенаправление пользователя на другую страницу "Спасибо":

    // Опционально: перенаправление на страницу "Спасибо"

    //header("Location: thank\_you.html");

}

?>

Рис.21.Опциональное действие

Эта строка кода закомментирована (символами // в начале строки), но если бы она была раскомментирована, то после отправки формы пользователь был бы перенаправлен на страницу "Спасибо". Это может быть полезно для отображения подтверждения успешной отправки формы.

Этот код выполняет обработку формы обратной связи, собирает введенные данные и отправляет их на указанный адрес электронной почты.

# **CSS-файл**

/\* Стили для шапки \*/

header {

  background-color: #1e1e1e;

  color: #fff;

  padding: 20px 0;

  display: flex;

  justify-content: space-between;

  align-items: center;

}

.logo img {

  width: 90px;

  height: 90px;

  margin-right: 10px;

}

.logo h1 {

  font-size: 24px;

  margin: 0;

}

hr {

  border: none;

  border-top: 1px solid #fff;

  margin: 0 20px; /\* Расстояние между линией и меню \*/

  height: 1px;

  flex-grow: 1; /\* Равномерное распределение пространства между логотипом и меню \*/

}

nav ul {

  list-style-type: none;

  margin: 0;

  padding: 0;

  display: flex;

}

nav li {

  margin-right: 20px;

}

nav a {

  text-decoration: none;

  color: #fff;

  font-weight: bold;

  transition: color 0.3s;

}

nav a:hover {

  color: #ff6600;

}

body {

  /\* Основной цвет фона \*/

  background-color: #1e1e1e;

  width: 85%;

  margin-left:auto;

  margin-right:auto;

}

/\* Стили для названия сайта \*/

.site-title {

  text-align: right;

  opacity: 0; /\* Начальная прозрачность: 0 \*/

  animation: fadeInUp 1.5s ease-out forwards; /\* Применение анимации \*/

}

/\* Анимация \*/

@keyframes fadeInUp {

  from {

      opacity: 0;

      transform: translateY(20px); /\* Двигаем название вниз \*/

  }

  to {

      opacity: 1;

      transform: translateY(0); /\* Возвращаем название в исходное положение \*/

  }

}

/\* Стили слайдера \*/

.page-title {

    text-align: center;

  }

.slider-container {

  position: relative;

  width: 80%;

  height: 400px;

  margin: 0 auto;

  overflow: hidden;

}

.slider {

  display: flex;

  transition: transform 0.5s ease-in-out;

}

.slider img {

  width: 100%;

  height: 100%;

  object-fit: cover;

}

.prev-button, .next-button {

  position: absolute;

  top: 50%;

  transform: translateY(-50%);

  width: 50px;

  height: 50px;

  background-color: transparent;

  border: none;

  font-size: 24px;

  color: #000;

}

.prev-button {

  left: 10px;

}

.next-button {

  right: 10px;

}

/\* Стили текста о нас \*/

.text\_o\_nas{

  color: #fff;

  width:80%;

  font-size:18px;

  text-align:center;

  font-family:Verdana, Geneva, sans-serif;

  margin-left:auto;

  margin-right:auto;

}

/\* Стили галереи \*/

.column-container {

  display: flex; /\* Размещение элементов в строку \*/

  justify-content: space-between; /\* Распределение пространства между колонками \*/

  color: #fff;

}

.pic {

  width: 100%;

  height: 50%;

}

/\* Стили для каждой колонки \*/

.column {

  flex: 1; /\* Равномерное распределение ширины между колонками \*/

  margin: 10px; /\* Отступ между колонками \*/

  padding: 10px; /\* Внутренний отступ внутри каждой колонки \*/

  border: 1px solid #a7401b; /\* Граница для каждой колонки \*/

  border-radius: 5px;

  background: #333333;

}

/\* Стили обратной связи \*/

.obratnuj-zvonok{

  width: 100%;

  max-width: 350px;

  margin-left: auto;

  margin-right: auto;

}

.form-zvonok{

  width: 100%;

  display: flex;

  flex-direction: column;

  padding: 0 20px;

  box-sizing: border-box;

  color: #fff;

}

.form-zvonok div{

  padding: 15px 0;

}

.bot-send-mail{

  box-sizing: border-box;

  width: 100%;

}

.form-zvonok label,.form-zvonok input{

  display: block;

  width: 100%;

  box-sizing: border-box;

}

.form-zvonok label{

  margin-bottom: 5px;

  font-weight: bold;

}

.form-zvonok input{

  padding: 10px 15px;

  margin-top: 10px;

  color: #fff;

  background-color: #333333;

  border: 1px #a7401b solid;

}

.form-zvonok label span{

  color: red;

}

.form-zvonok .bot-send-mail{

  padding: 15px;

  margin-top: 10px;

  background: none;

  text-transform: uppercase;

  color: #333333;

  font-weight: bold;

  background-color: #a7401b;

  cursor: pointer;

  border: 3px #a7401b solid;

  border-radius: 5px;

}

.form-zvonok .bot-send-mail:hover{

  color: #a7401b;

  background-color: #333333;

}

/\* Стили контактов \*/

footer {

  background-color: #1e1e1e; /\* Цвет фона \*/

  color: #fff; /\* Цвет текста \*/

  padding: 10px 0;

  text-align: center;

  width:100%;

}

.social-icons {

  font-size: 12; /\* Убираем междустрочный интервал для иконок \*/

}

.social-icon {

  display: inline-block;

  margin-right: 10px; /\* Расстояние между иконками \*/

}

/\* Стили для иконок (размер и цвет) \*/

.social-icon img {

  width: 32px; /\* Ширина иконки \*/

  height: 32px; /\* Высота иконки \*/

  vertical-align: middle; /\* Центрирование иконки по вертикали \*/

}

/\* Стили для ссылок (при наведении) \*/

.social-icon a {

  color: #fff; /\* Цвет ссылки \*/

  text-decoration: none; /\* Убираем подчеркивание ссылки \*/

  transition: color 0.3s; /\* Плавное изменение цвета при наведении \*/

}

.social-icon a:hover {

  color: #ff6600; /\* Цвет ссылки при наведении \*/

}

/\* Стили для контейнера полноэкранного изображения \*/

#fullscreen-container {

  display: none;

  position: fixed;

  top: 0;

  left: 0;

  width: 100%;

  height: 100%;

  background-color: rgba(0, 0, 0, 0.9);

  z-index: 999;

  text-align: center;

}

#fullscreen-image {

  max-width: 90%;

  max-height: 90%;

  margin: auto;

  display: block;

  position: relative;

  top: 50%;

  transform: translateY(-50%);

}

/\* Стили для кнопки закрытия \*/

.close-button {

  position: absolute;

  top: 10px;

  right: 10px;

  font-size: 30px;

  color: #fff;

  cursor: pointer;

}

/\* Стили для кнопки "Наверх" \*/

#scrollToTopButton {

  display: none;

  position: fixed;

  bottom: 20px;

  right: 20px;

  z-index: 999;

  padding: 10px 20px;

  background-color: #333333; /\* Цвет фона кнопки \*/

  color: #fff; /\* Цвет текста кнопки \*/

  border: none;

  border-radius: 5px;

  cursor: pointer;

}

#scrollToTopButton:hover {

  color:#ff6600;

}

Рис.22. CSS-файл

В этом CSS-коде определены стили для элементов вашего веб-сайта. Вот описание ключевых частей кода:

1. Стили для шапки (header):

* Устанавливается цвет фона, цвет текста, отступы и выравнивание элементов в шапке.
* Логотип и название сайта имеют свои стили: размер, отступы и т. д.
* Горизонтальная линия (hr) разделяет логотип и меню.

2. Стили для навигационного меню (nav):

* Устанавливаются стили для списка (`ul`) и элементов списка (`li`).
* Ссылки (`a`) получают стили, такие как цвет, жирность шрифта и анимация при наведении (`hover`).

3. Стили для тела (body):

* Устанавливается основной цвет фона, ширина, и центрирование содержимого.

4.Стили для названия сайта (site-title):

* Элементу с классом `site-title` применяются стили для анимации, позволяющей элементу плавно появиться сверху при загрузке страницы.

5. Стили для слайдера (slider):

* Слайдеру и его изображениям (`img`) устанавливаются стили для размеров и перехода между слайдами.
* Кнопкам "Предыдущий" и "Следующий" (`prev-button` и `next-button`) устанавливаются стили, такие как размер, цвет фона и позиция.

6. Стили для текста "О нас" (text\_o\_nas):

* Определяются цвет, ширина, размер шрифта, отступы и другие стили для текста.

7. Стили для галереи (column-container):

* Стили для контейнера колонок и изображений в галерее.

8. Стили для каждой колонки (column):

* Устанавливаются стили для каждой отдельной колонки, такие как размер, отступы, границы и фон.

9. Стили для формы обратной связи (obratnuj-zvonok):

* Определяются стили для ширины, отступов и цвета элементов формы, таких как поля ввода, метки и кнопка отправки.

10. Стили для контактов (footer):

* Устанавливаются стили для подвала, включая цвет фона, цвет текста и выравнивание.

11. Стили для социальных иконок (social-icons):

* Здесь определяются стили для иконок социальных сетей, включая размер и расстояние между иконками.

12. Стили для полноэкранного изображения (fullscreen-container):

* Элементу, отображающему полноэкранное изображение, присваиваются стили, чтобы он занимал весь экран и имел полупрозрачный фон. Элемент можно закрыть с помощью кнопки "Закрыть" (`close-button`).

13.Стили для кнопки "Наверх" (scrollToTopButton):

* Кнопке "Наверх" присваиваются стили, включая цвет фона, цвет текста, размер и позиция. При наведении кнопка меняет цвет.

Эти стили позволяют определить внешний вид различных элементов на веб-странице, от шапки и навигации до текста, изображений, формы обратной связи и кнопки "Наверх". С помощью CSS можно кастомизировать дизайн сайта и делать его более привлекательным для пользователей.

# **Промежуточный итог**

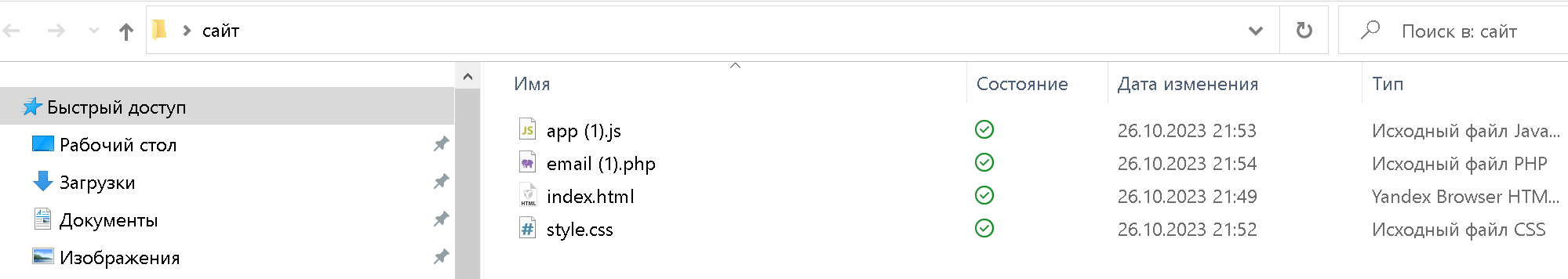
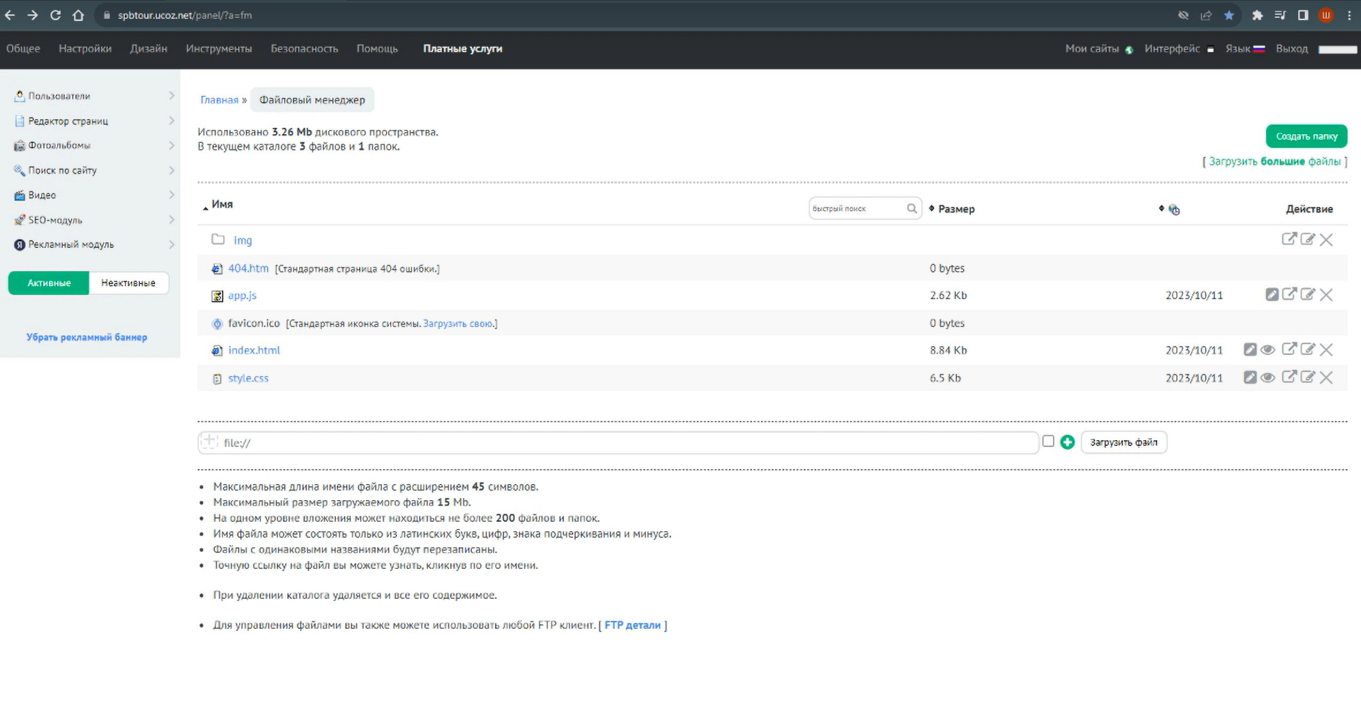
У нас получилось создать 4 файла, которые для удобства мы сложили в одну папку. Если все правильно сделать, у вас уже сейчас получится увидеть плод своей работы.

Рис.23. Папка с сайтом

# **Домен для сайта**

Для того, чтобы сайт правильно работал, ему нужен домен. Для этого мы использовали бесплатный веб-ресурс. После регистрации на сайте у нас появилась возможно присвоить собственному сайту домен.

Рис. 24. Создание домена

# **Результат**

Проделов все этапы, которые мы упоминали выше, мы создали собственный сайт.

<https://spbtour.ucoz.net/index.html>

# **Вывод**

Создание сайта - это увлекательный и многогранный процесс, который может служить различными целями: от личных портфолио и блогов до интернет-магазинов и корпоративных сайтов. Этот процесс предоставляет уникальные возможности для самовыражения, общения с аудиторией и развития бизнеса.

Также создание сайта - это непрерывный процесс, который может развиваться вместе с вами и вашими целями. Важно оставаться творческим и стремиться к улучшению своего сайта.

# **Для защиты работы требуется:**

* разработать сайт фотографа, подходящий под следующие требования: обязательно наличие слайдера, галереи, блока «О фотографе», шапка сайта с логотипом, подвал сайта с ссылками на социальные сети
* разработать сайт другого фотографа, подходящий под следующие требования: обязательно наличие слайдера, галереи, блока «О фотографе», шапка сайта с логотипом, подвал сайта с ссылками на социальные сети, добавить расписание фотосессий в виде таблицы, добавить анимацию.
* разработать сайт кондитера, подходящий под следующие требования: обязательно наличие слайдера, галереи, блока «О кондитере», шапка сайта с логотипом, подвал сайта с ссылками на социальные сети, добавить видео приготовления, добавить рецепты.
* разработать сайт грумера, подходящий под следующие требования: обязательно наличие слайдера, галереи, блока «О грумере», шапка сайта с логотипом, подвал сайта с ссылками на социальные сети, добавить таблицу с графиком работы. Обязательно много фотографий милых собачек и котиков.
* Напишите аналогичный сайт на свободную тему.