Государственное бюджетное общеобразовательное учреждение г. Москвы

«Инженерная школа № 1581»

Программа: Информационные технологии и системы обработки информации

Курс: Веб-дизайн и разработка

РЕФЕРАТ

на тему: «Основные принципы разработки веб-сайтов»

Выполнил: ученик 10-М класса
Карасев Е.В.

Проверил: Гришина А.А.

Москва

2024

Оглавление

1. Основные понятия разработки веб-сайтов

Веб-разработка – это процесс создания и поддержания работы веб-сайтов.

Веб-сайт – совокупность взаимосвязанных веб-страниц, объединенных общей темой и дизайном и размещенных под одним доменным именем в сети Интернет.

Веб-страница – это отдельный документ, который может содержать текст, изображения, видео и др. информацию, доступ к которому возможен с помощью веб-браузера,

Веб-браузер – прикладная программа для просмотра веб-страниц (веб-сайтов).

Все веб-сайты в совокупности составляют **сеть Интернет**, т.н. Всемирную Паутину (World-Wide-Web, сокращенно **WWW**), в которой вся информация объединена в единое целое – базу данных планетарного масштаба.

Веб-сайты могут быть **статическими** и **динамическими**. Статические веб-сайты состоят из фиксированных страниц, которые не изменяются без вмешательства разработчика. Динамические веб-сайты генерируют контент «на лету», основываясь на запросах пользователей и данных из баз данных.

Веб сайты построены по так называемой **клиент-серверной архитектуре**. **Клиент** – программа, которая запрашивает информацию с сервера. **Сервер** – программа, которая обрабатывает запросы клиента и предоставляет ему информацию. Веб-браузер (Chrome, Edge, Firefox, Yandex-браузер) по сути, является **клиентом**, а непосредственно веб-сайт, к которому мы обращаемся по сети через браузер – **сервером**.

HTTP (Hypertext Transfer Protocol) – сетевой протокол (набор правил), по которому происходит сетевое взаимодействие между узлами Всемирной паутины.

URL (Uniform Resource Locator) – адрес веб-сайта в сети Интернет в стандартном формате.

Виды веб-разработки

В веб-разработке есть разные направления, которые объединяют различные задачи при создании веб-сайтов или разработке веб-приложений.

Backend-разработка

Backend-разработка — это создание серверной части веб-сайта, которая обеспечивает доступ к данным, обработку запросов клиентов, и прочую механику работы сайта. Например, для интернет-магазина Backend-разработчик программирует внутреннюю механику процесса заказа товаров в интернет-магазине. Для этого он работает с базами данных, связывает их с веб-приложением, настраивает доступ к сайту, авторизацию, систему безопасности и резервное копирование информации.

Frontend-разработка

Frontend-разработка — создание интерфейса веб-сайта, т.е. среды взаимодействия с пользователем. Frontend-разработчик полностью отвечает за интерфейс: как будет выглядеть сайт в целом, как будет работать его логика, расставляет кнопки, оформляет визуальные компоненты, выстраивает логику переходов между разделами, верстальщик выполняет HTML-верстку сайта, т.е. переводит задумки дизайнера в код на HTML и CSS. От правильной верстки зависит скорость работы сайта или приложения, возможность попадать в топ выдачи поисковых систем, корректность отображения страниц на разных устройствах.

2. Обзор технологий разработки веб-сайтов

Все технологии разработки веб-сайтов, в зависимости от решаемых задач, условно можно разделить на несколько категорий.

2.1. Технологии frontend-разработки

Как было сказано ранее, под frontend понимается создание интерактивной среды взаимодействия с пользователем.

2.1.1. Программы для веб-дизайна

Конечный результат разработки (веб-сайт) должен быть приятным и удобным для пользователя с визуальной очки зрения. Кроме того, сайт должен отражать общепринятый стиль (корпоративный, тематический и пр.). Поэтому создание визуального дизайна сайта – важный этап разработки.

Adobe Photoshop(Adobe Illustrator)

Многофункциональные графические редакторы. **Photoshop** работает как с растровыми, так и с векторными изображениями. **Illustrator** – векторный редактор. Обе программы позволяют создавать изображения, редактировать их, работать со слоями и использовать другие графические возможности.

Sketch

Программа для дизайна сайтов macOS, работает с векторной графикой. Позволяет разрабатывать интерфейсы мобильных приложений и веб-сайтов, поддерживает возможность создания интерактивных прототипов сайтов и пр.

Figma

Онлайн-программа для создания дизайна, имеющая возможность совместной работы в режиме онлайн. Программа имеет офлайн-версию, доступную для Windows/macOS. Векторные объекты импортируются из Adobe Illustrator или Sketch.

Canva

Инструмент для создания любых элементов графического дизайна: от логотипов, кнопок, картинок для соцсетей до полноценных макетов для полиграфии. Программа дает возможности создавать изображения с помощью шаблонов с изменяемыми элементами. Имеет библиотеку шаблонов, стоковых фотографий, иллюстраций, шрифтов.

2.1.2. Языки программирования frontend

Языки программирования frontend – средства разработки пользовательского интерфейса.

HTML (HyperText Markup Language)

Язык гипертекстовой разметки текста**.** Он нужен для того, чтобы структурировать и оформлять контент на сайте: размещать на веб-странице любые возможные элементы - текст, картинки, таблицы и видео.

CSS (Cascading Style Sheets)

Язык описания внешнего вида документа. Он отвечает за то, как выглядят веб-страницы: цвет фона и декоративных элементов, размер и стиль шрифтов. CSS позволяет создавать адаптивные дизайны, для корректного отображения на различных устройствах и экранах.

JavaScript

JavaScript добавляет интерактивность на веб-страницы. С его помощью можно создавать динамические элементы, такие как слайдеры, формы для отправки информации, анимации, всплывающие окна. JavaScript также используется для обработки событий, таких как клики мыши и ввод данных пользователем.

2.1.3. Frontend фрэймворки и библиотеки

Фреймворки и библиотеки – готовые решения, которые упрощают веб-разработку и позволяют решать сложные задачи с меньшими усилиями. Фреймворк – предварительно написанный скрипт, который создает структуру и правила. Библиотека - набор готовых функций, классов и объектов для решения разных задач. Популярные фреймворки и библиотеки для фронтенд-разработки включают:

React

Библиотека для создания пользовательских интерфейсов. Позволяет создавать компоненты, которые можно повторно использовать в различных частях приложения.

Vue

Прогрессивный фреймворк для создания пользовательских интерфейсов. Отличается простотой и гибкостью, что делает его популярным среди разработчиков

2.2. Технологии backend-разработки

Под backend понимается создание серверной части веб-сайта, которая обеспечивает доступ к данным, обработку запросов клиентов, и прочую механику работы сайта.

2.2.1. Серверные языки программирования

Популярные серверные языки программирования включают:

JavaScript (Node.js)

Gозволяет использовать JavaScript для серверной разработки. Отличается высокой производительностью и поддержкой асинхронного ввода-вывода.

PHP

Язык программирования, специально созданный для веб-разработки. Широко используется для создания динамических веб-сайтов и приложений, работы с базами данных, авторизацией и пр.

Python

Язык программирования с простым синтаксисом, часто используемый для веб-разработки. Python популярен благодаря своей читаемости и большому количеству библиотек.

2.2.2. Backend фрэймворки и библиотеки

Фреймворки упрощают разработку серверной логики и управления базами данных. Популярные фреймворки для бэкенд-разработки включают:

Express (для Node.js)

Минималистичный фреймворк для создания веб-приложений. Позволяет быстро создавать серверные приложения с минимальными усилиями.

Django (для Python)

Высокоуровневый фреймворк, предоставляет множество встроенных инструментов и функций, что упрощает разработку серверной части веб-сайта.

Laravel (для PHP)

Фреймворк отличается удобством использования и богатым набором функций.

2.2.3. Базы данных

Базы данных используются для хранения и управления всеми данными, которые использует веб-сайт. Популярные базы данных включают:

MySQL

Самая распространенная база данных для веб-сайтов. Это реляционная база данных с открытым исходным кодом. Широко используется для создания веб-приложений благодаря своей надежности и производительности.

PostgreSQL

Мощная реляционная база данных с поддержкой расширенных функций. Отличается высокой производительностью и поддержкой сложных запросов.

MongoDB

Современная объектная база данных, которая ориентируется на документы, а не строки. Имеет систему контроля доступа и поддержки безопасности данных, позволяет назначать права доступа разным категориям пользователей.

2.3. CMS-системы и конструкторы сайтов

CMS (Content Management Systems), или **«движок»** – платформа для создания и управления содержимым сайта. CMS позволяет размещать, обновлять, форматировать, вносить правки в контент, отслеживать и устранять неполадки, взаимодействовать с пользователями. В CMS необходимые для функционирования веб-сайта элементы (например, авторизация пользователей, настройка прав доступа) уже реализованы. CMS оснащена панелью управления, которая содержит настройки управления контентом.

CMS можно назвать комбинированным frontend/backend инструментом разработки. Однако, реализовывать в CMS сложные функции без участия программиста нельзя. Если CMS не предусматривает функционала, который необходим пользователю, разработчику придется придумать, как его встроить. Это проще, чем начинать с нуля, но требует от программиста знания языка, выбранной CMS и ее особенностей.

Можно назвать следующие популярные CMS-системы:

WordPress

Самая популярная система управления контентом с открытым исходным кодом. Позволяет сделать сайт, настроить его, редактировать, взаимодействовать с пользователями. Подходит как для личных блогов, так и для серьезных новостных изданий.

1C-bitrix

Наверное, самая популярная в России профессиональная CMS для создания сайта и его дальнейшего администрирования. Позволяет создать неограниченное количество веб-страниц, централизованно хранить и управлять ими. Ее можно применять для создания

корпоративных, информационных порталов, форумов, интернет-магазинов. Битрикс интегрирован с системой «1С: Предприятие 8», что позволяет автоматизировать торговлю.

Opencart

Система ориентирована на интернет-магазины. Здесь можно добавлять страницы товара, категории, продукты с различными вариантами доставки и оплаты. Opencart не ограничивает пользователей в количестве страниц, позволяет редактировать дизайн.

Конструктор сайтов — это инструмент, который позволяет создавать веб-сайты без необходимости написания кода. Он предоставляет пользователям визуальный интерфейс для добавления и настройки различных элементов сайта, таких как текст, изображения, видео и формы. Конструкторы предлагают готовые шаблоны, визуальные редакторы и другие инструменты для создания сайта бквально за несколько шагов. Они особенно полезны малому бизнесу, фрилансерам и блогерам, которым нужно создать что-то простое без лишних затрат.

Tilda

Популярный блочный конструктор сайтов, не требующий навыков программирования. Позволяет создавать сайты, интернет-магазины, посадочные страницы, блоги

Craftum

Один из самых сайтбилдеров, даже новичок освоится в редакторе за час, без привлечения программистов и дизайнеров. При этом платформа подойдет и продвинутым пользователям. В основном сервис специализируется на создании лендингов, визиток, портфолио, блогов, промостраниц, а также небольших интернет-магазинов.

2.4. Вспомогательные технологии для веб-разработки

Кроме перечисленных основных технологий разработки веб-сайтов, существует ряд вспомогательных инструментов, повышающих эффективность разработки.

2.4.1. Интегрированные среды разработки (IDE)

IDE – это набор инструментов для разработки: текстовый редактор, компилятор или интерпретатор, средства автоматизации сборки, отладчик.

Visual Studio - Это линейка продуктов Microsoft, которая включает интегрированную среду разработки программного обеспечения. Она предоставляет возможность внедрять, тестировать, компилировать все веб-проекты. Это JS, HTML, PHP редакторы исходного кода с возможностью рефакторинга между файлами.

PyCharm - Это интегрированная среда разработки для Python, в том числе для многоязычных веб-приложений с фреймворками. Поддерживает фреймворки Django, Flask, Google App Engine, Pyramid, а также языки JavaScript, CoffeeScript, TypeScript, CSS, Cython и другие. Позволяет проводить мощный рефакторинг кода с широкими возможностями по выполнению быстрых глобальных изменений в проекте.

2.4.2. Системы контроля версий (VCS)

VCS нужны для централизованного хранения, просмотра кода, фиксации правок, возврата к любой версии. Они отслеживают и фиксируют в специальную базу данных все

изменения, которые вносятся в исходный код. Если в ходе разработки обнаружится ошибка, с помощью системы контроля ее можно будет отследить: более позднюю версию сравнивают с ранней, находят изменения, возвращают файлы к состоянию, в котором они были до возникновения ошибки.

Git - Это самая быстрая система с компактным хранилищем ревизий. Изменения фиксируются в коде, а каждая сохраненная версия сама является репозиторием. Все разработчики могут хранить и просматривать историю изменений в полном объеме. Можно возвращаться к любой предыдущей версии кода, работать над проектом параллельно, делать бэкап.

Mercurial - Распределенная система контроля версий, написаная на Python, поэтому ей чаще пользуются именно Python-программисты.

2.4.3. Локальные и веб-серверы

С помощью **локального сервера** можно запускать веб-сайт на ПК без хостинга, чтобы разрабатывать и тестировать его. Браузеры могут открывать веб-страницы, написанные на HTML или CSS, но не могут открывать динамические сайты. Локальный сервер переводит сайт на HTML-код перед отправкой в браузер, имитируя работу реального веб-сервера.

OpenServer - Портативный локальный сервер. Имеет продуманный интерфейс, удобное управление с добавленными сайтами, легко устанавливается и настраивается. Позволяет переключаться между разными версиями HTTP, MySQL, PHP из интерфейса программы.

Под **веб-сервером** понимается ПО, с помощью которого контролируется доступ пользователей к размещенным на сервере файлам. Такое ПО называется HTTP-сервером. Он работает с URL-адресами и HTTP-протоколами. Веб-серверы нужны для обработки HTTP-запросов и ответов на них в виде HTML страниц.

Nginx - Открытый веб-сервер. Имеет хорошую производительность, справляется с высокими нагрузками, работает с любым ПО.

Apache - Один из самых популярных веб-серверов. Состоит из ядра и модулей конфигурации ядра. Поэтому ядро сервера отличается гибкой настройкой, что делает Apache надежным.

2.4.4. Инструменты для совместной работы

Средства совместной работы актуальны для команд, работающих удаленно, именно им необходимо поддерживать связь друг с другом, организовывать рабочий процесс.

Slack - Адаптирован для рабочего общения. Поддерживает групповые чаты, личные сообщения, аудио, видеозвонки.

Jira - Разработан специально под нужды программистов. Помогает разбивать задачи на маленькие этапы, по завершении которых команда выдает готовые части продукта.

2.4.5. Инструменты для тестирования сайта

Google Lighthouse - Дает возможность отследить производительность, доступность, поисковую оптимизацию веб-страниц.

Responsively - Браузер для разработчиков, позволяет быстро проверить, как отображается веб-сайт на разных устройствах корректность отображения веб-страниц в разных браузерах.

3. Основы HTML и CSS

3.1. Основы HTML

HTML (HyperText Markup Language) — это язык гипертекстовой разметки текста. Он нужен, чтобы размещать на веб-странице элементы: текст, картинки, таблицы и видео.

Когда мы заходим на сайт, браузер подгружает HTML-файл с информацией о структуре и контенте веб-страницы. HTML указывает, где располагаются элементы, какой у них будет базовый дизайн, откуда брать стили для элементов и скрипты.

Для создания HTML_файла достаточно любого текстового редактора, файл сохраняется с расширением .html. Но часто удобно использовать специальные редакторы, они подсвечивают код разным цветом, работать становится удобней, особенно новичку.

Из чего состоит HTML-код

HTML состоит из **тегов** — команд, которые указывают браузеру, как отображать помещённый в них текст. Это и есть элементы веб-страницы. У каждого тега есть имя, которое заключается в угловые скобки – «<» и «>».

Теги бывают **парные** и **непарные**. Парные состоят из двух тегов — открывающего и закрывающего, а непарные — из одного.

У каждого тега есть **атрибуты**. С их помощью можно передавать элементам веб-страницы дополнительные данные: размеры, id элемента, ссылки на изображения и т.д.

Базовая структура HTML-документа

Содержит основные теги, определяющие структуру документа:

<!DOCTYPE html> - объявление типа документа, указывающее браузеру, что это HTML5-документ. Это важно для совместимости с современными стандартами веб-разработки.

<html> - корневой элемент, содержащий весь контент страницы. Все остальные элементы HTML-документа должны быть вложены в этот тег.

<head> - элемент, содержащий метаданные о документе, такие как заголовок страницы и ссылки на стили. Здесь также указывают кодировку, подключают внешние стили и скрипты.

<title> - тег, определяющий заголовок страницы, который отображается на вкладке браузера. Этот заголовок используется поисковыми системами для индексации страницы.

<body> - элемент, содержащий видимый контент страницы. Все, что мы видим на веб-странице, находится внутри этого тега.

Теги форматирования HTML-документа

Когда мы форматируем документ в каком-то текстовом редакторе, вся техническая сторона этого процесса происходит незаметно. Мы набираем текст, нажимаем Enter для перехода на новую строку, для форматирования выделяем нужный фрагмент и выбираем для него какое-то действие: сделать заголовком, выделить жирным или курсивом и пр. Чтобы сделать то же самое в веб-документе, используют соответствующие теги:

**<p>,
** - для абзаца текста и переноса на другую строку

**<h1>, <p>, , ** и др. - для организации текста и мультимедийных элементов.

**<b>, <i>, <u>, , ** и др. - для форматирования текста (жирный/курсив и пр.).

**<ul>, <ol> и ** - для создания маркированных и нумерованных списков.

<a> — для создания гиперссылки (с атрибутами).

<img>, <video>, <audio> и др. — для вставки изображений, видео, аудио и других мультимедийных элементов (с атрибутами).

<button>, <form>, <input>, <textarea> — для создания кнопок и форм для ввода.

Пример создания HTML-документа

Теперь, когда мы знаем основные теги, можно создать простую веб-страницу. Для этого в любом текстовом редакторе (например, Notepad или Visual Studio Code) нужно создать новый файл с расширением **.html.** Вставим в него следующий код:

----------HTML----------

```
<!DOCTYPE html>  # задали тип файла
<html>           # корневой тег
<head>           # заголовок (метаданные, ссылки на стили, скрипты)
  <title>Простейшая веб-страница</title>  # заголовок страницы, отображаемый браузером
</head>
<body>           # тело страницы – видимый контент
  <h1>Добро пожаловать!</h1>                                  # заголовок 1
  <p>Это простейшая веб-страница.</p>                         # текст
  <h2>Немного обо мне</h2>                                    # заголовок 2
  <p>Меня зовут Егор, и я учусь в Инженерной школе №1581</p>  # текст
  <img src=" https://i6.photo.2gis.com/images/branch/32/4503599639620975_c0cf.jpg" alt="Фото школы" height="250" width="350">  # картинка – фото школы из сети размер 350x250
  <h2>Сайт школы</h2>                                         # заголовок 2
  <ul>
    <li><a href="https://lycc1581.mskobr.ru/">Инженерная школа №1581</a></li>  # ссылка
  </ul>
</body>
</html>
```

Сохраним файл и откроем его в браузере, отобразится созданная веб-страница - **см. РИС. 1**.

3.2. Основы CSS

CSS (Cascading Style Sheets) — это язык стилей, используемый для описания внешнего вида HTML-документов. С его помощью можно задавать цвета, шрифты, отступы, выравнивание и другие визуальные параметры элементов на веб-странице. CSS делает веб-страницы более привлекательными и удобными для пользователей.

3.2.1. Назначение стилей и их объявление

CSS позволяет отделить структуру документа (HTML) от его представления (стилей). То есть, **изменив один файл CSS, можно изменить внешний вид всех страниц сайта**.

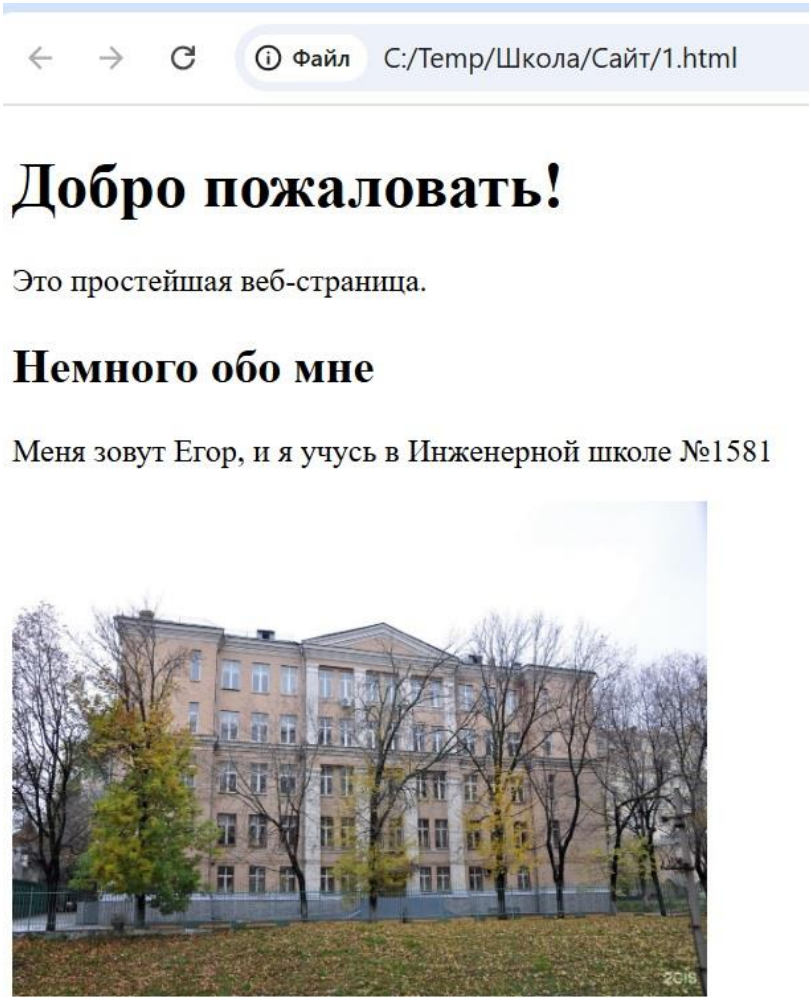

РИС. 1

Это особенно полезно для проектов, где множество страниц используют одни и те же стили, таким образом, разработчикам можно сосредоточиться на содержимом страниц, не беспокоясь о том, как они будут выглядеть. Кроме того, CSS позволяет создавать **адаптивные дизайны**, которые автоматически подстраиваются под различные устройства и экраны.

Объявление стиля состоит из двух частей: **селектора** и **объявления**. Объявление состоит из двух частей: имя свойства (например, «color») и значение свойства («blue»). Селектор сообщает браузеру, какой именно элемент форматировать, а в блоке объявления (фигурные скобки) перечислены форматирующие команды: свойства и их значения – **РИС. 2**.

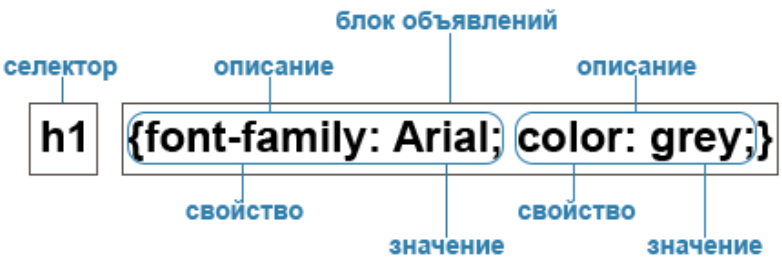

РИС. 2

3.2.2. Подключение CSS к HTML

Существует несколько способов подключения CSS к HTML-документу:

Встроенные стили

Встроенные стили указываются непосредственно в HTML-элементах с помощью атрибута **«style»**. Этот метод подходит для небольших изменений, но не рекомендуется для проектов, т.к. встроенные стили делают код менее читаемым и сложным для поддержки.

----------HTML----------

```
<p style="color: blue; font-size: 20px;">Текст со встроенным стилем цвет-размер</p>
```

Внутренние стили

Внутренние стили размещаются внутри тега **«style»** в разделе **«head»** HTML-документа. Этот метод удобен для страниц с уникальными стилями, но если страниц с одинаковыми стилями много, их использование приводит к усложнению кода.

----------HTML----------

```
<head>
  <title>Внутренний стиль</title>
  <style>
    p {
      color: blue;
      font-size: 20px;
    }
  </style>
</head>
<body>
  <p>Пример текста с внутренним стилем</p>
</body>
```

Внешние стили

Внешние стили хранятся в отдельном файле с расширением **«.css»** и подключаются к HTML-документу с помощью тега **«link»**. Этот метод наиболее предпочтителен для крупных проектов, так как позволяет централизованно управлять стилями, поскольку все стили веб-сайта задаются находятся в одном месте.

----------HTML----------

```
<head>
  <title>Внешний стиль</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <p>Пример текста с внешними стилями</p>
</body>
```

---------- styles.css ----------

```
p {
   color: blue;
   font-size: 20px;
}
```

3.2.3. Селекторы и их применение

Селекторы в CSS используются для выбора HTML-элементов, к которым будут применяться стили. Вот несколько основных селекторов:

Селектор по тегу

Применяет одинаковый стиль ко всем элементам определенного типа.

---------- styles.css ----------

```
p {
   color: blue;
}
```

Селектор по классу

Применяет стили к элементам с определенным классом. Классы задаются с помощью атрибута **«class»**. Этот метод позволяет более гибко управлять стилями, так как один и тот же класс можно применять к разным элементам на странице.

----------HTML----------

```
<p class="highlight">Текст с классом</p>
```

---------- styles.css ----------

```
.highlight {
   color: red;
}
```

Селектор по идентификатору

Применяет стили к элементу с определенным идентификатором. Идентификаторы задаются с помощью атрибута **«id»**. Идентификатор должен быть уникальным на странице, т.е. один и тот же «id» не может быть присвоен нескольким элементам.

----------HTML----------

```
<p id="unique">Текст с идентификатором (уникальный)</p>
```

---------- styles.css ----------

```
#unique {
   color: green;
}
```

3.2.4. Основные стили для текста, фона и элементов

Стили для текста	**Стили для фона**
«color» – задает цвет текста	**«background-color»** – задает цвет фона
«font-size» - задает размер шрифта	**«background-image»** –изображение фона
«font-family» - задает семейство шрифтов	**«background-repeat»** – задает повторение фона
«text-align» - задает выравнивание текста	
Стили для элементов	
«width» и «height» – ширина и высота	**«margin»** – задает внешние отступы
«border» – задает границу элемента	**«padding»** – задает внутренние отступы

3.2.5. Пример реализации CSS

Подключим файл внешних стилей styles.css к созданному ранее файлу HTML:

<link rel="stylesheet" href="C:\Temp\Школа\Сайт\styles.css">

---------- styles.css ----------

```
body { font-family: Arial, sans-serif;
   background-color: #90a5f0; }
h1 { color: #184;
   font-size: 20px; }
p {  color: #780;
   font-size: 14px;
   text-align: justify; }
```

И получим ту же веб-страницу с примененными внешними стилями – см. **РИС. 3.**

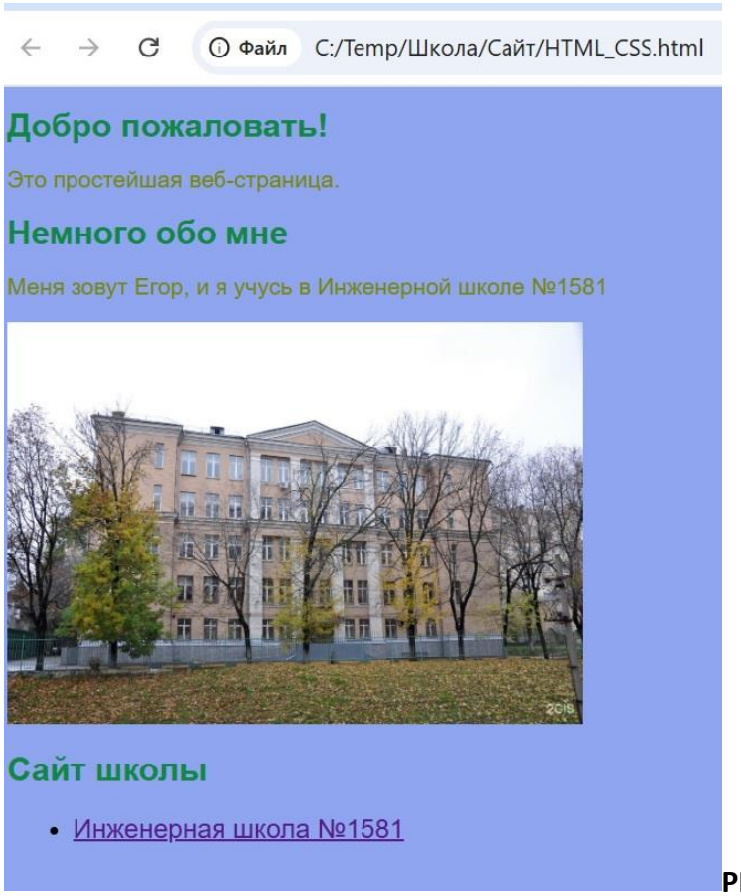

РИС. 3

4. Особенности фреймворка Django

4.1.Основные сведения о Django

Django — это бесплатный высокоуровневый фреймворк с открытым исходным кодом, предназначенный для создания приложений на языке Python. Его основная цель — помочь разработчикам быстро и безопасно создавать серверную часть сайтов (**backend**).

Django обладает обширным набором инструментов для типичных задач веб-разработки, таких как создание и структурирование приложения, работа с базами данных, внедрение бизнес-логики, управление пользовательскими аккаунтами, обработка форм и другое. Использование этих инструментов позволяет разработчикам сконцентрироваться на логике проекта, не отвлекаясь на технические детали реализации.

Например, при создании обычной формы регистрации на чистом Python, разработчику приходится писать отдельный код для валидации данных, хеширования паролей и сохранения информации в базе данных, настроить обработку ошибок и реализовать систему уведомлений о статусе регистрации. Набор встроенных инструментов и функций Django существенно упрощает эту задачу.

4.2.Встроенные инструменты Django

ORM (object-relational mapping)

Позволяет писать запросы к базам данных на Python вместо SQL. Абстрагируя взаимодействие с базами данных, автоматически генерирует схемы и упрощает разработку.

Система аутентификации

Обеспечивает комплексное управление учётными записями, группами и правами доступа. Упрощает реализацию функций регистрации, входа и выхода пользователей, контролирует доступ к различным разделам сайта и управляет клиентскими сессиями через cookies.

Шаблонизатор

Инструмент для создания динамического HTML-кода. Позволяет внедрять переменные в шаблоны для динамического отображения данных на веб-страницах. С его помощью можно создать базовый шаблон с общей структурой сайта, а затем расширять его для отдельных страниц, добавляя уникальный контент: текст, изображения, информацию о товарах, и др.

Работа с формами

Встроенная библиотека для отображения и проверки форм. Упрощает создание форм ввода данных и их валидацию по заданным критериям. Например, можно создать форму регистрации с полями для имени пользователя, email и пароля. Когда пользователь заполнит и отправит такую форму, Django проверит корректность ее заполнения.

URL-маршрутизация

Удобная система для связывания URL-адресов с функциями обработки запросов. Она позволяет создавать понятные ссылки на страницы вашего веб-приложения.

Встроенная административная панель

Автоматически генерируемый интерфейс для управления данными приложения. Она позволяет легко просматривать, добавлять, редактировать и удалять записи в базе данных без написания дополнительного кода.

Интернационализация

Встроенная поддержка многоязычности, локализация форматов даты, времени и чисел.

Защита от распространённых уязвимостей

Django обеспечивает защиту от SQL-инъекций, межсайтового скриптинга (XSS), подделки межсайтовых запросов (CSRF), кликджекинга и удалённого выполнения кода.

4.3. Архитектура Django

Django использует архитектуру **Model-View-Controller (MVC)**, которая структурирует код и разделяет приложение на три компонента:

Model (модель) - отвечает за структуру данных и бизнес-логику приложения.

View (представление) - отвечает за отображение информации.

Controller (контроллер) - обрабатывает запросы пользователя, взаимодействует с моделью и передаёт данные в представление.

Рассмотрим, как работает MVC в Django на примере интернет-магазина:

- Пользователь запрашивает страницу товара.
- URL-маршрутизатор Django направляет запрос к соответствующему Controller.
- Controller обращается к Model за данными о товаре.
- Model извлекает информацию из базы данных.
- Controller передаёт полученные данные в View.
- View формирует HTML-страницу с информацией о товаре.
- Django отправляет готовую страницу пользователю в ответ на его запрос.

4.4. Особенности Django

1. Полнота. Django позволяет разработчикам создавать проекты без сторонних элементов.

2. Универсальность. Подходит для программирования приложений и сайтов любого типа.

3. Высокий уровень надежности.

4. Масштабируемость. Любой элемент можно модифицировать, не трогая остальное.

5. Уровень безопасности. У библиотеки поддерживаются инструменты для защиты от хакерских атак и взлома.

6. Переносимость и гибкость.

7. Открытость. Любой желающий способен использовать фреймворк Django для создания сайтов и приложений, включая их коммерческое распространение.

8. Устаревший ORM.

9. Отсутствие многозадачности. Отдельные процессы в Джанго не могут работать с несколькими запросами одновременно.

5. Основные правила UX/UI

UX (User Experience) дизайн фокусируется на создании удобного и приятного опыта для пользователей, b включает в себя множество аспектов, таких как исследование пользователей, создание прототипов, тестирование и анализ. Это процесс, который требует глубокого понимания поведения и потребностей пользователей.

UI (User Interface) дизайн отвечает за визуальную составляющую интерфейса, за то, как продукт выглядит, и как пользователи взаимодействуют с ним на визуальном уровне.

Понимание основных принципов и лучших практик в этой области помогает создавать продукты, которые будут функциональными и привлекательными для пользователей.

5.1. Основные принципы UX-дизайна

5.1.1. Понимание пользователя

Основной принцип UX дизайна — это глубокое понимание потребностей и ожиданий пользователей. Для этого используются методы исследования, такие как интервью, опросы и тестирование прототипов. Собираются данные о том, как пользователи взаимодействуют с продуктом, какие задачи они выполняют и какие проблемы у них возникают. Важно не только собрать данные, но и правильно их интерпретировать, чтобы сделать обоснованные выводы и принять правильные решения в дизайне.

5.1.2. Интуитивная навигация

Навигация должна быть простой и интуитивно понятной. Пользователи должны легко находить нужную информацию и выполнять задачи без необходимости долго разбираться в интерфейсе, легко понять, как перемещаться по интерфейсу, даже если они используют его впервые. Это достигается за счет использования знакомых элементов, таких как меню, кнопки и ссылки, а также за счет логичной структуры и организации контента.

5.1.3. Консистентность

Консистентность в дизайне помогает пользователям быстрее адаптироваться к интерфейсу. Это касается как визуальных элементов (цвета, шрифты, иконки), так и функциональных (расположение кнопок, поведение элементов). Консистентный дизайн снижает когнитивную нагрузку на пользователя. Консистентность также помогает создать единый стиль и бренд, что делает продукт более узнаваемым и профессиональным.

5.1.4. Обратная связь

Пользователи должны получать обратную связь на свои действия. Это может быть визуальная (изменение цвета кнопки при нажатии), звуковая (звуковые сигналы) или текстовая (сообщения об ошибках или успехах). Обратная связь помогает пользователям понимать, что их действия имеют результат. Важно, чтобы обратная связь была своевременной и понятной, чтобы пользователи могли быстро реагировать на изменения и корректировать свои действия при необходимости.

5.1.5. Доступность

Доступность означает, что продукт должен быть удобен для всех пользователей, включая людей с ограниченными возможностями. Это включает использование контрастных цветов, альтернативного текста для изображений и возможность навигации с клавиатуры. Это не только улучшает пользовательский опыт, но и расширяет аудиторию продукта.

5.2. Основные принципы и правила UI-дизайна

5.2.1. Визуальная иерархия

Визуальная иерархия помогает пользователям быстро ориентироваться в интерфейсе. Важные элементы должны быть более заметными, чем второстепенные. Это можно достичь с помощью размера, цвета, контраста и расположения элементов. Особенно важно это для сложных интерфейсов с множеством элементов и информации.

5.2.2. Простота и минимализм

Простота и минимализм помогают избежать перегрузки пользователя информацией. Нужно убирать все лишние элементы и оставлять только то, что действительно необходимо. Это улучшает восприятие и делает интерфейс более понятным. Это помогает создать более эстетичный и профессиональный вид продукта, что особенно важно для мобильных устройств, где пространство ограничено и каждая деталь имеет значение.

5.2.3. Цветовая палитра

Цветовая палитра должна быть ограниченной и гармоничной, это играет важную роль в восприятии и эмоциональном отклике пользователей. Правильное использование цветов помогает создать нужное настроение и улучшить восприятие информации. Цвета должны быть согласованы с брендингом и не вызывать дискомфорта у пользователей.

5.2.4. Типографика

Необходимо использовать читабельные шрифты, следить за их размером и интервалами. Контраст между текстом и фоном должен быть достаточным для комфортного чтения. Важно также учитывать контекст и аудиторию, чтобы выбрать подходящие шрифты и стили для конкретного продукта.

5.2.4. Анимации и переходы

Анимации и переходы должны быть плавными и не отвлекать от основной задачи. Анимации могут помочь пользователям понять, что происходит в интерфейсе и как взаимодействовать с ним. Анимации и переходы также могут добавить элемент удовольствия и интерактивности в интерфейс. Они могут использоваться для привлечения внимания, улучшения навигации и создания более живого и динамичного опыта. Важно, чтобы анимации были функциональными и не перегружали интерфейс.

6. Основные этапы и правила разработки веб-сайтов

ЭТАП 1: Идея и планирование.

Определение цели сайта и целевой аудитории на которую он будет ориентирован. На этом этапе необходимо определить цель сайта, изучить целевую аудиторию, ее потребности. Важно провести всесторонний анализ конкурентов.

ЭТАП 2: Дизайн и прототипирование.

Этот этап включает создание каркасных макетов, разработку прототипов и визуальный дизайн. Каркасные помогают визуализировать структуру сайта и понять, как пользователи будут взаимодействовать с ним. Прототипы позволяют тестировать пользовательский интерфейс и вносить изменения до начала разработки. Качественный визуальный дизайн сайта - один из важнейших принципов UX/UI.

ЭТАП 3: Разработка и программирование.

Когда дизайн утверждён, начинается этап разработки и программирования. На этом этапе происходит вёрстка (преобразование дизайна в HTML/CSS), программирование (разработка функционала сайта, как фронтенд, так и бэкенд с использованием языков программирования) и интеграция с CMS (если используется система управления контентом).

ЭТАП 4: Тестирование и отладка.

Перед запуском сайта необходимо провести тщательное тестирование и отладку. Включает функциональное тестирование (проверка всех функций сайта на работоспособность), кроссбраузерное тестирование (убеждение, что сайт корректно отображается во всех популярных браузерах) и тестирование на различных устройствах (проверка, как сайт выглядит и работает на мобильных устройствах, планшетах и настольных компьютерах).

ЭТАП 5: Размещение сайта в интернет.

Выбор и регистрация доменного имени. Файлы сайта размещают на сервере провайдера (хостинга) и производят нужные настройки веб-сервера. На этом этапе сайт пока закрыт для посетителей.

ЭТАП 6: Наполнение контентом и публикация.

Сайт наполняют содержимым (контентом) — текстами, изображениями, файлами для скачивания и так далее.

ЭТАП 7: Мониторинг и поддержка.

Регулярное обновление контента, исправление ошибок и обеспечение безопасности сайта. Подключение веб-аналитики и анализ поведения пользователей. Отслеживание производительности сайта.

Список используемых источников

1. https://habr.com/ - Хабр. Сообщество IT-специалистов.
2. https://skillbox.ru/media/ - Отраслевое издание Skillbox Media.
3. https://sky.pro/wiki/ - отраслевое издание SkyPro-Wiki.
4. https://ru.wikipedia.org/wiki/ - Википедия- свободная энциклопедия.