МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
СЕВЕРО-КАВКАЗСКАЯ ГОСУДАРСТВЕННАЯ АКАДЕМИЯ

А. М. Кочкаров
Р. И. Селимсултанова

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ В ИНТЕРНЕТ

Лабораторный практикум
для обучающихся 3 курса направления подготовки
01.03.02 Прикладная математика и информатика

Черкесск
2025

УДК 004.4
ББК 32.973
К75

Рассмотрено на заседании кафедры «Математика»
Протокол №1 от 2 «09» 2024 г.
Рекомендовано к изданию редакционно-издательским советом СКГА.
Протокол № 27 от 07 «11» 2024 г.

Рецензент: Бежанова Е.Х. – к.ф-м.н., доцент кафедры «Математика»

К75 **Кочкаров, А.М.** Технологии программирования в интернет: лабораторный практикум для обучающихся 3 курса направления подготовки 01.03.02 Прикладная математика и информатика / А. М. Кочкаров, Р.И. Селимсултанова. – Черкесск: БИЦ СКГА, 2025. – 92 с.

Лабораторный практикум предназначен для изучения и освоения языка разметки HTML и основ web-программирования обучающимися направления подготовки 01.03.02 Прикладная математика и информатика. Описываются современные технологии создания сайтов: язык разметки гипертекста HTML, каскадные таблицы стилей CSS, язык программирования Java Script

Лабораторный практикум включает в себя лабораторные работы по освоению языка разметки HTML. Даны основные теоретические сведения о тегах, необходимые для выполнения лабораторных и индивидуальных заданий. Первая часть пособия содержит теоретический материал и практические задания по языку HTML и технологии CSS..

УДК 004.4
ББК 32.973

Содержание

Введение

В современном мире для любой коммерческой компании является большим плюсом собственное представительство в интернете. Многомиллионная аудитория Всемирной паутины - это широкие возможности для развития бизнеса, продвижения продукции и услуг. Сегодня веб-сайт является мощным маркетинговым инструментом. Это один из наиболее эффективных способов заявить о себе потенциальным покупателям и пользователям. Наличие у фирмы собственного сайта считается признаком надежности, стабильности и профессионализма. Цель практикума – дать обучающимся представление об основах Web-технологий, научить их создавать свои собственные сайты на базе изучения тегов html, содержательно обучить технологии CSS.

Данный лабораторный практикум посвящен изучению основ технологий программирования в интернет. В их рамках рассматриваются вопросы создания Web-страниц с помощью HTML. Все лабораторные работы посвящены изучению языка гипертекстовой разметки HTML. После каждой лабораторной работы выполняются индивидуальные задания, также прилагаются вопросы для самоконтроля.

После выполнения всех лабораторных работ, обучающиеся будут уметь: разрабатывать общую структуру сайтов, формировать страницы, из которых состоит эта структура, добавлять интерактивные средства и эффекты мультимедиа и, наконец, размещать готовые сайты в Internet путем загрузки их на Web-сервер.

Знакомство с Visual Studio Code

Visual Studio Code или просто **VS Code** – это бесплатный, популярный и имеющий множество дополнений текстовый редактор, который в первую очередь предназначен для создания и отладки современных веб- и облачных приложений. Разработан он компанией Microsoft и доступен для операционных систем Windows, MacOS и Linux.

После установки Visual Studio Code, которая не должна вызвать никаких проблем, пользователь получает полноценную платформу для написания кода. Без дополнительных манипуляций здесь доступен Emmet, а функционал легко дополняется расширениями через Visual Studio Marketplace, доступный также через интерфейс редактора. В магазин можно попасть, кликнув на иконку Extensions расположенную в боковой панели, рисунок 1. или нажав сочетание клавиш **Ctrl + Shift + X**

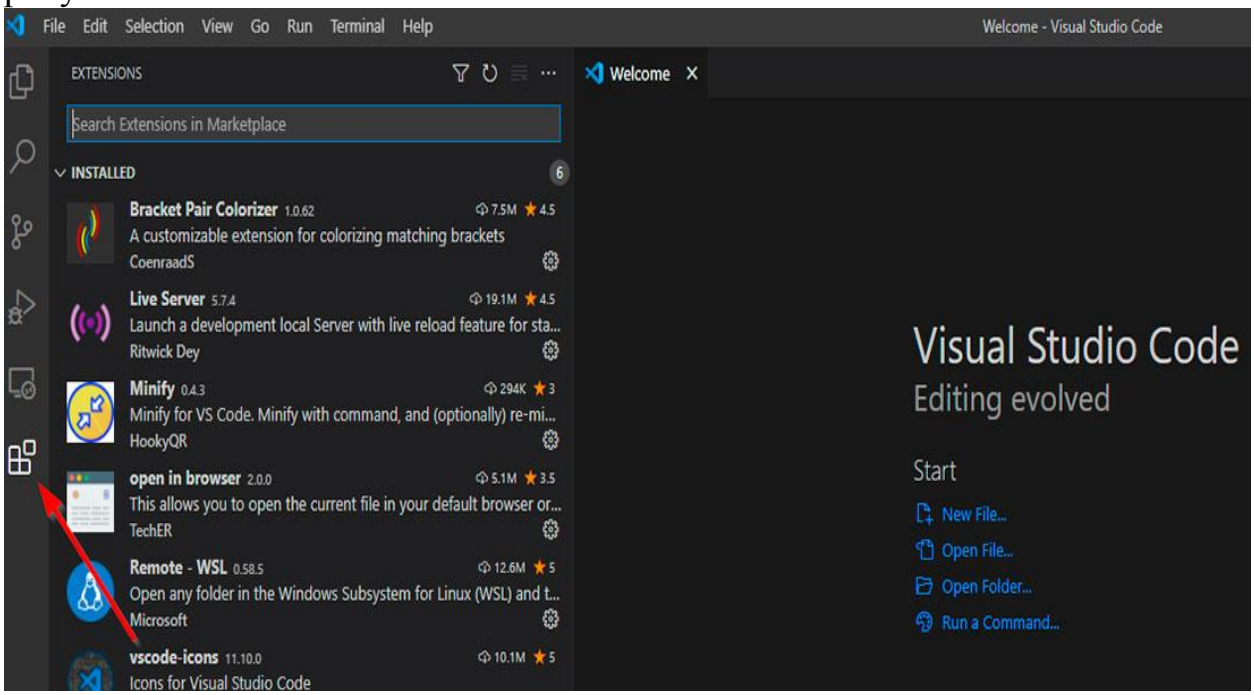


Рисунок 1.1 – Интерфейс редактора

Расширения для Frontend разработки.

Emmet

Упрощает процесс написания кода, позволяя с помощью коротких команд формировать большие структуры кода. Плагин уже встроен в VS code, а полный список возможностей можно посмотреть на официальном сайте расширения. Короткий пример позволяющий проиллюстрировать работу Emmet: вам необходимо создать 20 элементов нумерованного списка, вы набираете **ol>li*20** и нажимаете **Tab** или **Enter** – список из 20 элементов готов.

Live Server

Простое, но полезное расширение, позволяющее отслеживать на странице, без ее перезагрузки, изменения в JavaScript, HTML и CSS. Плагин не является полноценным сервером, но его функционал можно расширить в этом направлении с помощью других дополнений.

ESLint

Утилита для проверки стандарта написания кода на JavaScript. Дополнение относится к программам, называемые линтеры, которые проверяют код на правильность написания и соответствия современным практикам кодирования. После анализа ESLint выделяет ошибки и неточности, которые теперь легко увидеть и исправить.

Auto Rename Tag

Автоматически переименовывает парные теги HTML/XML – таким образом остается поменять только один из двух тегов закрывающий или открывающий.

Prettier - Code formatter

Позволяет автоматически приводить код в порядок согласно внутренним правилам плагина и индивидуальным настройкам пользователя. Проще говоря – расставит пробелы и переносы, заменит одинарные кавычки на двойные или наоборот и так далее.

CSS Peek

Позволяет смотреть и редактировать стили прямо в html файле. Просто зажимаете Ctrl далее наводите на класс и кликаете. После этих манипуляций появляется всплывающее окно – соответствующий файл CSS, куда и можно вносить изменения.

Debugger for Chrome

Расширения для отладки JavaScript кода. Данный плагин позволяет работать в браузерах на базе Chromium, для Firefox придется установить Debugger for Firefox.

Настраиваем VS Code под себя

Для более комфортной работы над проектами редактор содержит массу пользовательских настроек от возможности увеличить или уменьшить шрифт до включения автоматического сохранения файлов. Попасть в панель можно двумя способами:

- **File – Preferences – Settings**;
- **Ctrl + Shift + p** – откроется поисковая строка где вначале вы увидите Preferences: Open User Settings

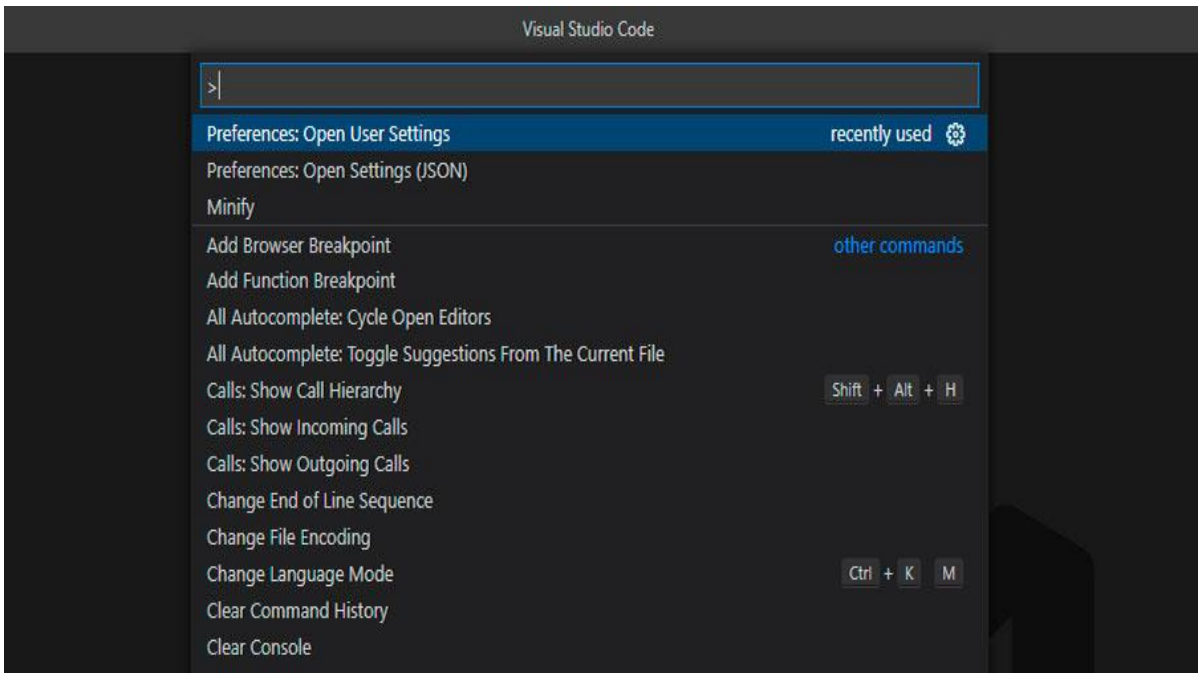


Рисунок 1.2 – Панель настроек VS Code

Рассмотрим некоторые полезные настройки

Auto Save – можно выбрать автоматическое сохранение, не сохранять автоматически, сохранять при смене или потере фокуса.

Font Size – размер шрифта в пикселях.

Line Height – высота строки.

Console: Font Size – размер шрифта для терминала.

Font Family – выбор шрифта.

Tab Size – можно задать количество пробелов в табуляции.

Open Files in New Window – позволяет настраивать создание файлов в новом окне или табе.

Word Wrap – управлением тем, как следует переносить строки – по ширине окна, не переносить и т.д.

Format On Paste – определяет, будет ли редактор автоматически форматировать вставленный код.

Minimap – включить или отключить миникарту.

Confirm Delete – убрать или оставить окно с подтверждением удаления файла.

Format On Save – настройка, отвечающая за автоматическое форматирование кода после сохранения файла. Модуль форматирования должен быть установлен отдельно, например, «Prettier - Code formatter», описанный выше.

Semi – включить / отключить автоматическое добавление ; в конце строи – настройка от «Prettier - Code formatter».

Это далеко не все настройки доступные в VS code и даже пробежаться по всем вскользь будет не так быстро. Для того, чтобы лучше ориентироваться во всем этом многообразии, можно воспользоваться левым меню, которое делит весь список на категории. Настройки чьи дефолтные значения были изменены можно увидеть в формате JSON кликнув на соответствующую иконку.

Лабораторная работа №1

Тема: Структура HTML-страницы, основные теги для форматирования текста

Цель: Закрепление основных понятий о структуре HTML-страницы, изучение основных тегов

Краткие теоретические сведения

Для того, чтобы понять структуру и сценарий Web-документа, необходимо рассмотреть несколько Web-страниц и выявить общие элементы.

Любой Web-документ состоит из тегов. Рассмотрим основные теги, входящие в каждый Web-документ. Прежде всего, это <HTML></HTML> Отличительный признак HTML-документа. Одним из принципов языка разметки является многоуровневое вложение элементов. HTML является самым внешним, так как между его стартовым и конечным тегами должна находиться вся Web-страница. Также основным тегом является <HEAD></HEAD> Область заголовка Web-страницы. Иными словами, ее первая часть. Так же, как и HTML, HEAD служит только для формирования общей структуры документа.

<BODY></BODY> Этот элемент заключает в себе гипертекст, который определяет собственно Web-страницу. Эта та часть документа, которую разрабатывает автор страницы и которая отображается браузером. Соответственно, конечный тег этого элемента надо искать в конце HTML-файла. Внутри BODY можно использовать все элементы, предназначенные для дизайна Web-страницы. Внутри стартового тега элемента BODY можно расположить ряд атрибутов, обеспечивающих установки для всей страницы целиком. Без этих тегов невозможно создать ни одну Web-страницу.

Структура Web-страницы:

```
<HTML>
  <HEAD>
    <TITLE>Название страницы</TITLE>
   </HEAD>
<BODY>
     Тело страницы
</BODY>
</ HTML>
```

Естественно, что конечным тегом </HTML> заканчиваются все гипертекстовые документы.

<HEAD></HEAD> Область заголовка Web-страницы. Иными словами, ее первая часть.

<TITLE></TITLE> Элемент для размещения заголовка Web-страницы. Строка текста, расположенная внутри, отображается не в документе, а в заголовке окна браузера. Эта особенность часто используется для

организации поиска в WWW. Поэтому авторы, создающие Web-страницы, должны позаботиться о том, чтобы строка внутри TITLE, не будучи слишком длинной, достаточно отображала назначение документа.

<BODY></BODY> Этот элемент заключает в себе гипертекст, который определяет собственно Web-страницу. Эта та часть документа, которую разрабатывает автор страницы и которая отображается браузером. Соответственно, конечный тег этого элемента надо искать в конце HTML-файла. Внутри BODY можно использовать все элементы, предназначенные для дизайна Web-страницы. Внутри стартового тега элемента BODY можно расположить ряд атрибутов, обеспечивающих установки для всей страницы целиком. Рассмотрим их по порядку.

Заголовки

В примере для заголовка используются теги <H1></h1>.

Существует шесть уровней заголовков, которые обозначаются H1...H6. Заголовок уровня H1 самый крупный, а уровень H6 - самый маленький заголовок. Для заголовков можно использовать атрибут **align**, задающий выравнивание влево, по центру или вправо.

<COMMENT></COMMENT> Текст комментария.

Текст, помещенный внутри COMMENT, игнорируется браузером. COMMENT может располагаться в любом месте кода Web-страницы. Без конечного тега, здесь по-видимому не обойтись: комментарий должен быть отделен от основного текста.

Существует и другой способ обозначения комментария. Он заключается в использовании восклицательного знака и обрамлении текста комментария двойным тире. Например:

<!--Строка комментария-->

Внутри подобной конструкции можно помещать и теги: они не будут восприниматься браузером.

Теги для форматирования текста

Текст - единственный объект Web-страницы, который не требует специального определения. Иными словами, произвольные символы интерпретируются по умолчанию как текстовые данные. Но для форматирования текста существует большое количество элементов.

<P></P> Элемент абзаца (paragraph). Он позволяет использовать только начальный тег, так как следующий элемент P обозначает конец предыдущего и начало следующего абзаца. Вместе с элементом P можно использовать атрибут выравнивания Align.

**
** Элемент, обеспечивающий принудительный переход на новую строку. Он имеет только стартовый (одиночный) тег. В месте его размещения строка заканчивается, а оставшийся текст печатается с новой строки.

<NOBR></NOBR> Этот элемент по своему действию является прямой противоположностью предыдущему. Текст, заключенный между его тегами, будет выведен в одну строку. Длинная строка не уместится на экране, и для ее просмотра придется использовать горизонтальную полосу прокрутки.

<PRE></PRE> Элемент для обозначения текста, отформатированного заранее. Подразумевается, что текст будет выведен в том виде, в каком он был подготовлен пользователем.

<DIV></DIV> Он позволяет выравнивать содержимое по левому краю, по центру или по правому краю. Для этого стартовый тег должен содержать соответствующий атрибут **Align.**

<B></B> Выделение текста полужирным шрифтом. Очень популярный элемент. Использование полужирного шрифта - прием, позаимствованный из текстовых редакторов.

<BIG></ BIG> Увеличенный размер шрифта

<SMALL></SMALL> Уменьшенный размер шрифта

<I></i> Выделение текста курсивом

<STRIKE></STRIKE> Элемент, создающий перечеркнутое начертание текста. В настоящее время его заменяют более простым: <S></S>

<U></u> Подчеркнутое начертание текста.

<STRONG></STRONG> Элемент, отвечающий за выделение текста. Обычно его применение равносильно использованию элемента для выделения полужирным.

**** Элемент, создающий эффект нижнего индекса.

**** Элемент, создающий эффект верхнего индекса.

<CITE></CITE> Предполагается, что этот элемент может быть использован для форматирования цитат и ссылок в обычном понимании этого слова. Текст, расположенный внутри него, выводится по умолчанию курсивом.

<FONT></FONT> управляет размером, начертанием и цветом текста. Имеет два атрибута **COLOR, FACE, SIZE.**

Например: <FONT COLOR= red > текст </ FONT> - выведет текст красным цветом (можно писать название цвета или его код, который можно найти в Adobe Photoshop – в его палитре инструментов.

Например: <FONT FACE= Arial > текст</ FONT> - приведет к выводу текста указанным шрифтом.

<FONT SIZE = 4 > текст</ FONT> - изменяет размер шрифта фрагмента текста, можно использовать в любом месте строки. Атрибуты могут перечисляться через пробел в произвольном порядке.

Например: **<FONT SIZE = 4 FACE= Arial COLOR= red> текст </ FONT>**

Цвета обычно кодируются в формате RGB (красный, зеленый, синий). Записывается это как «#RRGGBB», где шестнадцатеричное значение составляющих: RR-красный; GG - зеленый; BB - синий.

00 — обозначает отсутствие яркости, а FF — максимальную яркость.

Шестнадцатеричной называется система счисления, в которой для записи чисел используются 16 символов: 0,1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Таким образом, число FF соответствует десятичному числу 256.

Примеры некоторых цветов, закодированных таким образом:

#000000 - Черный #FFFFFF - Белый #FF000 - Красный #FFFF00 -Желтый	#00FFFF - Бирюзовый #0000FF- Синий #FF00FF- Фиолетовый #FF800- Оранжевый #8000FF- Лиловый

Теги для создания фона

Один из самых полезных для дизайна - атрибут, определяющий фон страницы. Его появление можно уподобить маленькой революции в WWW, так как одинаковые серые Web-страницы благодаря ему расцвели яркими цветными узорами. Фон страницы задается в начале основной части документа, т.е. атрибутами тэга <BODY>.

Атрибуты тэга <BODY> для изменения цвета текста и фона:

TEXT – задает цвет текста, например: <BODY TEXT= 7EA3B8> значением атрибута может служить как название цвета, так и его код.

BGCOLOR–задает цвет фона, например:

<BODY BGCOLOR = BLACK >

<BGCOLOR= #RRGGBB>

<BODY **BACKGROUND** = *.gif > - использовать в качестве фона изображения из файла.

BGPROPERTIES=FIXED - создание фона – водяного знака (фон, который не перемещается вместе с текстом) <BODY **BGPROPERTIES=FIXED**>

Поскольку фон страницы может изменяться, необходимо иметь возможность подбирать соответствующий цвет текста. Для этого имеется следующий атрибут:

text= #RRGGBB

Для задания цвета гиперссылок используется атрибут:

link= RRGGBB

Также можно задать цвет для использованных гиперссылок:

vlink= RRGGBB

Гипертекст, расположенный внутри элемента BODY, может иметь произвольную структуру. Ее определяют в первую очередь назначение Web-страницы и фантазия разработчика.

Вставка графических изображений

<**IMG SRC**=*имя файла.расширение* > вставка картинки из файла

Атрибуты тэга <IMG>:

SRC=*имя файла.расширение*

ALT=*название картинки или поясняющий текст* (Используется в качестве текстового описания к данному изображению, появляется при наведении указателя мыши на картинку)

WIDTH = *ширина* (в пикселях или в %)
HIGHT = *высота* (в пикселях или в %)
BORDER = *число в пикселях* определяет рамку вокруг изображения
HSPACE = *число в пикселях*
VSPACE = *число в пикселях*

LEFT
ALIGN = *RIGHT*
CENTER

Используется, чтобы сдвинуть картинку к левому, правому краю, по центру и установить обтекание картинки текстом, а затем продолжить текст в пустой области за объектом.

LEFT
CLEAR = *RIGHT*
ALL

Используется, чтобы очистить поле слева, справа или с обеих сторон графического объекта, а затем продолжить текст в пустой области за объектом.

Вставка видео и звука

<**BGSOUND SRC** =*имя файла*> - вставка звукового файла
Например: <**BGSOUND SRC**=BLIP.WAV>
<IMG DYNSRC =*имя файла*>- вставка видео файла

n
LOOP = *INFINITE*

Количество повторений видео или звукового клипа и непрерывное повторение соответственно.

Настройка Visual Studio Code

В этом подразделе будет пояснено, как настроить и использовать VS Code для редактирования HTML-документов с примерами и полезными инструментами.

Шаг 1: Установите Visual Studio Code.

Перейдите на офици альный веб-сайт VS Code: https://code.visualstudio.com/download .

Загрузите и установите версию для вашей операционной системы (Windows, macOS, Linux).

Шаг 2: Создайте HTML-документ

Откройте VS Code.

Создайте на своем компьютере новую папку для своего проекта (например, MyHTMLProject).

В VS Code выберите "Файл" → "Открыть папку"... и выберите созданную папку.

Щелкните правой кнопкой мыши в проводнике файлов на левой панели и выберите "Создать файл". Назовите файл index.html .

Теперь у вас есть пустой HTML-файл, открытый для редактирования.

Шаг 3: Написание HTML-кода

Когда вы начнете редактировать файл index.html VS Code предложит автоматическое завершение и подсказки для HTML.

Пример базового HTML-документа:

```
<!DOCTYPE html>
<html lang="ru">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Моя первая страница</title>
</head>
<body>
   <h1>Привет, мир!</h1>
   <p>Это моя первая HTML-страница в VS Code.</p>
</body>
</html>
```

Шаг 4: Использование Emmet для ускорения вашей работы

VS Code по умолчанию поддерживает Emmet — это инструмент для быстрого написания HTML и CSS кода. Например, чтобы создать базовую структуру HTML документа, просто введите ! и нажмите Tab.

Пример: В пустом файле index.html Введите! и нажмите Tab. VS Code автоматически развернет это в базовый HTML-шаблон.

Шаг 5: Просмотр HTML-документа в браузере

Чтобы увидеть результат выполнения вашего HTML-кода в браузере, выполните одно из следующих действий:

Вариант 1: Откройте файл вручную

Перейдите к папке с вашим проектом на вашем компьютере.

Дважды щелкните по файлу index.html чтобы открыть его в браузере.

Вариант 2: Установите Live Server

Для удобного автоматического обновления страницы в браузере вы можете установить расширение Live Server:

В VS Code перейдите на вкладку "Расширения" (значок выполнен в виде квадрата с черточкой внутри) или нажмите Ctrl+Shift+X.

В строке поиска введите "Live Server" и установите расширение от Ritwick Dey.

После установки откройте файл index.html и щелкните по нему правой кнопкой мыши, затем выберите "Открыть с помощью Live Server".

Откроется браузер с вашим HTML-документом. Теперь при каждом сохранении файла (Ctrl+S) страница будет автоматически обновляться.

Шаг 6: Полезные расширения для работы с HTML

Чтобы улучшить ваш рабочий процесс с HTML, вы можете установить дополнительные расширения:

HTML Snippets — добавляет дополнительные шаблоны для работы с HTML.

Установите с помощью расширений (поиск по названию "HTML Snippets").

Prettier - Code Formatter — автоматически форматирует HTML-код, делая его чистым и аккуратным.

Установите Prettier с помощью расширений.

Установите его в качестве основного средства форматирования:

Откройте настройки (значок шестеренки в левом нижнем углу) → Настройки(Settings)→ Введите "Средство форматирования по умолчанию"(Default Formatter) в строке поиска.

Выберите Prettier в качестве средства форматирования по умолчанию.

Включите автоматическое форматирование при сохранении: настройка Format On Save.

Auto Close Tag — автоматически закрывает открытые HTML-теги, что ускоряет написание кода.

Тег Auto Close — автоматически закрывает открытые HTML-теги, что ускоряет написание кода.

Тег Auto Rename — синхронизирует изменения в открывающем и закрывающем тегах, что помогает избежать ошибок редактирования.

<!DOCTYPE html> - это объявление типа документа (англ. "Document Type Declaration", сокращенно DOCTYPE), которое указывает браузеру, что документ написан на HTML5. Эта строка должна быть первой в каждом HTML-документе.

Основные моменты, касающиеся <!DOCTYPE html>:Что делает DOCTYPE?:

Он сообщает браузеру, какой стандарт HTML следует использовать при отображении страницы. В HTML5 <!DOCTYPE html> является краткой и простой версией этого объявления. Это указывает на то, что используется последняя версия HTML (HTML5), но не содержит каких-либо подробных спецификаций или ссылок на внешние файлы DTD (определение типа документа), как это было в предыдущих версиях HTML.

Почему это важно: Если вы не укажете DOCTYPE или укажете его неправильно, браузеры могут переключиться в режим совместимости (quirks mode), в котором они будут отображать страницу как в более старых версиях HTML, что может привести к некорректному отображению сайта.

Правильный DOCTYPE гарантирует, что браузеры будут работать стандартизированным образом.

html lang="ru"> - это открывающий тег для элемента <html>, который указывает на то, что документ написан на русском языке. Этот атрибут используется для улучшения взаимодействия с браузерами, поисковыми системами и вспомогательными технологиями, такими как программы чтения с экрана. Когда атрибут lang особенно важен:

Если вы разрабатываете многоязычные веб-сайты.

Если ваш сайт ориентирован на международную аудиторию или использует разные языки для разных пользователей.

Если ваш сайт должен быть доступен для людей с ограниченными возможностями (например, поддержка программы чтения с экрана).

<meta charset="UTF-8">- обеспечивает корректное отображение символов (особенно важно для многоязычных сайтов).

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

- делает сайт адаптивным для мобильных устройств, улучшая его внешний вид и удобство использования.

Индивидуальное задание 1. Создать структуру HTML-страницы. Внутри созданной страницы использовать выше перечисленные основные теги. Написать ФИО, № группы, название кафедры, используя теги форматирования текста.

Пример выполнения индивидуального задания 1

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Создание структуры HTML-страницы, основные теги и теги
форматирования текста </title>
</head>
<BODY BGCOLOR="#add8e6 ">
    <H1><CENTER>Индивидуальное задание №1 </CENTER> </H1> <BR>
    <B>Название: <I>Структура HTML-страницы, основные теги и теги
форматирования текста.</I></B><BR><BR>
    <B>Цель работы: <I> Закрепление основных понятий о структуре HTML-
страницы, назначение основных тегов</I></B>
    <BR><BR><H2>Выполнил</H2>
    <STRONG>Ф.И.О.:</STRONG><U>Студентов Студент Студентович</u><BR>
    <STRONG>Группа:</STRONG><U>ПМИ-241</U><BR>
    <STRONG>Кафедра:</STRONG><U>"Математика"</U><BR></BODY>
    >
</html>
```

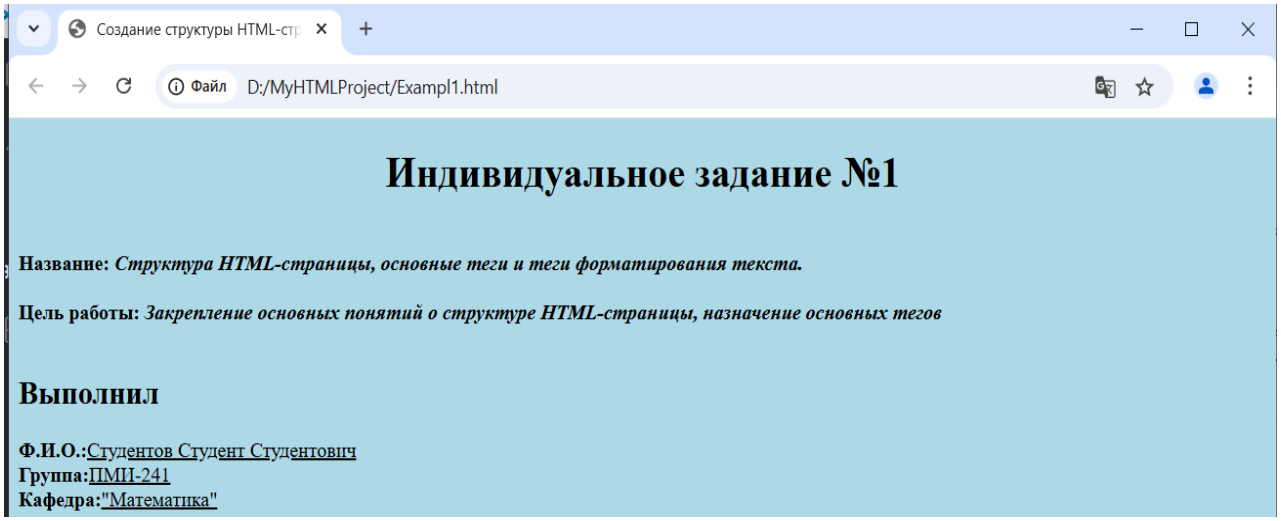


Рисунок 1 – Результат выполнения индивидуального задания 1

Индивидуальное задание 2. Форматирование текста и вставка графических изображений

Подготовка рабочей среды

1. Установите **Visual Studio Code** с сайта https://code.visualstudio.com (если ещё не установлен).
2. Создайте на рабочем столе или в любом другом месте папку с названием Lobachevsky_HTML.
3. Скопируйте в эту папку изображения:
   - lobach.jpg
   - giperb.webp
   - scale_1200.jpg
4. Создание HTML-документа

Откройте папку Lobachevsky_HTML в Visual Studio Code: в меню выберите Файл → Открыть папку.

5. Создайте новый файл: ПКМ в боковой панели → Создать файл → Назовите файл index.html.
6. Тексты перенести из текстовых файлов в HTML-документ копированием. Картинки вставить соответствующим тегом. Применить различное взаимное расположение текста и картинки: с обтеканием и без, выровнять по центру, по левому краю и т.д. Результат должен быть примерно такой, как на рисунке 2.

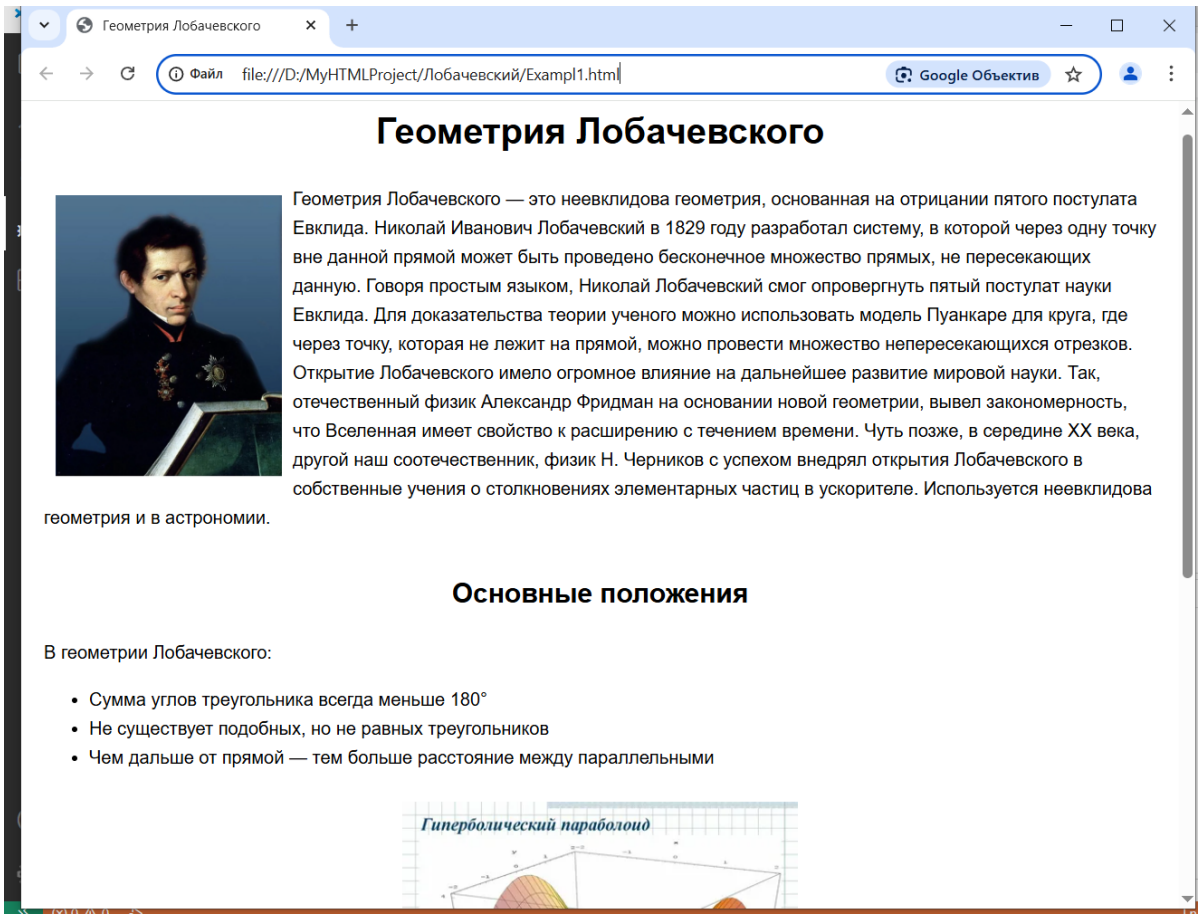


Рисунок 2 – Результат выполнения индивидуального задания 2

Лабораторная работа №2

Тема: Теги для вставки звука и видео

Цель: Изучение и закрепление знаний по тегам для вставки звука и видео

Вставка видео и звука

<**BGSOUND SRC** =*имя файла*> - вставка звукового файла
Например: <BGSOUND SRC="BLIP.WAV">
<IMG DYNSRC =*имя файла*>- вставка видео файла
Атрибут тэгов <BGSOUND SRC> и <IMG DYN SRC>

n

LOOP = *INFINITE*

Количество повторений видео или звукового клипа и непрерывное повторение соответственно.

Тег	Назначение
`<audio>`	Вставка звукового файла
`<video>`	Вставка видеофайла
`controls`	Показывает элементы управления (пауза, громкость и т.д.)
`autoplay`	Автоматическое воспроизведение (не всегда работает без `muted`)
`loop`	Зацикливание воспроизведения
`muted`	Воспроизведение без звука по умолчанию (полезно для автозапуска)
`<source>`	Указание источника медиа (можно добавить несколько форматов)

Индивидуальное задание 1. Вставка звука и видео

Действие:

1. Установите Visual Studio Code с сайта https://code.visualstudio.com, если он ещё не установлен.
2. Создайте новую папку, например: `media_html`.
3. Скопируйте в неё файлы для индивидуальному заданию по выбранной тематике, а также медиафайлы , например:`blip.wav`,`video.mp4`.
4. Откройте созданную папку в Visual Studio Code: **Файл → Открыть папку...**
5. Создайте новый файл `Exampl2.html`.
6. Убедитесь, что все медиафайлы находятся в **той же папке**, что и `Exampl2.html`.

Откройте файл через **Live Server** (или двойной клик → откроется в браузере или нажатием клавиши F5). Проверьте, звук воспроизводится и зацикливается. Видео запускается, имеет элементы управления.

Дополнительно (по желанию) можно разместить <audio> в скрытом виде: `<audio src="blip.wav" autoplay loop hidden></audio>`
Можно добавить постер (изображение-заставку) к видео:
`<video poster="preview.jpg" ...></video>`

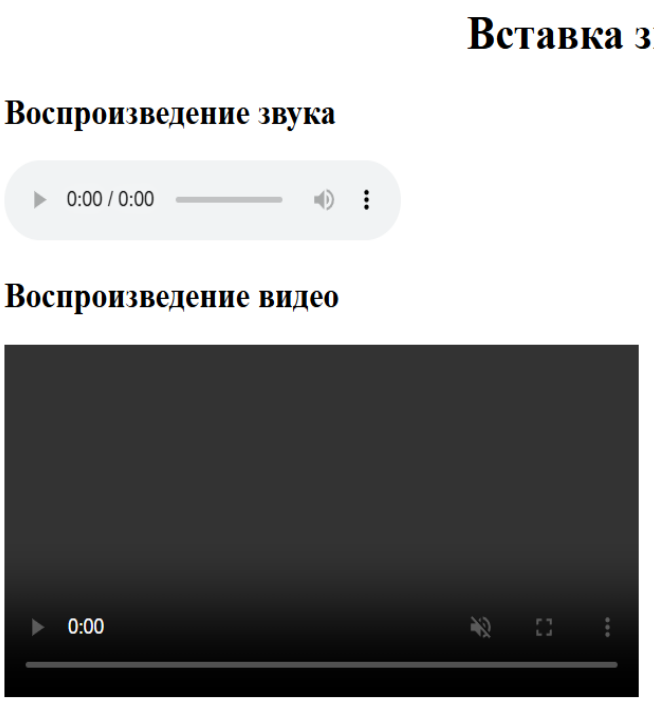


Рисунок 3 – Результат выполнения индивидуального задания 1

Контрольные вопросы

1. **Обзор HTML5-тегов для работы с медиа**
2. Назначение тегов <audio> и <video>
3. Атрибуты: controls, autoplay, loop, muted, poster, source
4. Создание HTML-документа с использованием медиафайлов
5. Создание структуры документа
6. Подключение локальных аудио- и видеофайлов
7. Организация управления воспроизведением
8. Устаревшие теги (<BGSOUND>, <IMG DYNSRC>)
9. Преимущества современных HTML5-тегов
10. Создание проекта и структуры папок
11. Использование fallback-сообщений (например: «Ваш браузер не поддерживает аудио...»)

Лабораторная работа №3

Тема: Списки

Цель: Изучение и закрепление знаний по тегам для формирования нумерованных и маркированных списков

Списки (list) в HTML5

Списки добавлены в HTML как удобный способ структурирования информации. Они позволяют автоматически упорядочивать элементы, будь то маркеры или номера. Пользователю не нужно заботиться о ручной нумерации — браузер делает это автоматически.

HTML поддерживает два основных типа списков:
- Маркированные (ненумерованные) списки — элементы отмечаются маркерами.
- Нумерованные списки — элементы автоматически нумеруются.

В HTML5 оформление списков рекомендуется выполнять с помощью CSS, а не устаревших атрибутов.

Теги для ненумерованных (маркированных) списков:

- (Unordered List) — начало маркированного списка
- — конец списка
- (List Item) — элемент списка

Современное оформление через CSS:

```
<ul style="list-style-type: square;">
  <li>Пункт 1</li>
  <li>Пункт 2</li>
</ul>
```

Вместо устаревших атрибутов type=DISC, CIRCLE, SQUARE рекомендуется использовать list-style-type:

Значение list-style-type	**Вид маркера**
disc	● (по умолчанию)
circle	○
square	■

Теги для нумерованных списков:

- (Ordered List) — начало нумерованного списка
- — конец списка
- — элемент списка

Примеры использования:

```
<ol type="I" start="3">
  <li>Третий</li>
  <li>Четвёртый</li>
  <li value="10">Десятый</li>
</ol>
```

Возможные значения атрибута type:

Значение type	**Последовательность**
1	1, 2, 3, 4, ...
i	i, ii, iii, ...
I	I, II, III, ...
a	a, b, c, ...
A	A, B, C, ...

Что больше не используется:

Атрибут / Тег	**Статус**	**Комментарий**
**type у **	Устарел	Используется list-style-type в CSS
SKIP	Не существует	Ошибочное или устаревшее описание
<img dynsrc> / <bgsound>	Устарели	Заменены на <audio> и <video>

Итог

HTML5 предоставляет мощные средства для создания списков, но современная практика требует:
- Использовать семантические теги <ul>, <ol>,
- Для внешнего вида применять CSS, а не устаревшие HTML-атрибуты
- При необходимости управлять нумерацией — использовать start, value и reversed

Индивидуальное задание 1

Действие:

1. Установите Visual Studio Code с сайта https://code.visualstudio.com, если он ещё не установлен.
2. Создайте новую папку, например: `spiski_html`.
3. Откройте созданную папку в Visual Studio Code: **Файл → Открыть папку...**
4. Создайте новый файл, например `Exampl3.html`.(*можно использовать и другое название файла*)
5. Сопоставить изображение в окне браузера и текст HTML- документа в Visual Studio Code, определить какие теги и атрибуты тегов позволяют создать маркированный, нумерованный и многоуровневый списки. Результат должен быть примерно такой, как на представленном ниже рисунке 4.

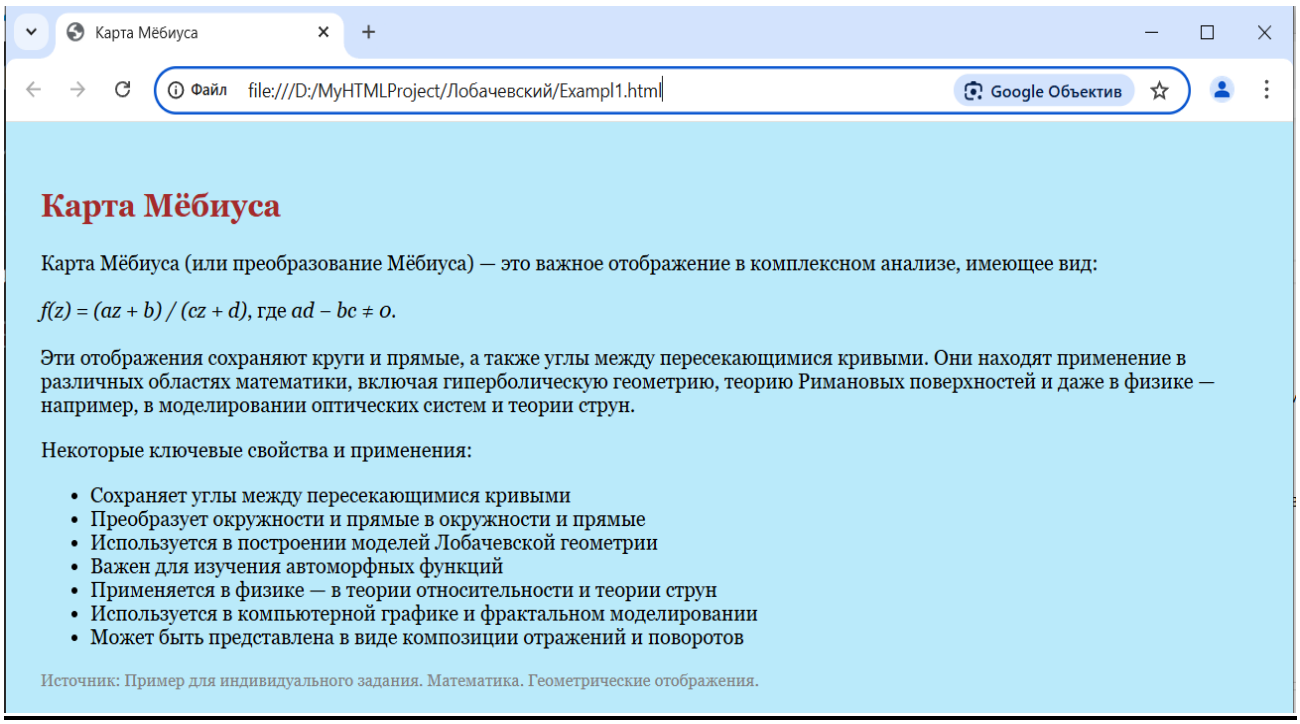


Рисунок 4 – Результат выполнения индивидуального задания 1

Контрольные вопросы

1. Что такое карта Мёбиуса и в каком разделе математики она используется?
2. Какие свойства имеет преобразование Мёбиуса?
3. Какие теги используются для форматирования заголовков в HTML?
4. Чем отличаются теги <ul> и ? В каких случаях применяются?
5. Как задать цвет фона страницы в HTML?
6. Для чего используется атрибут style в HTML-теге?
7. Какие теги используются для задания абзацев и строк формул?
8. Как отобразить математическое выражение, например: $f(z) = \frac{az+b}{cz+d}$, в HTML?
9. Как задать шрифт и его стиль (курсив, полужирный) через встроенные стили HTML?

Лабораторная работа №4

Тема: Таблицы

Цель: Изучение и закрепление знаний по тегам для формирования простых и сложных таблиц и научиться создавать HTML-страницы с помощью таблиц

Краткие теоретические сведения

Таблицы в HTML формируются нетрадиционным способом - построчно. Сначала с помощью элемента TR необходимо создать ряд таблицы, в который затем элементом TD помещаются ячейки.

TABLE - элемент для создания таблицы. Обязательно должен иметь начальный и конечный тэги.

Ячейки внутри таблицы создаются с помощью элементов TR, TD, TH и CAPTION. Параметры: align - определяет способ горизонтального выравнивания таблицы, возможные значения: left, center, right. Значение по умолчанию - left; valign - определяет способ вертикального выравнивания таблицы.

Возможные значения: top, bottom, middle; border - определяет ширину внешней рамки таблицы (в пикселях). При BORDER="0" или при отсутствии этого параметра рамка отображаться не будет; cellpadding - определяет расстояние (в пикселях) между рамкой каждой ячейки таблицы и содержащимся в ней материалом; cellspacing - определяет расстояние (в пикселях) между границами соседних ячеек; width - определяет ширину таблицы.

Ширина задается либо в пикселях, либо в процентном отношении к ширине окна браузера. По умолчанию этот параметр определяется автоматически в зависимости от объема содержащегося в таблице материала; height - определяет высоту таблицы. Высота задается либо в пикселях, либо в процентном отношении к высоте окна браузера. По умолчанию этот параметр определяется автоматически в зависимости от объема содержащегося в таблице материала; bgcolor - определяет цвет фона ячеек таблицы. Задается либо RGB-значением в шестнадцатеричной системе, либо одним из 16 базовых цветов; background - позволяет заполнить фон таблицы рисунком. В качестве значения необходимо указать URL рисунка.

CAPTION - задает заголовок таблицы. Содержание заголовка должно состоять только из текста. Использование блочных элементов в этом случае недопустимо. Атрибуты: **align** - определяет способ вертикального выравнивания заголовка таблицы. Возможные значения: **top** - помещает заголовок над таблицей (значение по умолчанию); **bottom** - помещает заголовок под таблицей.

TR - создает новый ряд (строку) ячеек таблицы. Ячейки в ряду создаются с помощью элементов TD и TH. Атрибуты: **align** - определяет способ горизонтального выравнивания содержимого всех ячеек данного ряда. Возможные значения: **left**, **center**, **right**; **valign** - определяет способ

вертикального выравнивания содержимого всех ячеек данного ряда. Возможные значения: top, bottom, middle; bgcolor - определяет цвет фона для всех ячеек данного ряда. Задается либо RGB-значением в шестнадцатеричной системе, либо одним из 16 базовых цветов.

TD/TH - элемент TD создает ячейку с данными в текущей строке. Элемент TH также создает ячейку, но определяет ее как ячейку-заголовок. Такое разграничение позволяет браузерам оформлять содержимое ячейки-заголовка и ячеек с данными разными шрифтами. В качестве содержимого ячейки можно использовать другие таблицы.

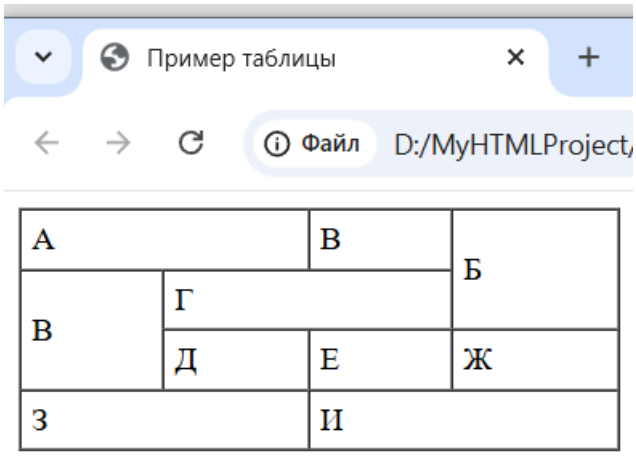
Атрибуты: **align** - определяет способ горизонтального выравнивания содержимого ячейки. Возможные значения: **left, center, right**.

Для TD по умолчанию выполняется выравнивание по левому краю, а для TH - центрирование; **valign** - определяет способ вертикального выравнивания содержимого ячейки. Возможные значения: **top, bottom, middle**. По умолчанию происходит выравнивание по центру, если значение этого параметра не было задано ранее в элементе TR; **width** - определяет ширину ячейки. Ширина задается в пикселях или в процентном отношении к ширине таблицы; **height** - определяет высоту ячейки. Высота задается в пикселях или в процентном отношении к ширине таблицы; **colspan** - определяет количество столбцов, на которые простирается данная ячейка. По умолчанию имеет значение 1; **rowspan** - определяет количество рядов, на которые простирается данная ячейка. По умолчанию имеет значение 1; **nowrap** - блокирует автоматический перенос слов в пределах текущей ячейки; **bgcolor** - определяет цвет фона ячейки. Задается либо RGB-значением в шестнадцатеричной системе; **background** - заполняет ячейку фоновым рисунком, при этом указать URL рисунка.

Атрибут	Назначение
`border`	Толщина рамки таблицы
`cellspacing`	Расстояние между ячейками
`cellpadding`	Отступ внутри ячейки
`width` / `height`	Ширина и высота таблицы
`bgcolor`	Цвет фона ячейки или строки
`colspan`	Объединение ячеек по горизонтали
`rowspan`	Объединение ячеек по вертикали
`align`	Выравнивание содержимого (left, center, right)
`valign`	Вертикальное выравнивание (top, middle, bottom)

Пример:

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>Пример таблицы</title>
</head>
<body>
    <table border="1" cellpadding="5" cellspacing="0" width="300">
        <tr>
            <td colspan="2">А</td>
            <td>В</td>
            <td rowspan="2">Б</td>
        </tr>
        <tr>
            <td rowspan="2">В</td>
            <td colspan="2">Г</td>
        </tr>
        <tr>
            <td>Д</td>
            <td>Е</td>
            <td>Ж</td>
        </tr>
        <tr>
            <td colspan="2">З</td>
            <td colspan="2">И</td>
        </tr>
    </table>
</body>
</html>
```



А		В	Б
В	Г		
	Д	Е	Ж
З		И	

Результат работы вышеприведенного примера.

Теги для формирования таблиц

<TABLE></TABLE> – начало и конец таблицы
<CAPTION></CAPTION> – начало заголовка и конец заголовка таблицы

заголовок располагается прямо по центру относительно ширины таблицы

<TH></TH> (**T**able **H**eader) – начало заголовков столбцов или строк таблицы и конец заголовков столбцов или строк таблицы

<TR></TR> (**T**able **R**ow) – начало строки таблицы и конец строки таблицы

<TD></TD> - начало ячейки таблицы и конец ячейки таблицы

Атрибуты тэга <TABLE>

- WIDTH=*ширина таблицы в пикселях или %*

- BORDER=*ширина границы таблицы в пикселях или %*

- CELLSPACING= *ширина промежутков между ячейками в пикселях или %*

-CELLPADDING= *ширина промежутков между содержимым ячейки и её границами в пикселях или %*

Например:

```
<TABLE BORDER=3 CELLSPACING=2 CELLPADDING=2 WIDTH="80%">
<CAPTION> ... заголовок таблицы ... </CAPTION>
<TR>
    <TD> первая ячейка таблицы </TD>
     <TD> вторая ячейка таблицы </TD
</TR> ...
    ...
  </TABLE>
```

Атрибуты тэга <TH>, <TR>, <TD>

Таблица может быть простой или сложной, когда несколько строк или столбцов объединяются

ROWSPAN = *количество объединяемых строк*

COLSPAN = *количество объединяемых столбцов*

BGCOLOR = *цвет фона заголовка, строки, ячейки*

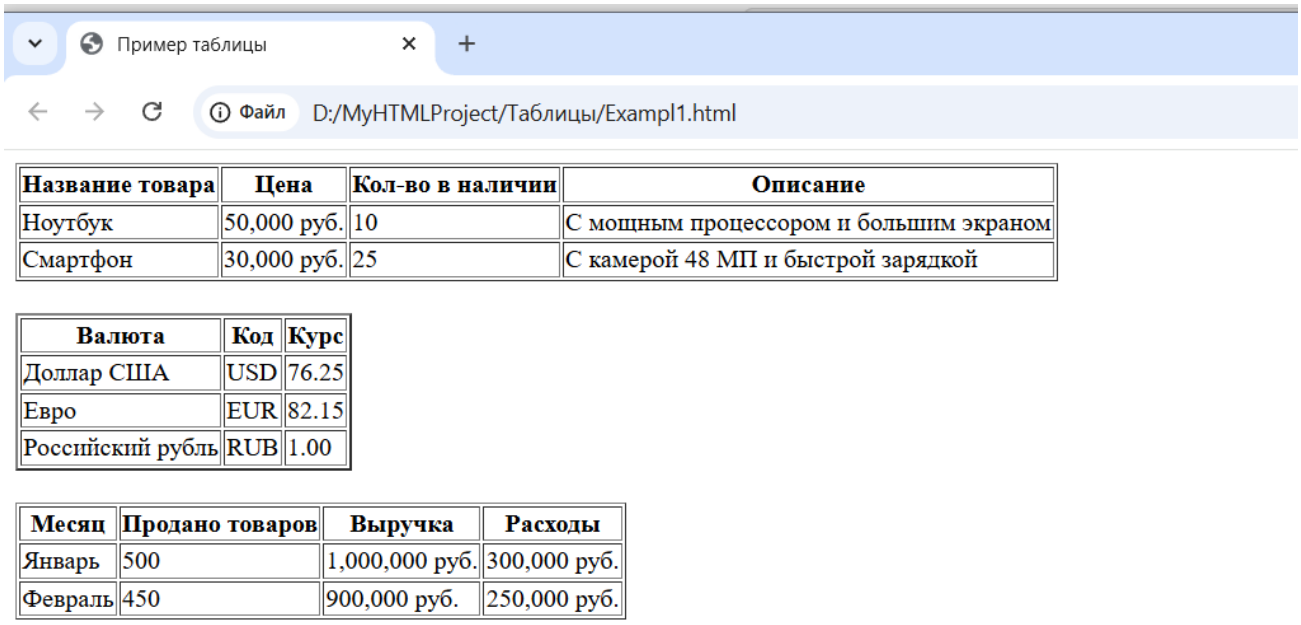
Пример сложной таблицы

```
<HTML>
<BODY background=Puzzle.jpg><br>
<br><br>
<TABLE border=5 cellspacing=3>
<caption><b><Font size=4 color=E88CB7>Заголовок таблицы</caption></b>
  <tr><th ROWSPAN=4 bgcolor=silver><font color=FBDC2F>Заголовок 1
 <tr><TD>Ячейка 1
   <TD>Ячейка 2
   <TD>Ячейка 3
<tr><TD>Ячейка 4
   <TD>Ячейка 5
   <TD>Ячейка 6
</TABLE>
</BODY>
</HTML>
```

Индивидуальное задание 1 (Таблицы простые и сложные)

Действие:

1. Установите Visual Studio Code с сайта https://code.visualstudio.com, если он ещё не установлен.
2. Создайте новую папку, например: `Tablici_html`.
3. Откройте созданную папку в Visual Studio Code: **Файл → Открыть папку...**
4. Создайте новый файл, например `Exampl4.html`.(*можно использовать и другое название файла*)
5. Сопоставить изображение в окне браузера и текст HTML- документа в Visual Studio Code, определить какие теги и атрибуты тегов позволяют создать таблицы. Результат должен быть примерно такой, как на представленном ниже рисунке 5.Ззадать цвет фона на свое усмотрение



Название товара	Цена	Кол-во в наличии	Описание
Ноутбук	50,000 руб.	10	С мощным процессором и большим экраном
Смартфон	30,000 руб.	25	С камерой 48 МП и быстрой зарядкой

Валюта	Код	Курс
Доллар США	USD	76.25
Евро	EUR	82.15
Российский рубль	RUB	1.00

Месяц	Продано товаров	Выручка	Расходы
Январь	500	1,000,000 руб.	300,000 руб.
Февраль	450	900,000 руб.	250,000 руб.

Рисунок 5 – Результаты выполнения индивидуального задания 1

Контрольные вопросы

1. Теги для формирования таблиц
2. Все атрибуты тега таблицы
3. Что такое сложная таблица
4. Что такое простая таблица
5. Как объединить строки и столбцы

Лабораторная работа №5

Тема: **Гиперссылки**

Цель: Изучение основных тегов для организации ссылок в пределах одного документа и ссылки на другие документы

Гиперссылки

Тело документа может содержать ссылки на другие документы, текст и другую форматированную информацию.

Для организации перехода внутри одного HTML–файла следует ввести необходимый текст в двух местах:

- записать метку в том месте HTML- файла, куда необходимо перейти.
- записать переход на эту метку в том месте, откуда будет выполнен переход.

Гипертекстовая ссылка состоит из двух частей: указателя и адреса (URL-адреса).

Ссылки на метки

Можно организовать ссылку на другую часть этого же документа, если предварительно расставить в начале таких частей - **метки**:

<**A HREF**= *#имя метки* >активный текст или изображение **</A>**

Создает ссылку на метку в текущем HTML-документе:

<**A NAME**= *имя метки* >начало текстового фрагмента **</A>**

Можно также создать ссылку на метку в другом документе, если в нем имеется метка:

<**A HREF**= *имя файла#имя метки* >активный текст или изображение **</A>**

<A NAME=”METKA”></A>, где METKA – это любое, состоящее из букв английского алфавита и цифр. Для перехода на эту метку запишите: *<A HREF= METKA> Перейти на метку </A>*.

Например: Требуется организовать переход с текущего места документа в начало HTML – файла. Переход осуществляется левой кнопки мыши на текст «Вернуться в начало». Для выполнения этого перехода задайте:

- в начале HTML – файла записать: *<A NAME=”A0”></A>*
- в необходимом для перехода месте HTML– файла записать:

<A HREF= HTML0 >Вернуться в начало раздела </A>.

Для перехода из любого места HTML–файла h1.htm в начало HTML – файла h2.htm, в файле h1.htm следует записать: *<A HREF= «h2.htm»> Переход к файлу h2.htm </A>*

Можно перейти на заданную метку METKA другого документа h2.htm с помощью следующей команды: *<A HREF= «h2.htm#METKA»> Переход в начало файла h2.htm </A>*

Для записи гипертекстовой ссылки в HTML – файл используйте следующую команду: *<A HREF= «URL»> Переход по гипертекстовой ссылке </A>*

При нажатии левой кнопкой мыши на текст перехода по гипертекстовой ссылке Вы перейдете на ресурс, заданный в URL. Для возврата используйте клавишу «Back» браузера.

Гипертекстовая ссылка с помощью изображения

Допустим, что Вы выводите на экран изображение IMAGE- файл, который называется, например, picture.gif. Вы хотите организовать страницу так, чтобы при нажатии левой кнопки мыши на изображение перейти на другую домашнюю страницу, например, на HTTP://WWW.HOME.COM, то следует записать следующую команду:

<A HREF= «HTTP://WWW.HOME.COM»><IMG SRC=HOME.gif>

Создание ссылки на почтовый адрес

Для организации ссылки в домашней страничке на почтовый адрес следует записать в конце первой домашней страницы HTML – файла:

<A HREF= mailto:ataev_ruslan@hotmail.com>Webmaster</A>

Формат тэга, создающего ссылку на другой документ (HTML-документ или файл с изображением, звуком, видео):

<A HREF= *имя файла. расширение*> активный текст или изображение **</A>**

Если файл находится в другой папке, то нужно указывать путь к нему.

Например:

<A HREF= "web/sprav.pps"> активный текст или изображение </A>

Если создается ссылка на ресурс Интернета, то нужно указать его URL:

<A HREF= *URL- адрес* >активный текст или изображение **</A>**

Например:

<A HREF=http:// upm.ipschool.spb.ru> активный текст или изображение </A>

Бегущая строка

<MARQUEE> – начало бегущей строки

</MARQUEE>– конец бегущей строки

Фрагмент текста, картинки или любые другие объекты между этими тэгами будут двигаться так, как определят атрибуты.

Атрибуты тэга <MARQUEE>

WIDTH = *ширина области бегущей строки в пикселях или в % от высоты экрана*

BGCOLOR = *название цвета или его код*

Определяет цвет фона бегущей строки.

DIRECTION = *LEFT*
RIGHT

Определяет направление бегущей строки

n

LOOP = *INFINITE*

Определяет количество "прокруток" бегущей строки: при первом значении атрибута – n, при втором значении атрибута – бесконечно (по умолчанию)

ALIGN = *TOP* / *MIDDLE* / *BOTTOM*

Позволяет разместить текст в верхней, средней или нижней части области бегущей строки.

BEHAVIOR = *SCROLL* / *SLADE* / *ALTERNATE*

Определяет режим вывода ("поведения") бегущей строки: прокрутка циклическая, выход и остановка и прыжки соответственно.

SCROLLAMOUNT = *скорость перемещения текста или графики в строке [1…3000]*

HIGHT=*высота бегущей строки (в пикселях или в % от высоты экрана*)

HSPACE= *ширина в пикселях левого и правого полей между областью бегущей строки и окружающим её текстом или графикой*

VSPACE= *размер отступа в пикселях сверху и снизу от бегущей строки до текста или графики*.

№	Тип ссылки или элемента	Описание	Пример HTML-кода
1	Ссылка внутри одного документа	Ссылка на часть текста внутри текущего HTML-документа (якорь)	`<a href="#section1">Перейти к разделу 1</a>, <a name="section1">Раздел 1</a>`
2	Ссылка на другой HTML-документ	Переход на другой HTML-файл	`<a href="page2.html">Открыть страницу 2</a>`
3	Ссылка на внешний сайт (URL)	Открытие внешнего ресурса в интернете	`<a href="https://example.com" target="_blank">Перейти на сайт</a>`
4	Ссылка по изображению	При клике на изображение — переход по ссылке	`<a href="https://example.com"><img src="img.png" alt="Кликни меня"></a>`
5	Ссылка на метку в другом документе	С переходом на определённый фрагмент другого HTML-документа	`<a href="page2.html#bottom">Перейти в конец страницы 2</a>`
6	Ссылка на почтовый адрес	Открывает почтовый клиент с указанным адресом	`<a href="mailto:name@example.com">Написать письмо</a>`
7	Бегущая строка	Двигающийся текст, используя тег `<marquee>`	`<marquee behavior="scroll" direction="left">Бегущая строка</marquee>`

Пример работы с сылками:

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
```

```
  <title>Примеры ссылок и бегущей строки</title>
</head>
<body>

  <h2>Ссылка с текста</h2>
  <p><a href="https://example.com">Перейти на внешний сайт</a></p>

  <h2>Ссылка с картинки</h2>
  <a href="https://example.com">
    <img src="example.jpg" alt="Пример изображения" width="200">
  </a>

  <h2>Ссылка на другой HTML-документ</h2>
  <p><a href="page2.html">Открыть вторую страницу</a></p>

  <h2>Ссылка на URL-адрес</h2>
  <p><a href="https://www.google.com" target="_blank">Открыть Google</a></p>

  <h2>Бегущая строка</h2>
  <marquee behavior="scroll" direction="left" scrollamount="5" bgcolor="#f0f0f0"
width="80%">
    Это пример бегущей строки. Добро пожаловать!
  </marquee>

</body>
</html>
```

Индивидуальное задание 1. Создание ссылки с текста, с картинки, на другой HTML- документ, на URL-адрес. Формирование бегущей строки

Действие:

1. Установите Visual Studio Code с сайта https://code.visualstudio.com, если он ещё не установлен.
2. Создайте новую папку, например: `Ssilcki_html`.
3. Откройте созданную папку в Visual Studio Code: **Файл → Открыть папку...**
4. Создайте новый файл, например `Exampl5.html`.(*можно использовать и другое название файла*)
5. Создать и сохранить в этой папке HTML-документ, используя выбранные файлы, так чтобы он содержал бегущую строку и ссылки на один из сайтов/ Организовать ссылки с одного HTML-документа на другой.
6.

Индивидуальное задание 2. Ссылки на метки

Действ:

1. Сопоставить изображение в окне браузера и текст HTML- документа В Visual Studio Code, определить какие теги и атрибуты тегов позволяют создавать ссылки на метки в документе.
2. Создать в Visual Studio Code и сохранить в этой папке HTML-документ, используя необходимые теги (расставить метки, создать ссылки на метки и предусмотреть ссылку возврата к началу документа).
1. Результат должен быть примерно такой, как на рисунке 6.

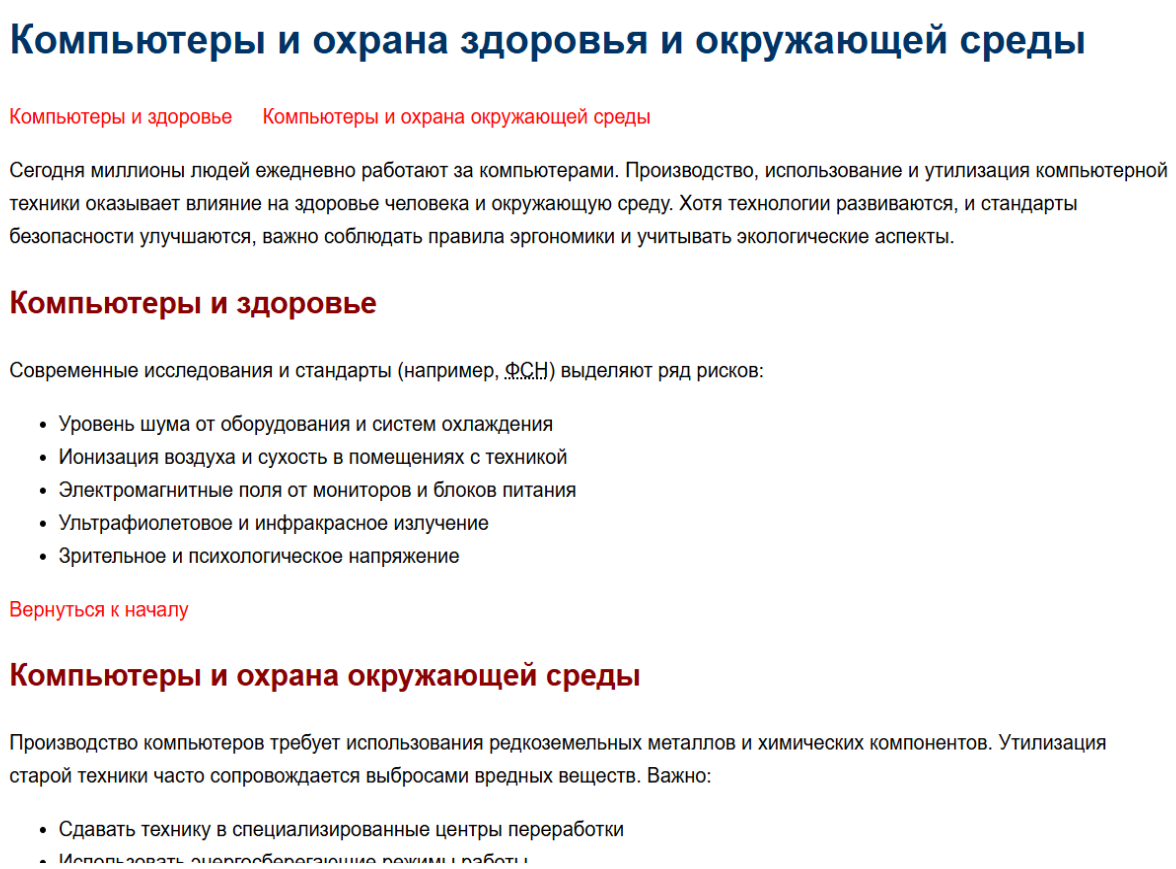


Рисунок 6 – Результаты выполнения индивидуального задания 2

Контрольные вопросы

1. Что такое гипертекст
2. Как осуществляется ссылка в пределах одного документа
3. Как организовать ссылку на другой документ
4. Как организовать ссылку на почтовый ящик
5. Что такое бегущая строка
6. Назовите тег бегущей строки и его атрибуты

Лабораторная работа № 6

Тема: Изучение фреймовой структуры HTML-страницы
Цель: Научиться создавать HTML-страницы с помощью фреймов

Краткие теоретические сведения

Фреймы

Кадры (**frames**) – это независимые части, на которые можно разбить окно браузера и в каждую часть можно загружать отдельную страницу. Эти страницы могут быть связаны между собой ссылками.

Количество частей (кадров) и их размеры, выбор горизонтального или вертикального деления, т.е. **кадровая (фреймовая) структура** задается тэгом:

<FRAMESET>… </FRAMESET>- Определяет расположение фреймов на странице.

Внимание! Использование тэга <FRAMESET> исключает использование тэга <BODY>! Следует запомнить, что не все браузеры отображают фреймы корректно!

Определяя структуру, указывают части сверху вниз и слева направо с помощью атрибутов.

Атрибуты тэга <FRAMESET>

ROWS = *ширина 1 части, ширина 2 части,… (в пикселях или %)*- определяет количество и размеры горизонтальных фреймов.

Например: <FRAMESET ROWS =100,300,100>;

или в процентах < FRAMESET ROWS =25%,35%,40%>

COLS = *ширина 1 части, ширина 2 части,… (в пикселях или %)*- определяет количество и размеры вертикальных фреймов.

Атрибуты тэга <FRAME>

<FRAME SRC=*имя файла.расширение*> - позволяет задать, какой html-документ, текстовый или графический, будет загружаться в отдельный кадр.

SRC - его значение есть имя документа, который будет в этот кадр загружаться.

NAME = *имя кадра*- задает имя кадра. Это имя потом используется в качестве значения атрибута **target** в тэге **<A HREF>** для того, чтобы документ, на который указывает ссылка, загружался в нужном кадре.

Например:

<FRAME SRC = my.htm NAME = *right*>

Если при этом в другом кадре загружен документ, содержащий тэг:

<A HREF = 1.jpg target= right >*загрузить рисунок* **</A>** тогда по щелчку на словах "загрузить рисунок" рисунок 1.jpg загрузится в кадр, в котором до этого был загружен документ my.htm.

Имена кадров могут начинаться с цифры или буквы. Для удобства можно указывать в качестве имени кадра слово, указывающее расположение кадра, например: **top, bottom, right, left.**

Еще **атрибуты** тэга <FRAME>:

scrolling - определяет наличие полей прокрутки фрейма, возможные значения: yes - отображать, no - не отображать, auto - отображать, если нужно.

Noresize - не позволяет мышкой изменять размеры (границы) между фреймами (кадрами)

frameborder - определяет наличие рамок у фрейма. Возможные значения: yes/no - отображать/ не отображать.

NOFRAMES - определяет, что будет отображать браузер, если он не поддерживает фреймы.

Таблица по тегам и атрибутам фреймов

Тег / Атрибут	Назначение	Пример использования
`<FRAMESET>`	Задает структуру фреймов (деление окна браузера)	`<FRAMESET ROWS="50%,50%">`
`ROWS` / `COLS`	Деление окна по горизонтали / вертикали	`ROWS="100,300", COLS="25%,75%"`
`<FRAME>`	Загружает HTML-документ в определённый кадр	`<FRAME SRC="menu.html" NAME="left">`
`SRC`	Путь к документу, который загрузится в фрейм	`SRC="page.html"`
`NAME`	Имя кадра, используется как target в ссылках	`NAME="main"`
`scrolling`	Полоса прокрутки: `yes`, `no`, `auto`	`scrolling="auto"`
`noresize`	Запрещает изменение размера фрейма	`noresize`
`frameborder`	Показать рамку: `1` (да) или `0` (нет)	`frameborder="0"`
`<NOFRAMES>`	Что показывать, если браузер не поддерживает фреймы	`<NOFRAMES>Ваш браузер не поддерживает фреймы</NOFRAMES>`
`<IFRAME>`	Вставка внутреннего окна в таблицу/документ	`<iframe src="info.html" width="400" height="300">`

Пример: это текст файла, задающего фреймы (frame1.html, frame2.html, frame3.html)

```
<FRAMESET FRAMEBORDER="0" FRAMESPACING="0" BORDER="0"
COLS="250,*">
<FRAME SRC="frame1.html" NAME="page">
<FRAMESET ROWS="150,*">
<FRAME SRC="frame2.html" NAME="menu1" MARGINWIDTH="0">
<FRAME SRC="frame3.html" NAME="menu2" MARGINWIDTH="0">
```

</FRAMESET>
<NOFRAMES>Ваш браузер не поддерживает фреймы</NOFRAMES>
</FRAMESET>
<BODY></BODY>

Фреймовую структуру также можно создать с помощью тэгов для создания таблиц и тэга **<IFRAME>** …**</IFRAME>**

Например:

<td>

<iframe width=600 height=330 align=center scrolling=none frameborder=0 name="frame" src=1.htm></iframe>

</td>

Это универсальный и поддерживаемый метод. Позволяет вставить одну страницу внутрь другой в любом месте страницы, например, в таблице.

Что делает данный способ:

- вставляет страницу 1.htm внутрь текущей страницы в виде встроенного окна шириной 600px и высотой 330px.
- name="frame" позволяет загружать в этот <iframe> другие страницы при помощи ссылок с target="frame" (как ты уже делал).
- Можно использовать внутри таблиц, , и комбинировать с CSS.

На рисунке 7 представлено окно, в котором фреймовая структура организована именно таким способом.

Соответствующий HTML-код:

```
<!DOCTYPE html>
<html>
<head>
  <title>Выдающиеся личности XX века</title>

</head>
<body link="brown" alink="gold" vlink="brown">
  <table cellpadding="0" cellspacing="0" width="700" height="400">
    <tr>
      <td colspan="2" height="30">
        <center><b><font size="5">Выдающиеся личности XX века</font></b></center>
      </td>
    </tr>

    <tr>
      <td width="300" height="30">
        <a href="einstein.html" target="frame"><font size="4">Альберт
Эйнштейн</font></a>
      </td>
      <td width="100" height="30">
        <img src="enshtein.webp" alt="Альберт Эйнштейн" class="center-image" />
```

```
      </td>
    </tr>

    <tr>
      <td width="300" height="30">
        <a href="curie.html" target="frame"><font size="4">Мария Кюри</font></a>
      </td>
      <td width="100" height="30">
        <img src="Curi.webp" alt="Мария Кюри" class="center-image" />
      </td>
    </tr>

    <tr>
      <td width="300" height="30">
        <a href="mandela.html" target="frame"><font size="4">Нельсон
Мандела</font></a>
      </td>
      <td width="100" height="30">
        <img src="nelsen.webp" alt="Нельсон Мандела" class="center-image" />
      </td>
    </tr>

    <tr>
      <td width="300" height="30">
        <a href="gagarin.html" target="frame"><font size="4">Юрий
Гагарин</font></a>
      </td>
      <td width="100" height="30">
        <img src="Gagarin.webp" alt="Юрий Гагарин" class="center-image" />
      </td>
    </tr>

    <tr>
      <td width="300" height="30">
        <a href="jobs.html" target="frame"><font size="4">Стив Джобс</font></a>
      </td>
      <td width="100" height="30">
        <img src="jobs.webp" alt="Стив Джобс" class="center-image" />
      </td>
    </tr>
  </table>
</body>
</html>
```

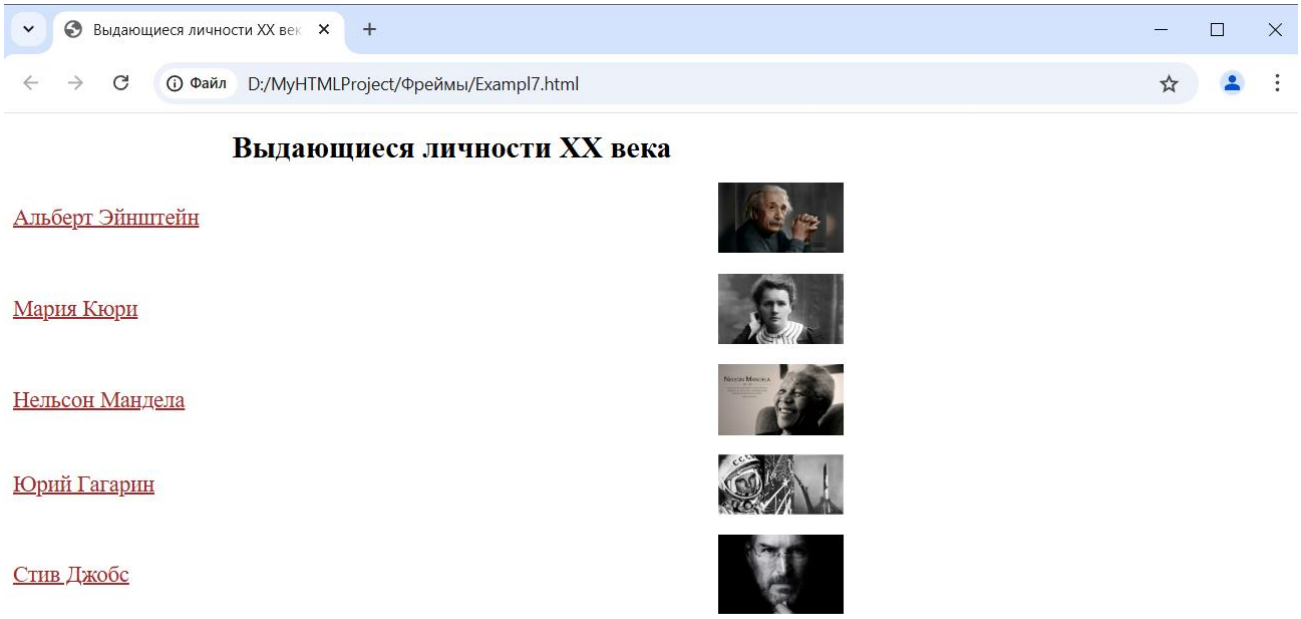


Рисунок 7 – Фреймовая структура, организованная с помощью таблиц и IFRAME

Индивидуальное задание 1. Создать страницу ссылок на ресурсы сети, обязательно с использованием фреймов. Разбить всю страницу на 4 фрейма: 2 по горизонтали и 2 по вертикали. В левом верхнем фрейме создать меню, состоящее из следующих ссылок:

- Поисковые системы;
- Каталоги;
- Литература.

В правом верхнем фрейме разместить какой-нибудь логотип, картинку.
В левом нижнем фрейме выводить ссылки из левого верхнего фрейма.

- Для поисковой системы – ссылки www.yandex.ru, www.google.ru, www.aport.ru, www.rambler.ru, www.filesearsh.ru оформить в виде ненумерованного списка;
- Для каталогов – ссылки www.list.ru, www.catalog.ru, www.rambler.ru, www.filesearsh.ru оформить в виде нумерованного списка;
- Для литературы – все ссылки оформить как списки с определениями, то есть разложить по рубрикам библиотека (www.lib.ru, www.aldebaran.ru), развлечение – игры, музыка, клипы и тд. (www.games.ru, www.partypoker.com.ru, www.mp3real.ru, www.zaycev.net, www.delit.net, www.mp3.uz), техническая литература (www.citforum.ru, www.intuit.ru, www.kurs-lab.ru).

Все поисковые системы выводить с их логотипом.

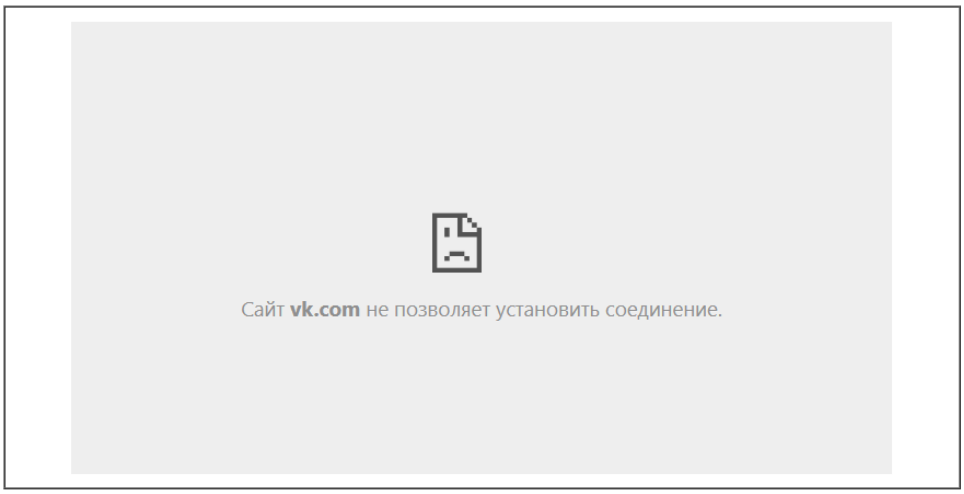
В правом нижнем фрейме отображать непосредственно содержание ссылок из левого нижнего фрейма, то есть загружать сами сайты.

Примечание:

Большинство крупных сайтов (такие как Google, Yandex, VK, Rambler и т.д.) запрещают загрузку себя в <iframe> из соображений безопасности. Это делается с помощью специальных HTTP-заголовков, например:

- X-Frame-Options: DENY
- Content-Security-Policy: frame-ancestors 'none'

Поэтому, при попытке открыть эти сайты во фрейме, можно получить ошибку «сайт не разрешает встраивание» или сайт **vk.com** не позволяет установить соединение.



Пример выполнения 1-го индивидуального задания: изучение возможностей языка HTML – фреймы и списки.

Как можно выполнить задание «на практике»:

Чтобы всё визуально работало, ты можешь:

1. Сделать страницы-заглушки вместо настоящих сайтов (например `google.html`, `yandex.html` и т.д.), в которых просто будет ссылка и логотип.
2. Подключить их во фреймах так, как описано в задании.
3. Оформить списки в нужных форматах: UL, OL, DL (по заданию).
4. Использовать картинки логотипов, сохранив их в ту же папку.

Структура файлов:

Создай папку и помести туда эти файлы:

- `index.html` — основная страница с фреймами
- `menu.html` — левый верхний фрейм (меню)
- `list.html` — левый нижний фрейм (ссылки, списки)
- `logo.html` — правый верхний фрейм (логотип)
- `content.html` — правый нижний фрейм (открывается при нажатии), картинки (логотипы) — по желанию
- `search.html`
- `literature.html`
- `catalogs.html`

HTML код:

index.html

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>Фреймы и списки</title>
</head>
<frameset cols="30%,*">
  <frameset rows="25%,*">
    <frame src="menu.html" name="menu">
    <frame src="list.html" name="list">
  </frameset>
  <frameset rows="25%,*">
    <frame src="google.gif" name="logo">

    <frame src="content.html" name="content">
  </frameset>
</frameset>
<noframes>
  Ваш браузер не поддерживает фреймы.
</noframes>
</html>
```

menu.html

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>Меню</title>
</head>
<body bgcolor="#d0f0f0">
  <ul>
    <li><a href="search.html" target="list">Поисковые системы</a></li>
    <li><a href="catalogs.html" target="list">Каталоги</a></li>
    <li><a href="literature.html" target="list">Литература</a></li>
  </ul>
</body>
</html>
```

list.html

```
<!DOCTYPE html>
<html lang="ru">
<head><meta charset="UTF-8"><title>Список</title></head>
<body bgcolor="#f0f0f0">
  <p>Выберите категорию в меню выше.</p>
</body>
```

```
</html>
```

logo.html

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>Логотип</title>
</head>
<body bgcolor="#e0f0ff" align="center">
  <img src="logo.png" alt="Логотип" width="150">
</body>
</html>
```

content.html

```
<!DOCTYPE html>
<html lang="ru">
<head><meta charset="UTF-8"><title>Контент</title></head>
<body bgcolor="#ffffff">
  <p>Здесь будет отображаться сайт по ссылке.</p>
</body>
</html>
```

search.html

```
<!DOCTYPE html>
<html lang="ru">
<head><meta charset="UTF-8"><title>Поисковые системы</title></head>
<body bgcolor="#ffffff">
  <ul>
    <li><img src="yandex.png" width="50"> <a href="https://yandex.ru"
target="content">Yandex</a></li>
    <li><img src="google.png" width="50"> <a href="https://google.com"
target="content">Google</a></li>
    <li><a href="https://aport.ru" target="content">Aport</a></li>
    <li><a href="https://rambler.ru" target="content">Rambler</a></li>
    <li><a href="https://filesearch.ru" target="content">Filesearch</a></li>
  </ul>
</body>
</html>
```

literature.html

```
<!DOCTYPE html>
<html lang="ru">
<head><meta charset="UTF-8"><title>Литература</title></head>
<body bgcolor="#ffffff">
  <dl>
    <dt><b>Библиотеки:</b></dt>
    <dd><a href="https://lib.ru" target="content">lib.ru</a></dd>
```

```
    <dd><a href="https://aldebaran.ru" target="content">aldebaran.ru</a></dd>

    <dt><b>Развлечения:</b></dt>
    <dd><a href="https://games.ru" target="content">games.ru</a></dd>
    <dd><a href="https://partypoker.com.ru"
target="content">partypoker.com.ru</a></dd>
    <dd><a href="https://mp3real.ru" target="content">mp3real.ru</a></dd>
    <dd><a href="https://zaycev.net" target="content">zaycev.net</a></dd>
    <dd><a href="https://delit.net" target="content">delit.net</a></dd>
    <dd><a href="https://mp3.uz" target="content">mp3.uz</a></dd>

    <dt><b>Техническая литература:</b></dt>
    <dd><a href="https://citforum.ru" target="content">citforum.ru</a></dd>
    <dd><a href="https://intuit.ru" target="content">intuit.ru</a></dd>
    <dd><a href="https://kurs-lab.ru" target="content">kurs-lab.ru</a></dd>
  </dl>
</body>
</html>
```

catalogs.html

```
<!DOCTYPE html>
<html lang="ru">
<head><meta charset="UTF-8"><title>Каталоги</title></head>
<body bgcolor="#ffffff">
  <ol>
    <li><a href="https://list.ru" target="content">List.ru</a></li>
    <li><a href="https://catalog.ru" target="content">Catalog.ru</a></li>
    <li><a href="https://rambler.ru" target="content">Rambler</a></li>
    <li><a href="https://filesearch.ru" target="content">Filesearch</a></li>
  </ol>
</body>
</html>
```

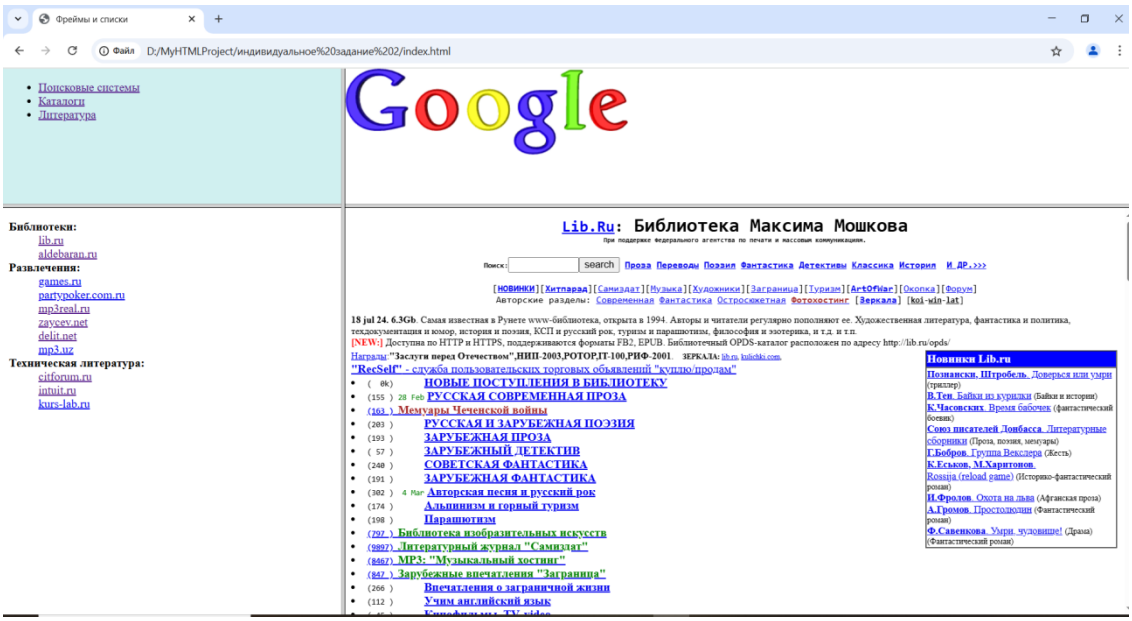


Рисунок 8 – Результат выполнения 2-го индивидуального задания

Индивидуальное задание 2. Фреймы, ссылки в заданный фрейм

Задание:

Создать HTML-страницу, которая разбита на четыре фрейма:

- Верхний левый — меню тем
- Верхний правый — изображение или логотип
- Нижний левый — ссылки на статьи по выбранной теме
- Нижний правый — отображение содержимого статьи

Математическая тема.

Меню (верхний левый фрейм):

Алгебра
Геометрия
Теория чисел

Ссылки по каждой теме (нижний левый фрейм):

Алгебра (Системы уравнений, Многочлены, Линейные функции)
Геометрия (Теорема Пифагора, Площади фигур, Окружность и круг
Теория чисел (Простые числа, Делители и кратные, Чётность)

Все статьи можно оформить в отдельных HTML-файлах, например: `algebra1.html`, `geometry2.html` и т.д.

Информационная тема .

Меню (верхний левый фрейм):

Языки программирования
Компьютерные сети
Искусственный интеллект

Ссылки (нижний левый фрейм):

Языки программирования (HTML, Python, C++)
Компьютерные сети
Что такое IP-адрес (TCP/IP модель, Протоколы передачи данных)
Искусственный интеллект (Машинное обучение, Нейронные сети, ChatGPT)

Каждый HTML-документ, на который делается ссылка, должен содержать текстовый фрагмент и картинку (для размещения текста и картинки использовать неявную таблицу и форматирование текста по ширине).

Контрольные вопросы

1. Что такое фрейм
2. Теги для организации фрейма
3. Объясните назначение каждого атрибута
4. Что такое index.html
5. В чем разница: FRAME, FRAMESET, IFRAME

Лабораторная работа № 7

Тема: Карты изображений в HTML-документах

Цель: Научиться создавать HTML-страницы с использованием карты изображений, организовать ссылки на различные элементы рисунка

Краткие теоретические сведения

Использование **карт изображений (image maps)** в HTML всё ещё работает и поддерживается браузерами, но **актуальность этого подхода снизилась** из-за появления более гибких и адаптивных технологий — таких как SVG, CSS и JavaScript. Однако, **для образовательных целей** и понимания работы с координатами на изображении это хороший опыт.

Карта изображения — это способ сделать отдельные части одного изображения кликабельными, каждая из которых ведёт на свою ссылку.

Минусы и почему редко используется:

Плохо адаптируется к мобильным экранам (размеры координат фиксированы)

Нет гибкого стилирования (в отличие от SVG или div-ов с CSS)

Трудно поддерживать и изменять

Могут быть проблемы с доступностью

Можно организовать ссылки с картинки на другие документы, но можно также организовать ссылки с разных частей картинки, если предварительно выделить эти части (области) и определить их координаты. Такую картинку называют **изображение-карта** (map).

<MAP>… </MAP> Этот тэг позволяет определить карту. Между открывающим и закрывающим тэгами <MAP> определяются области карты при помощи тэгов:

<AREA>

Для каждой области карты должен быть создан свой элемент <AREA>, который должен включать атрибут, определяющий ссылку HREF=*адрес ссылки*

Атрибуты тэга <MAP>

Задаётся имя изображения-карты при помощи атрибута NAME.

NAME=*имя*

Атрибуты тэга <AREA>

ALT=*название области изображения-карты*

Это атрибут для задания текста, заменяющего изображение-карту, не является обязательным.

TARGET=*имя фрейма*

Определяет имя фрейма, в котором будет отображаться документ, на который делается ссылка с данной области.

Атрибут, определяющий форму области на карте

SHAPE = *rect*
poly
circle
point

COORDS =*x,y,x1,y1*

Описывает координаты прямоугольной области *rect* (значениями атрибута будут координаты левого верхнего угла прямоугольника и правого нижнего, т.е. координаты противоположных его вершин).

COORDS =*x,y,x1,y1… xn,yn*

Описывает координаты многоугольной области *poly* (значениями атрибута будут пары координат всех вершин многоугольника).

COORDS =*x,y,r*

Описывает координаты области, имеющей вид окружности *circle* (значениями атрибута будут координаты центра окружности и значение радиуса).

COORDS =*x,y*

Описывает координаты области, имеющей вид точки (значениями атрибута будут её координаты).

Для того, чтобы записать координаты выделенной области в качестве значения атрибута COORDS,нужно их определить:

-загрузить картинку (которая станет изображением-картой) в графический редактор;

-выделить с помощью соответствующего инструмента область (прямоугольную, окружность, или область произвольной формы);

-в окне **Информация** нужно щёлкнуть в левом нижнем углу и выбрать Пиксели в качестве единицы измерения;

-выделив, например, прямоугольную область на рисунке и подведя указатель мыши к левому верхнему углу, в окне Информация рядом с обозначениями X и Y можно увидеть и записать координаты этой вершины.

-переместить указатель мыши в правый нижний угол и записать координаты противоположной вершины прямоугольника. Эти пары координат и будут значениями атрибута COORDS.

Атрибуты тэга <IMG> для изображения-карты

USEMAP = *#имя*

Определяет имя карты. Это имя должно совпадать с именем, которое указано в качестве значения атрибута NAME тэга <MAP>.

Пример карты изображения для карты мира

Допустим, мы хотим сделать кликабельными следующие регионы:

1. **Северная Америка (США и Канада)** — левый верх
2. **Африка** — центр
3. **Азия** — правый верх

```
<!DOCTYPE html>
<html lang="ru">
<head>
```

```
  <meta charset="UTF-8">
  <title>Карта изображения</title>
</head>
<body>
  <h1>Карта мира с активными зонами</h1>
  <p>Нажмите на континенты:</p>
  <img src="carta.jpg" usemap="#worldmap" alt="Карта мира" width="1029"
height="593">
  <map name="worldmap">
    <!-- Северная Америка -->
    <area shape="rect" coords="80,100,210,250"
          href="https://ru.wikipedia.org/wiki/Северная_Америка"
          alt="Северная Америка" title="Северная Америка">
    <!-- Африка -->
    <area shape="circle" coords="530,330,70"
          href="https://ru.wikipedia.org/wiki/Африка"
          alt="Африка" title="Африка">
    <!-- Азия -->
    <area shape="poly" coords="650,150,750,130,880,180,860,270,760,260"
          href="https://ru.wikipedia.org/wiki/Азия"
          alt="Азия" title="Азия">

  </map>

</body>
</html>
```

Результат работы данного скрипта представлен на рисунке 9

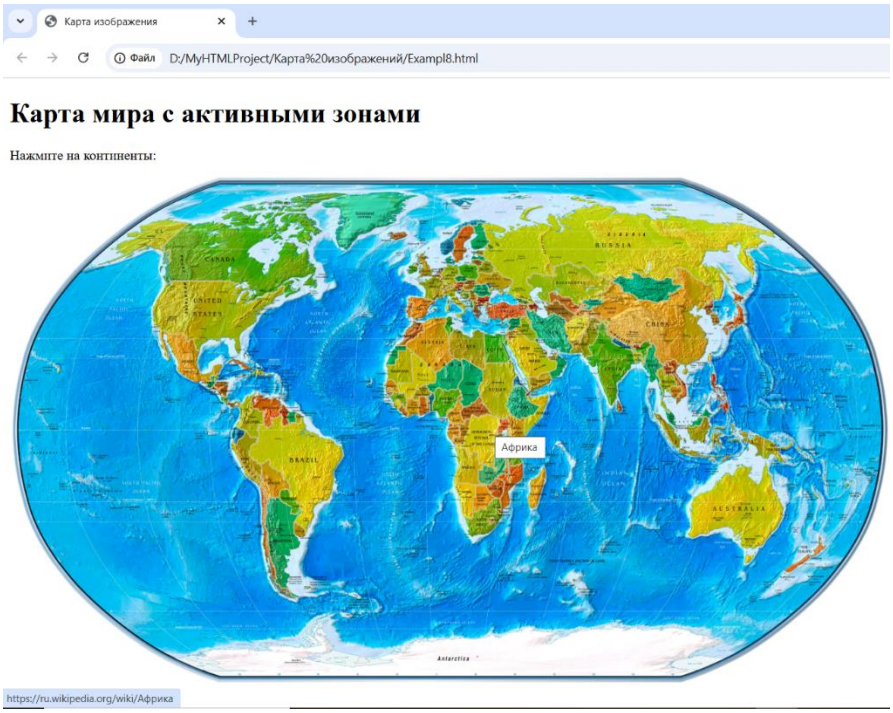


Рисунок 9 – Пример карты изображения для карты мира

Индивидуальное задание 1. Изображение-карта

В рамках данного задания вам необходимо создать веб-страницу, которая будет содержать несколько изображений с картами. Каждое изображение должно быть связано с интерактивными областями, которые приведут к различным ссылкам, например, к страницам с информацией о достопримечательностях, локациях или объектах на карте города.

Действие:

1. **Создайте HTML-документ** с несколькими изображениями. Каждое изображение должно быть связано с картой, использующей теги `<map>` и `<area>` для выделения интерактивных областей.
2. **Используйте изображения с картами** (например, карту города, карту отеля, план парка или любой другой объект). Для каждого изображения создайте соответствующие области, используя формы `rect` (прямоугольник), `circle` (круг) и `poly` (многоугольник).
3. **Ссылки на объекты**. Для карты города: создайте области для различных достопримечательностей (например, музей, парк, площадь). Для карты отеля: добавьте области для разных зон отеля (например, лобби, бассейн, ресторан). Для карты парка: создайте области для различных маршрутов, входов или зон отдыха.
4. **Сделайте изображения адаптивными** с использованием атрибутов `width` и `height` в процентах. Результат должен быть примерно такой, как на рисунке

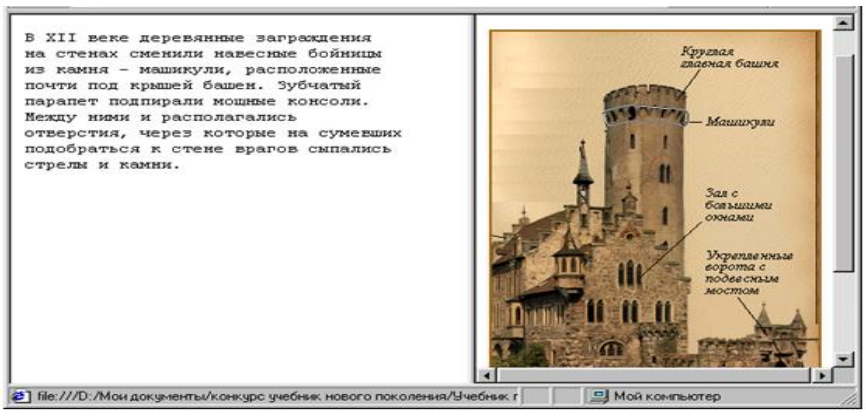


Рисунок 10 – Результат выполнения индивидуального задания

Контрольные вопросы:

1. Перечислить все теги для изображения–карты и действия их атрибутов
2. Что означает TARGET="OKNO"
3. Что определяет тег **USEMAP** = *#имя*
4. Объясните назначение каждого атрибута

Лабораторная работа № 8

Тема: Изучение тегов для организации форм

Цель: Научиться создавать интерактивные HTML-формы с использованием современных атрибутов и элементов HTML5

Краткие теоретические сведения

ФОРМЫ

Формы - это один из важнейших инструментов для взаимодействия пользователя с веб-страницей. Они необходимы для создания интерактивных веб-приложений, таких как регистрации, авторизации, опросы, сбор данных и многое другое. Знание принципов работы с формами в HTML является основой для разработки таких интерфейсов.

Форма – это набор таких знакомых нам по диалоговым окнам элементов, как поля ввода, поля выбора, переключатели. Форма – это то, что позволяет создавать интерактивные страницы, т.е. организовывать диалог с пользователем, создавать заполняемые анкеты, опросники и т.д.

Таким образом, HTML-форма — это элемент веб-страницы, позволяющий пользователю вводить данные и отправлять их на сервер. Формы необходимы для регистрации, авторизации, поиска, загрузки файлов, опросов и других взаимодействий с сайтом.

Основной тег формы:

```
<form action="URL" method="post">
  <!-- элементы формы -->
</form>
```

Атрибут	Описание
`action`	Указывает URL-адрес, на который будут отправлены данные формы
`method`	Метод передачи данных: `GET` (в URL) или `POST` (в теле запроса)
`enctype`	Способ кодирования данных (для `POST`):

<FORM>…</FORM> Этот тэг используется для создания заполняемых форм. Между <FORM> и </FORM> можно использовать тэги для создания полей нескольких типов: текстовые поля в одну или несколько строк, группы радиокнопок - их еще называют переключателями, окошки, в которые можно устанавливать флажки, меню, командные кнопки (см. рисунок 11).

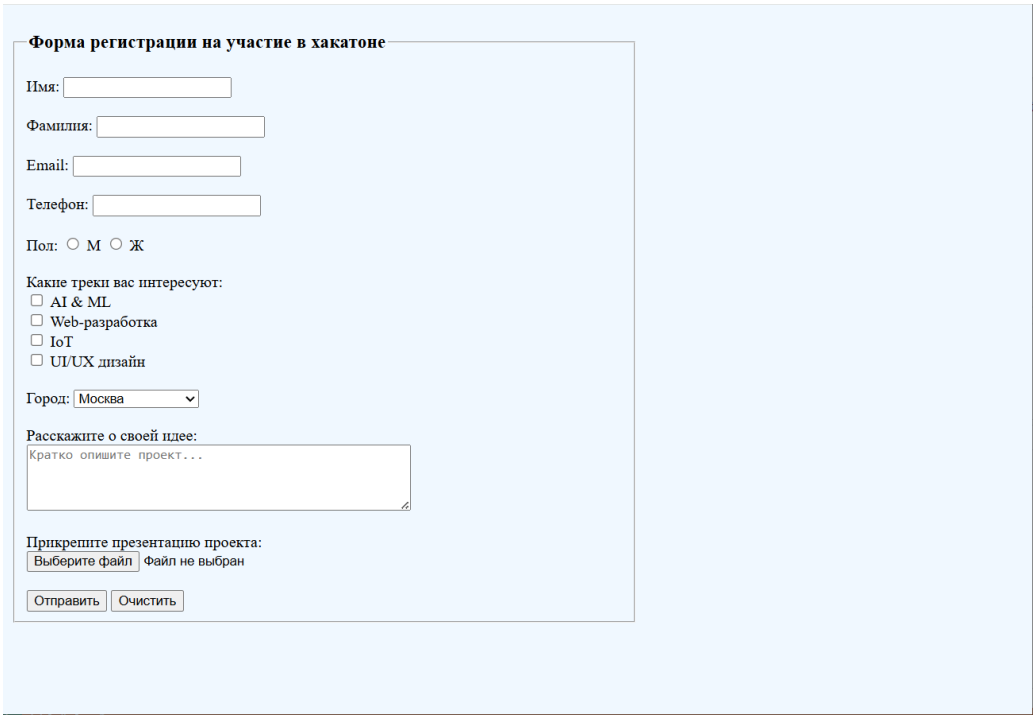


Рисунок 11 – Пример формы, содержащей разные поля

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Регистрация на хакатон</title>
</head>
<body bgcolor="#f0f8ff">
  <form action="https://example.com/submit" method="post"
enctype="multipart/form-data">
    <fieldset style="width: 600px;">
      <legend><h3>Форма регистрации на участие в хакатоне</h3></legend>

      Имя: <input type="text" name="first_name"><br><br>
      Фамилия: <input type="text" name="last_name"><br><br>
      Email: <input type="email" name="email"><br><br>
      Телефон: <input type="tel" name="phone"><br><br>

      Пол:
      <input type="radio" name="gender" value="m"> М
      <input type="radio" name="gender" value="f"> Ж<br><br>

      Какие треки вас интересуют:<br>
      <input type="checkbox" name="track1" value="ai"> AI & ML<br>
      <input type="checkbox" name="track2" value="web"> Web-разработка<br>
      <input type="checkbox" name="track3" value="iot"> IoT<br>
      <input type="checkbox" name="track4" value="design"> UI/UX дизайн<br><br>

      Город:
      <select name="city">
```

```
        <option value="moscow">Москва</option>
        <option value="spb">Санкт-Петербург</option>
        <option value="kazan">Казань</option>
        <option value="ekb">Екатеринбург</option>
      </select><br><br>

      Расскажите о своей идее:<br>
      <textarea name="idea" rows="4" cols="50" placeholder="Кратко опишите
проект..."></textarea><br><br>

      Прикрепите презентацию проекта:<br>
      <input type="file" name="file" accept=".pdf,.pptx"><br><br>

      <input type="submit" value="Отправить">
      <input type="reset" value="Очистить">
    </fieldset>
  </form>
</body>
</html>
```

Атрибуты тэга <FORM>

ACTION=*адрес*- обязательный атрибут, он определяет URL, по которому будет отправлено содержимое формы. Это может быть либо адрес электронной почты, либо путь к скрипту сервера, обслуживающему данную форму. *Например,* ACTION =mailto:имя@сервер.домен либо для запуска на сервере посредством протокола HTTP специальной программы, обслуживающей данную форму. *Например:* ACTION ="http://www.prim.com/cgi- bin/reg.pl"

METHOD- определяет, какой из HTTP методов будет использоваться для пересылки серверу (адрес которого задан предыдущим атрибутом) содержимого текущей формы. Это может быть либо GET, либо POST (по умолчанию используется GET).

ENCTYPE - этот атрибут определяет механизм, который следует использовать при кодировании содержания данной формы. По умолчанию используется application/x-www-form-urlencoded.

TYPE=*password* - в данном случае вводимые символы отображаются на экране в виде условных значков, таких как *. Делается это, чтобы скрыть текст от глаз при вводе паролей с клавиатуры.

TYPE=*checkbox* -этот тип используется для ввода в заполняемую форму простых значений булевого типа ("да"/"нет"). *Например*:

<input type=checkbox name=section value=yes>1
<input type=checkbox name=section value=yes> 2
<input type=checkbox name=section value=yes> 3
<input type=checkbox name=section value=yes> 4

TYPE=*radio* - используется для ввода в форму параметра, являющегося результатом однозначного выбора из набора альтернативных

вариантов. Этим вариантам при разметке ставится в соответствие группа "радиокнопок", в каждую из которых должен быть записан один и тот же атрибут NAME. В радиокнопках обязательно следует указывать также и атрибут VALUE. При заполнении формы среди группы радиокнопок только активированная генерирует пару "название/значение" в соответствующем поле данных. В каждой группе радиокнопок одна должна быть изначально активирована посредством атрибута checked.

Например, альтернативные варианты:

<INPUT type="radio" name=answer value=Yes> Да
<INPUT type="radio" name=answer value=No> Нет

TYPE=*image* - этот атрибут создает графический вариант для кнопок, инициирующих передачу данных.

URL для соответствующего изображения задается атрибутом src. Выравнивание картинки осуществляется согласно значению атрибута align. В этом отношении графические изображения кнопок подобны элементам IMG, их можно точно так же выравнивать по правому, верхнему, нижнему краю, либо ставить их по центру.

Например, <INPUT align="middle" name=Ok type=image src=gradient.jpg >

TYPE=*reset* - этот атрибут позволяет создать кнопку, на которой можно щелкнуть, чтобы вернуть все поля формы в исходное состояние (как было в момент загрузки документа). Информация о кнопке перезагрузки никогда не включаются в набор сведений, пересылаемых на сервер после заполнения формы. *Например,* <input type=reset>

TYPE=*file* - дает пользователям возможность дополнить содержимое текущей формы файлом. При разметке этого элемента обычно создается поле для ввода текста, к которому прилагается кнопка. Щелчок на этой кнопке приводит к раскрытию нового окошка, где можно просмотреть имеющиеся файлы, и выбрать один из них. Имя файла можно также ввести непосредственно в исходном текстовом поле. Точно так же, как и в случае type=text, можно использовать здесь атрибут size, чтобы выбрать ширину данного поля формы (единицей измерения здесь служит средняя ширина символов). Можно также установить верхний предел для длины вводимого имени файла посредством атрибута maxlength.

Например,<input type=file name=photo size=40>

Создание в формах меню

<SELECT>...</SELECT>
Тэг <SELECT создает в заполняемой форме меню типа "выбор одного пункта из многих", либо "несколько пунктов из многих". Между открывающим и закрывающим тэгами SELECT должен быть один или несколько элементов OPTION, описывающих отдельные пункты меню. Меню типа "один из многих" обычно реализуется как выпадающее меню, в то время как меню типа "несколько из многих" обычно предстает в виде списка с окошками, в которые можно устанавливать флажки против каждого пункта.

Атрибуты тэга <SELECT>

NAME=*имя* - Сообщает название для данного качества, которое затем будет использоваться во время передачи данных на сервер, чтобы указать, какие пункты в меню выбраны. Каждому пункту меню соответствует пара значений "название/величина".

SIZE=*количество* - В меню типа "несколько из многих" устанавливает количество одновременно видимых пунктов.

MULTIPLE- Этот атрибут указывает, что в данном меню пользователи могут сразу выбрать несколько пунктов. По умолчанию - только один пункт.

Атрибуты тэга <OPTION>

SELECTED- Если для тэга <OPTION> указан атрибут selected, то соответствующий этому элементу пункт меню уже при загрузке документа изначально помечается как выбранный. Однако если в меню типа "один из нескольких" изначально таким образом помечено более одного пункта, то это будет ошибкой.

VALUE=*значение* - Задает значение, которое соответствует данному пункту меню. В последнем случае это значение будет объединено с названием

Например:

```
<SELECT NAME=town>
 <OPTION VALUE=a> Москва
 <OPTION VALUE=b> Санкт-Петербург
 <OPTION VALUE=c> Архангельск
<OPTION VALUE=d>Астрахань
 </SELECT>
```

Создание в формах текстовых полей

<TEXTAREA>…</TEXTAREA> Задает поля для ввода нескольких строк текста Обычно это поле содержит текст инициализации, который при загрузке документа изначально будет записываться в данное поле.

Атрибуты тэга <TEXTAREA>

NAME=*имя* - Определяет название, которое будет использовано для идентификации данного поля textarea при предоставлении заполненной формы на сервер.

ROWS=*количество*- Задает количество строк текста, видимых на экране.
COLS=*количество*- Определяет ширину создаваемого текстового поля (единицей измерения служит средняя ширина символов).
Например: <TEXTAREA NAME=address ROWS=4 COLS=40>Я считаю, что… </TEXTAREA>

В таблице 1 собраны и представлены все теги для создания форм

Таблица 1 –Теги создания форм

<form></form>	Создает формы
<select multiple name="NAME" size=?> </select>	Создает скролируемое меню. Size устанавливает кол-во пунктов меню, которое будет показано на экране, остальные будут доступны при использовании прокрутки.
<option>	Указывает каждый отдельный элемент меню
<select name="NAME"></select>	Создает ниспадающее меню
<option>	Указывает каждый отдельный элемент меню
<textarea name="NAME" cols=40 rows=8></textarea>	Создает окно для ввода текста. Columns указывает ширину окна; rows указывает его высоту.
<input type="checkbox "name="NAME">	Создает checkbox. За тегом следует текст.
<input type="radio" name="NAME" value="x">	Создает radio кнопку. За тегом следует текст.
<input type=text name="foo" size=20>	Создает строку для ввода текста. Параметром Size указывается длина в символах.
<input type="submit" value="NAME">	Создает кнопку "Принять"
<input type="image" border=0 name="NAME" src="name.gif">	Создает кнопку "Принять" - для этого используется изображение
<input type="reset">	Создает кнопку "Отмена"

Дополнительные полезные атрибуты в HTML5

Атрибут	Применение	Пример
`required`	Обязательное поле	`<input type="text" required>`
`placeholder`	Текст-подсказка	`<input type="email" placeholder="example@mail.com">`
`pattern`	Валидация по шаблону (RegExp)	`<input pattern="[0-9]{3}">`
`accept`	Разрешённые	`<input type="file"`

	типы файлов	`accept=".jpg,.png">`
`min/max`	Ограничения значений для `number`, `date` и т.п.	`<input type="number" min="1" max="10">`

Индивидуальное задание

Создать страницу в Visual Code для загрузки формы резюме. Форма загрузки должна иметь следующие поля: имя, Email, загрузка файла формата `.pdf`, комментарий (textarea), кнопка "Отправить. Результат должен соответствовать рисунку 12.

Рисунок 12 – Форма загрузки резюме

Пояснение:

1. `enctype="multipart/form-data"` — обязательно для загрузки файлов
2. `accept=".pdf"` — ограничивает выбор файла только PDF-документами
3. `required` — обязательное заполнение полей "Имя", "Email", "Резюме"
4. Использованы теги: `<input>`, `<textarea>`, `<form>`, `<label>`

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
```

```
  <title>Форма загрузки резюме</title>
</head>
<body  bgcolor="#b4f8ff">
  <h2>Отправьте нам своё резюме</h2>

  <form action="/submit_resume" method="post" enctype="multipart/form-data">
    <!-- Имя -->
    <label for="name">Имя:</label><br>
    <input type="text" id="name" name="name" required><br><br>

    <!-- Email -->
    <label for="email">Email:</label><br>
    <input type="email" id="email" name="email" required><br><br>

    <!-- Загрузка файла -->
    <label for="resume">Прикрепите файл резюме (PDF):</label><br>
    <input type="file" id="resume" name="resume" accept=".pdf" required><br><br>

    <!-- Комментарий -->
    <label for="message">Комментарий:</label><br>
    <textarea id="message" name="message" rows="5" cols="40"
placeholder="Дополнительная информация..."></textarea><br><br>

    <!-- Кнопка отправки -->
    <input type="submit" value="Отправить">
  </form>
</body>
</html>
```

Варианты индивидуальных заданий:

1.Создайте форму регистрации пользователя, включающую следующие поля: Имя (обязательное), Email (с валидацией), пароль (тип `password`), кнопка отправки

2.Сверстайте форму обратной связи, включающую: тему обращения (выпадающий список), сообщение (многострочное поле), флажок согласия на обработку данных, кнопку "Отправить"

3.Сделайте форму для заказа товара, включающую: поле для имени и телефона, список с выбором товара, чекбоксы для дополнительных опций, кнопку "Оформить заказ"

4.Создайте форму опроса с радиокнопками на тему: "Какой браузер вы используете чаще всего?" Варианты: Chrome, Firefox, Safari, Edge, Другой.

5.Разработайте форму загрузки резюме: имя, Email, телефон для связи домашний и рабочий, должность, загрузка файла `.pdf`, комментарии (textarea), кнопка "Отправить"

6.Создайте форму бронирования столика в кафе: имя, количество гостей (input type="number"), дата и время (input type="datetime-local"), выбор зала (select), кнопка "Забронировать"

7.Форма входа в аккаунт: Email, пароль, чекбокс "Запомнить меня", кнопки "Войти" и "Сбросить"

8.Сделайте анкету студента: ФИО, пол (радиокнопки), группа (выпадающий список), хобби (checkbox), сообщение о себе, кнопка отправки

9.Создайте форму подписки на рассылку: Email, выбор тем рассылки (checkbox), частота писем (radio), кнопка "Подписаться"

10.Форма оценки сайта:оценка по 5-балльной шкале (radio), комментарий, имя (необязательное), кнопка "Оценить"

Контрольные вопросы:

1. Что такое Форма
2. Перечислите все теги для создания формы
3. Перечислите название атрибутов формы и их действия
4. Как работает тег SELECT
5. Как работает тег TEXTAREA

Лабораторная работа № 9

Тема: Каскадные таблицы стилей CSS. Шрифты и цветовая гамма.

Цель: Научить применять свойства шрифта и использовать цветовую гамму с помощью CSS.

Краткие теоретические сведения

Применение CSS

Каскадные таблицы стилей или CSS (Cascading Style Sheets) были революцией для WWW. Если до этого Web-дизайнер не знал, как будет выглядеть его творение в разных программах Web-обозревателей, то теперь он может контролировать все: от начертания шрифта до положения картинки на странице.

Предположим, нужно изменить цвет текста в HTML-документе с черного на синий. Для этого помещаете текст в пару тегов <FONT> и </FONT> следующего вида: <P><FONT color=blue >Это синий текст</FONT></P>.

А теперь представим, что вы внесли определение внешнего вида текста в другое место документа: P.bluetext {color: blue }

Эта строка обозначает, что мы определили для текста, находящегося внутри тега <P> и помеченного стилевым классом bluetext, синий цвет шрифта. Такая конструкция HTML **называется определением стиля или просто стилем.**

В результате в HTML-тексте у нас останутся только теги логического форматирования текста:

<P class="bluetext">Это синий текст</P>

Здесь мы пометили нужный текст с помощью атрибута **class**, присвоив ему значение bluetext. Атрибут **class** задает имя стилевого класса для тега, и его поддерживают все теги.

Можно переназначить цвета текста для всех тегов <P>. В этом случае не задаем имя стилевого класса: P {color: blue}.

Или можно задать форматирование для стилевого класса, не привязанного ни к какому тегу: **.bluetext {color: blue }**

И теперь можно присваивать стилевой класс тексту, заключенному в любые теги:

<H1 class="bluetext">Это синий цвет</H1>

<CENTER class="bluetext">Это синий цвет</CENTER>

Это <B class="bluetext">жирный синий</B> текст

Можно дать специальное форматирование тегу только в том случае, если он заключен внутрь другого тега:

H7 B {color: blue}

И теперь:

<H7><B>Этот</B> текст будет синим</H7>

<P>А <B>этот</B> - не будет!</P>

Более того, можно встроить определение стиля прямо в тег:

<P style="color: blue">Это синий текст</P>

Это достигается при помощи атрибута **style**, который также поддерживают все теги HTML.

И еще один способ привязать стиль к какому-либо тегу - использовать атрибут **id**, задающий уникальное имя элемента HTML.

#header of document {font-size: 20pt }

Здесь задан размер шрифта 20 пунктов.

<H1 id="header of document">Это заголовок документа</H1>

Можно задавать несколько атрибутов в определении стиля. В этом случае они разделяются точкой с запятой:

P {color: blue; font-size: 9ptl; text-align: center }

Определение стилей, вынесенные в заголовок HTML-документа, составляют таблицу стилей. Таблица стилей заключается в теги <STYLE> и </STYLE>:

<Style [type="text/css"]>

. . .

</STYLE>

Тег <STYLE> может содержать необязательный атрибут **type**, содержащий обязательное значение **text/css.**

Таблицу стилей можно вынести в отдельный файл и использовать сразу в нескольких документах. В этом случае в заголовке HTML-документа необходимо разместить тег <**LINK**>, указывающий на эту таблицу стилей:

<LINK rel="stylesheet" href="{Адрес файла таблицы стилей}">

Свойства шрифта

font - задает параметры шрифта элемента страницы.

Заменяет атрибуты font-family, font-height, font-size, font-style, font-variant и font-weight. Значения этих атрибутов могут располагаться в любом порядке.

font: {font-family} [{font-height}] [{font-size}] [{font-style}] [{font-variant}] [{font-weight}];

Значение по умолчанию - normal "Times New Roman".

Альтернативный формат:

font: caption|icon|menu|message-box|small-caption|status-bar;

В этом случае доступны шесть предопределенных значений, задающие один из стандартных шрифтов, используемых в элементах интерфейса Windows:

font: caption; - шрифт заголовка кнопок, текстовых меток и т.п.;

font: icon; - шрифт подписей под пиктограммами;

font: menu; - шрифт пунктов меню;

font: message-box; - шрифт содержимого стандартных окон-предупреждений;

font: small-caption; - мелкий шрифт заголовков;

font: status-bar; - шрифт содержимого строки состояния.

Поддерживается IE начиная с 4.0

font-family - указывает имя или семейство шрифта.

font-family: {Имя шрифта}|serif|san-serif|fantasy|monospace;

В качестве значения этого атрибута задается либо непосредственно имя нужного шрифта, либо одно из пяти предопределенных значений, задающих имя шрифтового семейства. Можно задавать одновременно несколько шрифтов, разделив их имена запятыми; в этом случае Web-обозреватель сможет выбрать из них тот, который установлен на компьютере клиента. Если имя шрифта содержит пробелы, его следует взять в кавычки.

font-family: "Times New Roman",sans-serif;

Поддерживается IE.

font-weight - задает "жирность" шрифта.

font-weight: normal|bold|bolder|lighter|100..900;

"Жирность" может быть задана следующими способами. font-weight: normal; - обычный;

font-weight: lighter; - светлее;

font-weight: bold; - жирный;

font-weight: bolder; - жирнее;

font-weight: от 100 до 900 - любое значение, кратное 100

Значение по умолчанию normal.

Поддерживается IE начиная.

font-size - задает размер шрифта.

Возможно задание либо абсолютного размера шрифта в одной из поддерживаемых CSS единиц измерения, либо как процент от размера шрифта родителя. Также доступны девять определенных значений.

font-size: 200% - относительная величина (проценты)

font-size: 150px - размер в пикселях

font-size: 300pt - размер в пунктах

font-size: {xx-small,small,medium,large,x-large,xx-large} - задают один из семи размеров шрифтов, поддерживаемых HTML

font-style - задает начертание шрифта.

font-style: normal|italic|oblique;

font-style: normal; - задает обычный вид шрифта (используется по умолчанию);

font-style: italic - курсивное начертание;

Цветовая гамма

color - Определяет цвет элемента.

color: {Цвет};

Поддерживается IE.

background - задает все свойства фона элемента страницы в один прием. Заменяет собой атрибуты background-attachment, background-color, background-image, background-position и background-repeat.

background: [{background-color}] [{background-image}] [{background-repeat}] [{background-attachment}] [{background-position}];

Значения этих свойств могут располагаться в любом порядке.

Поддерживается IE.

background-color - задает фоновый цвет Web-страницы или ее элемента.

background-color: {Цвет}|transparent;

scrollbar-track-color - задает цвет рабочей части полосы прокрутки, т.е. той ее части, по которой перемещается бегунок.

scrollbar-track-color: {Цвет};

Индивидуальное задание 1. Создать HTML-страницу с использованием CSS.

Определить для тега <P> синий цвет шрифта, семейство шрифта, размер шрифта (на своё усмотрение).

Определить универсальный стиль, в котором переопределить цвет текста, размер шрифта, семейство шрифта (на своё усмотрение).

Переопределите тег <body>: цвет фона, размер шрифта, цвет шрифта, семейство шрифта (на своё усмотрение).

Пример выполнения практического задания: Каскадные таблицы стилей CSS. Использование свойств шрифта и цветовой гаммы.

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Практическое задание</title>
    <style>
        body {
            color: black;
            background-color: #efecb9;
            font-family: sans-serif;
            font-size: 100%;
        }
        p {
            color: blue;
            font-family: Courier, monospace;
            font-size: 20px;
        }
        .universal {
            color: purple;
            font-size: 30px;
            font-family: "Comic Sans MS", cursive, sans-serif;
        }
        table {
            width: 95%;
            height: 80%;
            margin: auto;
            border: 1px solid black;
            border-collapse: collapse;
        }
        td, th {
```

```
            border: 1px solid black;
            vertical-align: middle;
            padding: 10px;
        }
        caption h1 {
            margin: 0;
        }
    </style>
</head>
<body>
    <table>
        <caption><h1>Практическое задание</h1></caption>
        <tr>
            <td colspan="2">
                Переопределённый тег &lt;BODY&gt;:
                <br>
                <code>BODY { color: black; background-color: #c5c5c5; font-
family: sans-serif; font-size: 100%; }</code>
            </td>
        </tr>
        <tr>
            <td width="50%">
                После переопределения тега &lt;P&gt; в <code>P { color: blue;
font-family: Courier; font-size: 20px; }</code> текст выглядит следующим образом:
                <p>Переопределённый текст</p>
            </td>
            <td width="50%">
                А это универсальный стиль <code>.universal { color: purple; font-
size: 30px; font-family: Comic; }</code>
                <h1 class="universal">Вот так выглядит текст, переопределённый
универсальным стилем</h1>
            </td>
        </tr>
    </table>
</body>
</html>
```

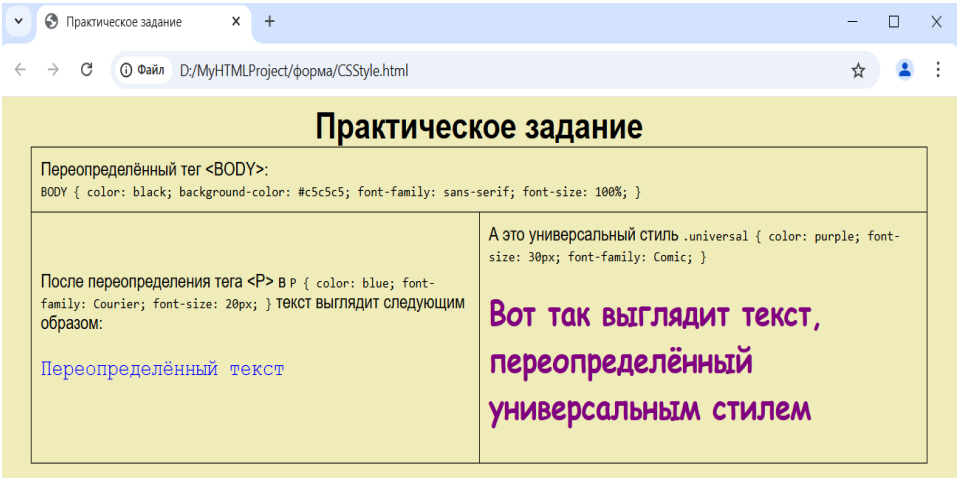


Рисунок 13 – Результат выполнения индивидуального задания

Лабораторная работа №10

Тема: Каскадные таблицы стилей CSS. Использование свойств текста и псевдостилей гиперссылок

Цель: Научить применять свойства текста и использовать псевдостили гипертекста

Краткие теоретические сведения

Применение CSS в современных веб-технологиях

Каскадные таблицы стилей (CSS, Cascading Style Sheets) произвели революцию в веб-разработке. Если ранее веб-дизайнеры не могли предсказать, как будет выглядеть их творение в разных браузерах, то теперь они могут контролировать все элементы: от начертания шрифта до точного расположения элементов на странице.

Предположим, вам нужно изменить цвет текста в HTML-документе с черного на синий. Для этого раньше использовались теги, например:
<P><FONT color="blue">Это синий текст</FONT></P>
Однако с развитием CSS подход изменился. Вместо использования устаревших тегов, вы можете назначить стиль через классы:
P.bluetext { color: blue; }
В результате в HTML-документе остается только структура:
<P class="bluetext">Это синий текст</P>

Здесь мы применяем CSS-класс `bluetext` к элементу `<P>`. Атрибут `class` позволяет задавать имя стилистического класса для тега, и такие классы поддерживаются всеми тегами. В случае необходимости можно задать стиль для всех тегов `<P>` без использования класса:
P { color: blue; }

Стиль –это указание браузеру как отображать тот или иной элемент, то есть как страница должна выглядеть. Всё, что находится внутри тега <body>, браузер будет отображать в соответствии с описанными стилями.

Способы применения стилей:

1. **Вложение стилей (Inline styles).** Это метод, при котором стиль прописывается непосредственно в теге через атрибут style. Это позволяет задать уникальное оформление для одного элемента, но такой подход не рекомендуется для масштабных проектов, так как делает код менее гибким и трудным для поддержки. Пример:
Это текст с синим цветом и размером 20px
2. **Встраивание стилей (Internal styles)** Этот метод заключается в том, что CSS-правила прописываются внутри тега `<style>`, который размещается в разделе `<head>` документа. Такой способ удобен, когда нужно применить стили только к одному документу, но он ограничивает

возможность повторного использования стилей в других страницах. Пример:

```
<html>
 <head>
  <style type="text/css">
   body {
    background-color: #f0f0f0;
    font-family: Arial, sans-serif;
   }
   p {
    color: blue;
    font-size: 20px;
   }
  </style>
 </head>
 <body>
  <p>Это текст с синим цветом и размером 20px</p>
 </body>
</html>
```

3. **Связывание внешнего файла стилей (External styles)** Это самый эффективный способ управления стилями для больших сайтов. В этом случае стили хранятся в отдельном файле, например `style.css`, и подключаются к HTML-документу с помощью тега `<link>`. Такой подход позволяет разделить структуру страницы и оформление, облегчая поддержку и улучшая производительность, так как один файл стилей может быть использован на нескольких страницах. Например, в файле `style.css`:

```
body {
  background-color: #f0f0f0;
  font-family: Arial, sans-serif;
}
p {
  color: blue;
  font-size: 20px;
}
```

В HTML-документе подключение стилей:

```
<html>
 <head>
  <link rel="stylesheet" type="text/css" href="style.css">
 </head>
 <body>
  <p>Это текст с синим цветом и размером 20px</p>
```

```
  </body>
</html>
```

Описание атрибутов тега <link>: rel="stylesheet" — указывает, что подключаемый файл является таблицей стилей. type="text/css" — задает тип подключаемого файла (в данном случае это CSS). href="style.css" — путь к файлу с таблицей стилей. Тег <link> является **непарным** (не требует закрывающего тега), и его нужно размещать внутри тега <head>.

Таким образом, для эффективного и удобного оформления веб-страниц рекомендуется использовать внешний файл стилей (метод связывания), что позволяет легко управлять стилями на нескольких страницах одновременно, улучшая поддержку и расширяемость проекта.

В **CSS** основными элементами являются **селекторы** и **правила.**
Селектор — это часть CSS, которая указывает, к каким элементам на странице применяются стили. Селектор может быть основан на **имени тега, идентификаторе** или **классе** элемента.
Правило записывается в фигурных скобках `{}` и состоит из **свойств** и их **значений**. Каждое свойство и значение разделяются двоеточием `:`, а различные свойства — точкой с запятой `;`.

```
Селектор { свойство1:значение;
           Свойство2:значение;
            ..............
           свойствоN: значение;
         }
```

Рассмотрим типы селекторов.

1. **Селекторы тегов** (или элементы)

Формат: `name {}` Этот селектор позволяет назначить стили всем элементам на странице, которые используют определённый тег. `name` — это имя тега.
Например,

```
p {
  color: blue;
  font-size: 16px;
}
```

В этом примере все теги `<p>` на странице будут отображаться с синим цветом текста и размером шрифта 16px.

2. **Селекторы идентификаторов**

Формат: `#name {}`
Этот селектор применяется к элементам с определённым уникальным идентификатором. Идентификатор задаётся в HTML через атрибут `id`.
Например,

```
#header {
  background-color: yellow;
  font-size: 20px;
}
```

В этом примере только элемент с `id="header"` получит жёлтый фон и размер шрифта 20px.
HTML пример: Заголовок

3. Селекторы классов

Формат: `.name {}` Этот селектор используется для назначения стилей группе элементов, имеющих одинаковое имя класса. Элементам в HTML нужно присваивать атрибут `class` с нужным значением. **Наример:**

```
.highlight {
  color: red;
  font-weight: bold;
}
```

В этом примере все элементы с классом `highlight` будут отображаться с красным цветом текста и жирным шрифтом.
HTML пример:

```
<p class="highlight">Это выделенный текст</p>
<span class="highlight">Это тоже выделенный текст</span>
```

4. Групповые селекторы

Формат: `селектор1, селектор2, селекторN {}`

Групповые селекторы позволяют назначить одинаковые стили для нескольких элементов. Они записываются через запятую, и могут быть использованы все типы селекторов, включая селекторы тегов, идентификаторов и классов. Например,

```
h1, p, .highlight {
  font-family: Arial, sans-serif;
  color: green;
}
```

В этом примере теги `<h1>`, `<p>`, а также все элементы с классом `highlight` будут иметь шрифт Arial и зелёный цвет текста.

Основные шаги настройки VS Code для работы с CSS:

1. Создание проекта и файлов

-Откройте VS Code.
-Создайте новую папку проекта или откройте существующую.
-Внутри этой папки создайте два файла:
index.html для HTML-кода.
style.css для CSS-стилей.

Расширения для VS Code

Для более комфортной работы с HTML и CSS в Visual Studio Code, рекомендуется установить несколько полезных плагинов:

Live Server (очень полезно!). Это расширение позволяет запустить локальный сервер прямо из VS Code и автоматически обновлять страницу в браузере при изменении кода.

Для установки:

Откройте **Extensions** (слева иконка квадрата с четырьмя блоками). В поиске введите **Live Server** и установите его.

После установки вы сможете запустить сервер, нажав правой кнопкой мыши на файл `index.html` и выбрав **"Open with Live Server"**.

CSS Peek. Этот плагин позволяет быстро переходить к определению стилей из HTML-файла, просто нажав на класс или ID.

Для установки: в Extensions найдите и установите **CSS Peek**.

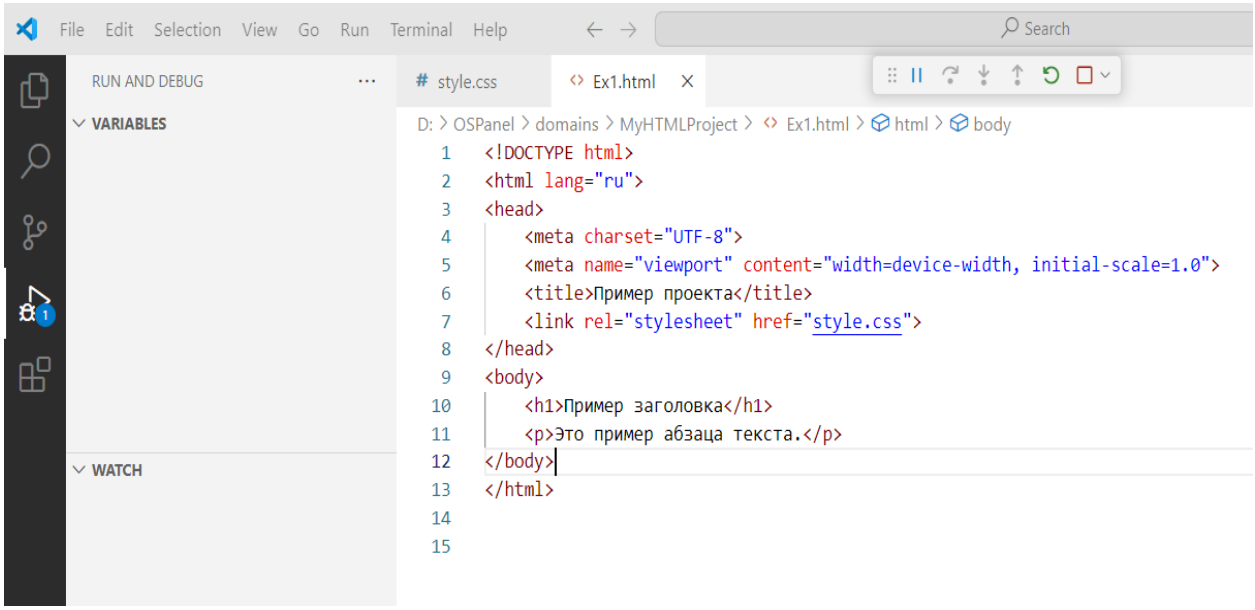
Prettier - Code Formatter. Это расширение автоматически форматирует ваш код (HTML, CSS, JavaScript) для улучшения читаемости.

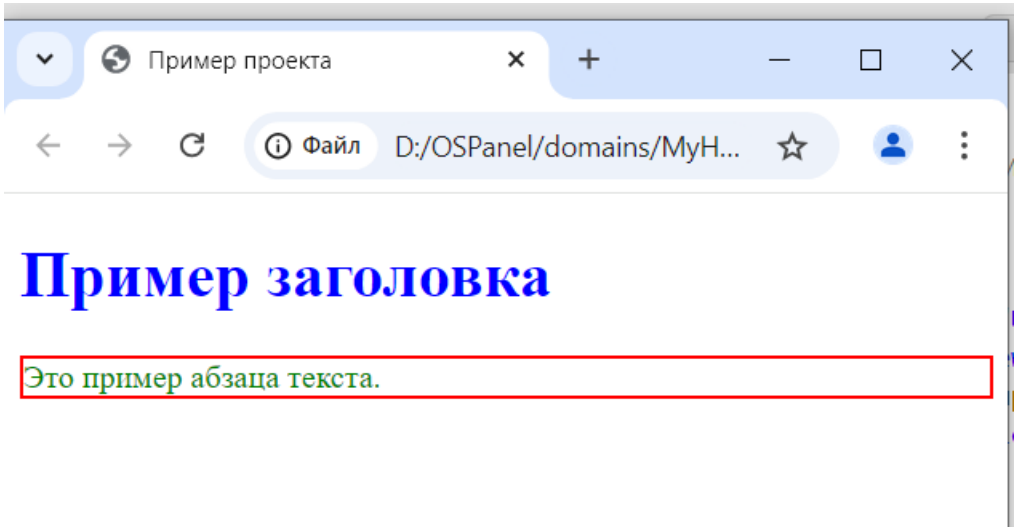
Для установки: найдите в Extensions **Prettier - Code Formatter** и установите его.

Файл style.css

```
h1 {
    color: blue;
}

p {
    color: green;
    border: 2px solid red;
}
```





Вариант 1. Основные стили текста и фона

Задача: Стилизовать страницу с заголовками, абзацами и списками.

1. Создайте страницу с заголовками (`<h1>`, `<h2>`, `<h3>`), абзацами (`<p>`) и списками (`<ul>` или `<ol>`).
2. Примените стили к заголовкам: измените цвет текста, размер шрифта и выравнивание.
3. Стилизуйте абзацы: измените межстрочный интервал, цвет фона и шрифт.
4. Примените стили к спискам: поменяйте маркеры или цифры на пользовательские и измените цвет элементов списка.
5. Добавьте рамку вокруг абзацев и измените отступы.

Вариант 2. Использование классов и идентификаторов

Задача: Стилизовать различные части страницы с использованием классов и идентификаторов.

1. Создайте несколько элементов, таких как `<div>`, `<section>`, `<article>`, и примените к ним классы и идентификаторы.
2. Стилизуйте элементы с одинаковыми классами, изменив цвет фона и отступы.
3. Для элементов с уникальными идентификаторами примените свои стили: добавьте рамки, измените цвет текста.
4. Примените различные шрифты к каждому элементу.

Вариант 3. Псевдоклассы и псевдоэлементы

Задача: Применить псевдоклассы и псевдоэлементы к элементам страницы.

1. Создайте ссылки на странице и стилизуйте их с помощью псевдоклассов: измените цвет ссылок для состояний `:hover`, `:visited`, `:active`.
2. Примените псевдокласс `:hover` для изменения фона и цвета текста кнопок (`<button>` или `<a>`).
3. Используйте псевдоэлементы `::before` и `::after` для добавления контента до и после текстовых блоков.

4. Примените псевдоэлемент `::first-letter`, чтобы увеличить и изменить цвет первой буквы в абзацах.

Вариант 4. Сетки и Flexbox

Задача: Создать страницу с использованием CSS Flexbox или CSS Grid для построения сетки.

1. Создайте страницу с контейнером, который содержит несколько блоков (`<div>` или `<section>`).
2. С помощью Flexbox выровняйте блоки горизонтально, задайте отступы между ними.
3. Примените свойства выравнивания (например, `justify-content`, `align-items`) для управления позиционированием элементов.
4. Используйте CSS Grid для создания сетки из блоков разного размера (например, сетка 2x2 или 3x3).
5. Сделайте один из блоков больше других, используя свойства Grid (`grid-column`, `grid-row`).

Вариант 5. Работа с медиа-запросами

Задача: Сделать страницу адаптивной для разных экранов.

1. Создайте страницу с несколькими разделами (например, заголовок, контент и подвал).
2. Примените медиа-запросы для изменения стилей в зависимости от ширины экрана (например, для экранов меньше 768px измените выравнивание текста, размер шрифта и отступы).
3. Настройте элементы так, чтобы они занимали 100% ширины экрана на мобильных устройствах и были выровнены в столбик.
4. Для экранов шире 768px сделайте элементы расположенными в строку, используя Flexbox или Grid.

Вариант 6. Подключение внешних шрифтов и иконок

Задача: Использовать внешние ресурсы для улучшения дизайна.

1. Подключите шрифты с помощью Google Fonts и примените их к заголовкам и абзацам.
2. Используйте иконки из внешнего источника (например, Font Awesome) и добавьте их в меню или раздел с контактами.
3. Измените размер иконок и цвет в зависимости от их состояния (`:hover`).
4. Используйте несколько разных шрифтов для различных разделов страницы.

Вариант 7. Анимации и переходы

Задача: Добавить CSS-анимации и переходы к элементам страницы.

1. Создайте кнопки и примените к ним плавные переходы (`transition`) для изменения цвета и фона при наведении курсора.
2. Используйте CSS-анимации для создания эффекта плавного перемещения или изменения размера элемента при загрузке страницы.
3. Примените ключевые кадры (`@keyframes`) для анимации объектов (например, двигающийся блок, меняющий цвет).

Вариант 8. Создание формы с CSS-стилями

Задача: Стилизовать форму для отправки данных.

1. Создайте форму с элементами: текстовые поля, выпадающие списки, чекбоксы и кнопки.
2. Стилизуйте текстовые поля, изменив их ширину, высоту и цвет границ.
3. Примените стили к кнопкам: измените их цвет при наведении, добавьте тень или градиент.
4. Сделайте форму адаптивной, чтобы её элементы изменяли размеры в зависимости от ширины экрана.

Вариант 9. Использование фонов и изображений

Задача: Применить фоны и изображения к элементам страницы.

1. Добавьте фоновое изображение к заголовку или разделу страницы и сделайте его фиксированным.
2. Используйте свойства `background-size`, `background-position` и `background-repeat` для настройки отображения фонового изображения.
3. Добавьте к блоку градиентный фон и настройте его углы.
4. Примените полупрозрачные фоны к элементам текста для создания эффекта наложения.

Вариант 10. Создание навигационного меню

Задача: Стилизовать навигационное меню.

1. Создайте горизонтальное или вертикальное навигационное меню с использованием элементов списка (`<ul>` и `<li>`).
2. Стилизуйте ссылки в меню: измените их размер, цвет и добавьте эффекты при наведении (например, изменение цвета, подчеркивание).
3. Сделайте меню адаптивным, чтобы на мобильных устройствах оно превращалось в выпадающее (hamburger) меню.

Лабораторная работа №11

Тема: Блочная модель CSS

Цель: Научиться применять свойства блочной модели CSS для управления внешним видом и позиционированием HTML-элементов.

Инструменты: HTML, CSS, браузер.

Краткие теоретические сведения

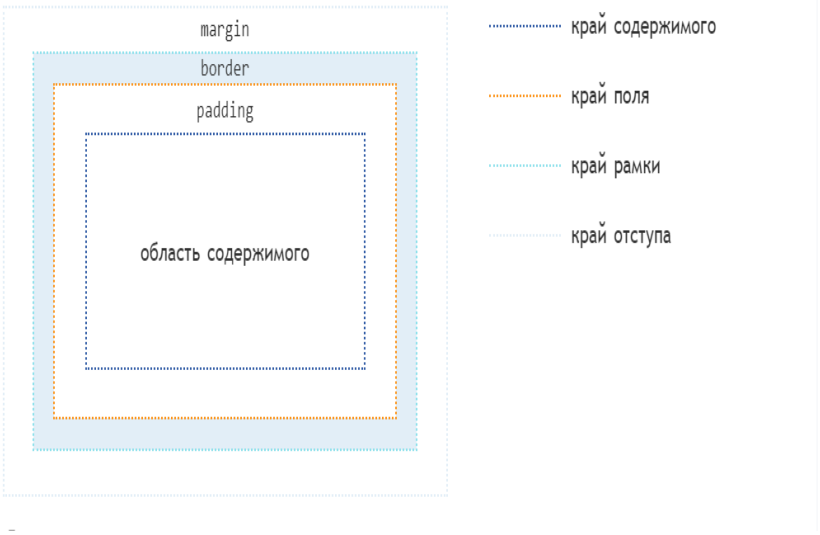
Блочная модель CSS — это способ представления элементов веб-страницы в виде прямоугольных блоков. Каждый элемент состоит из:

1. **Content** — основное содержимое (текст, изображения).
2. **Padding** — внутренние отступы вокруг контента.
3. **Border** — рамка элемента.
4. **Margin** — внешние отступы вокруг элемента.

Эти свойства влияют на общий размер и расположение элемента на странице.

Содержимое (content) — пространство, в котором размещается контент элемента (текст, изображения и т.д.). **Внутренний отступ (padding)** — пространство между содержимым и границей элемента. **Граница (border)** — рамка вокруг элемента, которая отделяет padding от внешнего отступа. **Внешний отступ (margin)** — пространство между границей элемента и другими элементами.

На схеме блочной модели браузеры вычисляют итоговый размер элемента, включая все вышеуказанные области.

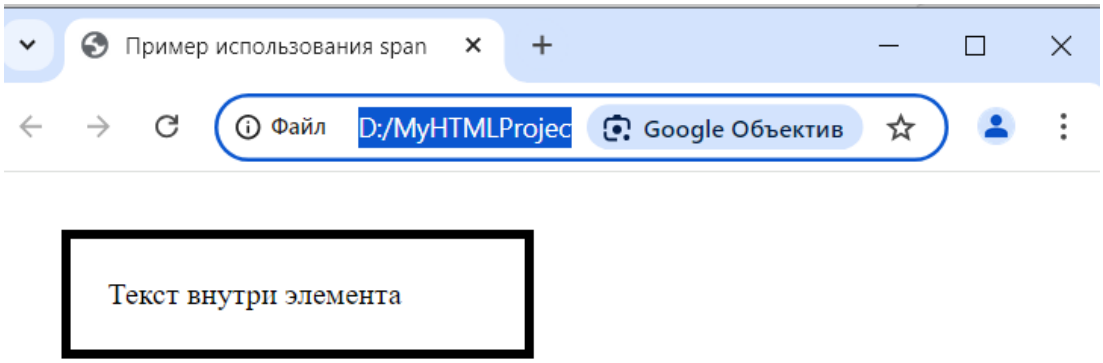


Например,

exmp2.html

D: > MyHTMLProject > exmp2.html > html > head > style > .box

```
1  <!DOCTYPE html>
2  <html lang="ru">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Пример использования span</title>
7    <style>
8
9    .box {
10   width: 200px;          /* Ширина содержимого */
11   padding: 20px;         /* Внутренний отступ */
12   border: 5px solid ■black; /* Граница */
13   margin: 30px;          /* Внешний отступ */
14 }
15   </style>
16 </head>
17 <body>
18   <div class="box">Текст внутри элемента</div>
19 </body>
20 </html>
```



Текст внутри элемента

Очистить стили браузеров для всех элементов можно при помощи универсального селектора:

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
```

Отступы окружают край рамки элемента, обеспечивая расстояние между соседними блоками. Свойства отступов определяют их толщину. Применяются ко всем элементам, кроме внутренних элементов таблицы. Сокращенное свойство margin задает отступы для всех четырех сторон, а его

подсвойства задают отступ только для соответствующей стороны. Смежные вертикальные отступы элементов в блочной модели схлопываются.

Схлопывание вертикальных отступов

Смежные вертикальные отступы двух или более элементов уровня блока margin объединяются (перекрываются). При этом ширина общего отступа равна ширине большего из исходных. Исключение составляют отступы корневого элемента, которые не схлопываются.

Объединение отступов выполняется только для блочных элементов в нормальном потоке документа. Если среди схлопывающихся отступов есть отрицательные значения, то браузер добавит отрицательное значение к положительному, а полученный результат и будет расстоянием между элементами. Если положительных отступов нет, то максимум абсолютных значений соседних отступов вычитается из нуля.

Отступы не схлопываются:

Между плавающим блоком и любым другим блоком;

У плавающих элементов и элементов со значением overflow, отличным от visible, со своими дочерними элементами в потоке;

У абсолютно позиционированных элементов, даже с их дочерними элементами;

У строчно-блочных элементов.

Для предотвращения проблемы схлопывания рекомендуется задавать для всех элементов только верхний или нижний margin.

Выпадение вертикальных отступов

Если внутри одного блока расположить другой блок и задать ему margin-top, то внутренний блок прижмется к верхнему краю родительского, а у родительского элемента появится отступ сверху, т.е. внутренний блок «выпадет» из родительского блока. Если у родительского элемента также был задан верхний отступ, то выберется наибольшее из значений.

Чтобы избавиться от эффекта выпадения, можно задать родительскому элементу padding-top или добавить border-top: 1px solid transparent.

Физические свойства отступов: свойства margin-top, margin-right, margin-bottom, margin-left

Свойства устанавливают верхний, правый, нижний и левый отступ блока элемента соответственно. Отрицательные значения допускаются, но могут существовать ограничения для конкретной реализации.

Значения:	
длина	Размер отступа задается в единицах длины, например, px, in, em. Значение по умолчанию 0.
%	Вычисляется относительно ширины блока контейнера. Изменяются, если изменяется ширина родительского элемента.
auto	Для элементов уровня строки, плавающих (float) значения margin-left или margin-right вычисляются в 0. Если для элементов уровня блока задано margin-left: auto или margin-right: auto — соответствующее поле расширяется до края содержащего блока, если оба — их значения становятся равными, что горизонтально центрирует элемент относительно краев содержащего блока.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис:

```
margin-top: 20px;
margin-right: 1em;
margin-bottom: 5%;
margin-left: auto;
margin-top: inherit;
margin-right: initial;
```

Краткая запись отступов: свойство margin

Свойство margin является сокращенным свойством для установки margin-top, margin-right, margin-bottom и margin-left в одном объявлении.

Если существует только одно значение, оно применяется ко всем сторонам.

Если два — верхний и нижний отступы устанавливаются на первое значение, а правый и левый — устанавливаются на второе.

Если имеется три значения — верхний отступ устанавливается на первое значение, левый и правый — на второе, а нижний — на третье.

Если есть четыре значения — они применяются сверху, справа, снизу и слева соответственно.

3. Поля элемента

Область полей представляет собой пространство между краем области содержимого и рамкой элемента. Свойства полей определяют толщину их области. Применяются ко всем элементам, кроме внутренних элементов

таблицы (за исключением ячеек таблицы). Сокращенное свойство padding задает поля для всех четырех сторон, а подсвойства устанавливают только их соответствующие стороны.

Фоны элемента по умолчанию закрашивают поля элемента и пространство под его рамкой. Это поведение можно настроить с помощью свойств background-origin и background-clip.

Физические свойства полей: свойства padding-top, padding-right, padding-bottom, padding-left

Свойства устанавливают верхнее, правое, нижнее и левое поля соответственно. Отрицательные значения недопустимы.

Свойства не наследуются.

Значения:	
длина	Поля элемента задаются при помощи единиц длины, например, px, pt, cm. Значение по умолчанию 0.
%	Вычисляются относительно ширины родительского элемента, могут меняться при изменении ширины элемента. Поля сверху и снизу равны полям слева и справа, т.е. верхние и нижние поля тоже вычисляются относительно ширины элемента.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис:

```
padding-top: 0.5em;
padding-right: 0;
padding-bottom: 2cm;
padding-left: 10%;
padding-top: inherit;
padding-bottom: initial;
```

Краткая запись полей: свойство padding

Свойство padding является сокращенным свойством для установки padding-top, padding-right, padding-bottom и padding-left в одном объявлении.

Если существует только одно значение, оно применяется ко всем сторонам.

Если есть два значения, верхнее и нижнее поля устанавливаются на первое значение, а правое и левое — на второе.

Если имеется три значения, верхнее поле устанавливается на первое значение, левое и правое — на второе, а нижнее — на третье.

Если есть четыре значения — они применяются сверху, справа, снизу и слева соответственно.

Рамки элемента

Рамки элемента заполняют область рамок, визуально очерчивая края блока. Свойства рамок определяют толщину области границы блока, а также ее стиль и цвет.

Индивидуальное задание

Создайте HTML-файл с подключённым CSS. Оформите несколько блоков с разными параметрами блочной модели.

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>Блочная модель CSS</title>
    <style>
        .box {
            width: 200px;
            height: 100px;
            background-color: lightblue;
            padding: 20px;
            border: 5px solid blue;
            margin: 30px;
        }

        .box2 {
            width: 200px;
            height: 100px;
            background-color: lightgreen;
            padding: 10px;
            border: 3px dashed green;
            margin: 10px;
        }
    </style>
</head>
<body>
    <h1>Пример блочной модели</h1>
    <div class="box">Блок 1</div>
    <div class="box2">Блок 2</div>
</body>
```

```
</html>
```

Результат задания представлен на рисунке 14

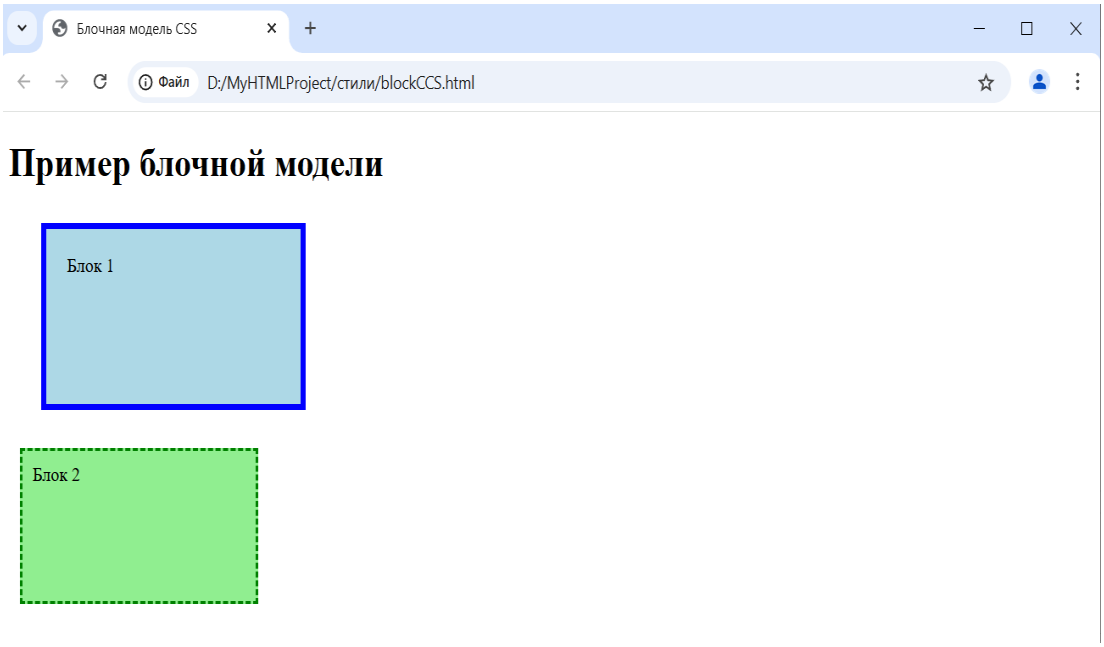


Рисунок 14 – Результат работы задания

Варианты индивидуальных заданий:

1. **Задание 1.1:** Создайте веб-страницу, содержащую несколько блочных элементов. Используйте CSS для задания отступов (margin) и полей (padding) для каждого из элементов.

   Вариант 1: Используйте `margin` для создания внешних отступов у всех элементов. Задайте разные отступы для каждой стороны.

   Вариант 2: Используйте `padding` для внутренних отступов у всех элементов. Создайте элементы с разным цветом фона, чтобы визуально различать внутренние отступы.

2. **Задание 1.2:** Примените рамки (`border`) к блочным элементам и настройте их стиль, цвет и толщину.

   Вариант 1: Задайте рамку только для одного элемента и измените её толщину с каждой стороны.

   Вариант 2: Сделайте рамки разных цветов и стилей для всех сторон (например, `dotted`, `solid`, `dashed`).

3. **Задание 1.3:** Реализуйте блоки с фиксированной высотой и шириной. Экспериментируйте с процентными и фиксированными значениями размеров (`width`, `height`).

   Вариант 1: Задайте ширину и высоту в пикселях.

   Вариант 2: Задайте ширину и высоту в процентах от родительского элемента.

Лабораторная работа №12

Тема:CSS-позиционирование

Цель работы: научиться навыкам работы с CSS позиционированием, научиться применять различные методы и изучить их сочетание в реальных задачах веб-разработки.

Индивидуальное задание 12.1. Центрирование элемента по горизонтали и вертикали

Условие: Нужно разместить элемент посередине родительского контейнера (по горизонтали и вертикали).

Решение:

Для центрирования элемента внутри родителя можно использовать несколько подходов. Рассмотрим вариант с **абсолютным позиционированием** и применением transform.

Объяснение:

1. Родительский контейнер .parent имеет position: relative, чтобы элемент .child был позиционирован относительно этого контейнера.
2. Элемент .child с position: absolute сначала смещается на 50% сверху и слева (top: 50% и left: 50%).
3. Использование transform: translate(-50%, -50%) корректирует смещение, позволяя центрировать элемент по обоим направлениям относительно его собственного размера (Рисунок 15)

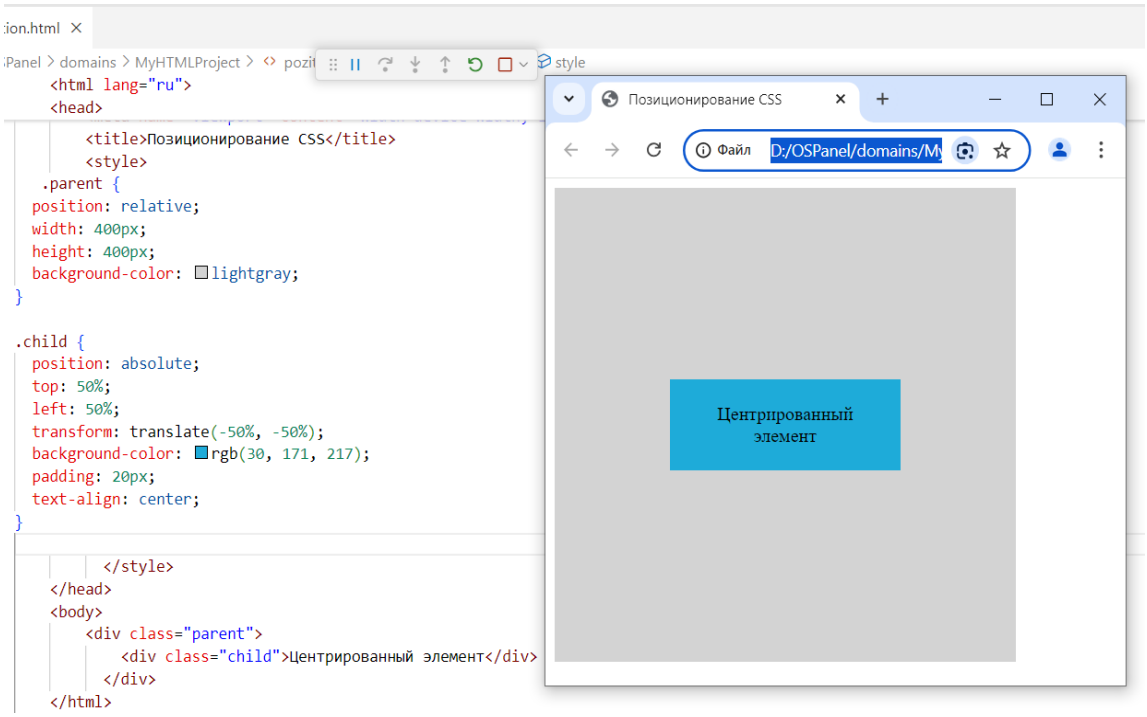


Рисунок 15 – Результат работы индивидуального задания 12.1

Индивидуальное задание 12.2. Закрепить элемент внизу экрана

Условие: Нужно сделать так, чтобы элемент оставался в нижней части экрана при прокрутке страницы.

Решение:

Для этого задачи используется **фиксированное позиционирование**.

Объяснение:

1. Свойство position: fixed делает элемент фиксированным относительно окна браузера.
2. Свойство bottom: 0 закрепляет элемент в нижней части окна.
3. Элемент будет оставаться на месте, даже если страница будет прокручиваться. (Рисунок 16)

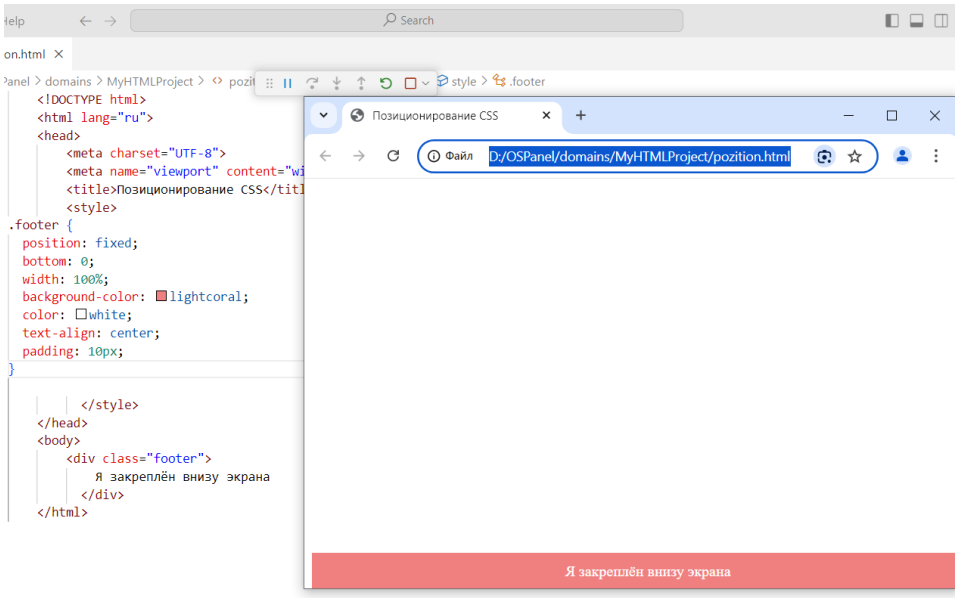


Рисунок 16 – Результат работы индивидуального задания 12.2

Индивидуальное задание 12.3. Обтекание текста вокруг изображения

Условие: Нужно разместить изображение слева, чтобы текст обтекал его с правой стороны.

Решение:

Для выполнения этой задачи используется **обтекание** через float.

Объяснение:

1. Свойство float: left заставляет изображение "всплыть" влево.
2. Текст в параграфе будет автоматически обтекать изображение с правой стороны.
3. Свойство margin-right создаёт отступ между изображением и текстом.(Рисунок 17)

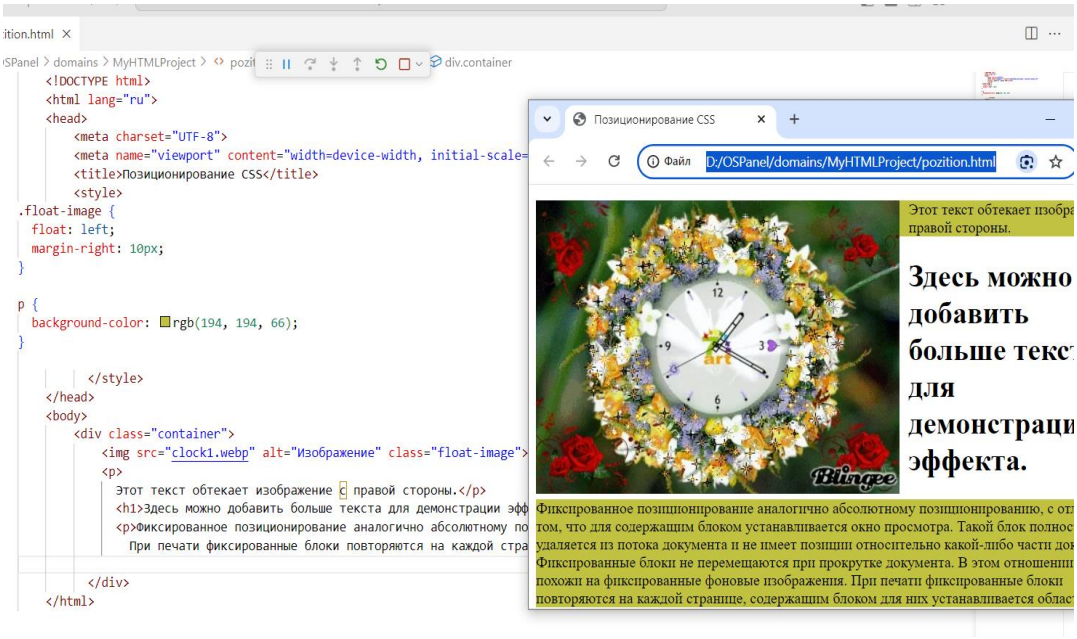


Рисунок 17 – Результат работы индивидуального задания 12.3

Индивидуальное задание 12.4. Создать липкий заголовок

Условие: Сделать заголовок, который остаётся на экране, когда пользователь прокручивает страницу вниз, но только до определённого момента.

Решение:

Для решения этой задачи используется **липкое позиционирование** (position: sticky).

Объяснение:

1. Свойство position: sticky позволяет заголовку оставаться фиксированным в верхней части экрана, когда пользователь прокручивает страницу.
2. top: 0 определяет расстояние, на котором заголовок должен "залипнуть" относительно верхней части окна. (Рисунок 18)

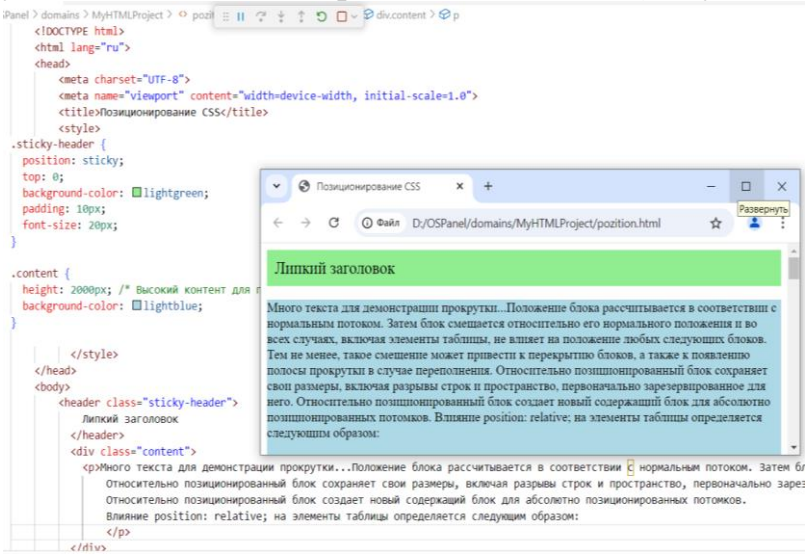


Рисунок 18 – Результат работы индивидуального задания 12.4

Индивидуальное задание 12.5. Абсолютное позиционирование внутри контейнера

Условие: Нужно поместить элемент в нижний правый угол контейнера.

Решение:

Для выполнения задачи используется **абсолютное позиционирование**.

Объяснение:

1. Контейнер .container имеет position: relative, чтобы внутренний элемент .box был позиционирован относительно его границ.
2. Свойства bottom: 0 и right: 0 размещают элемент в нижнем правом углу контейнера. (Рисунок 19)

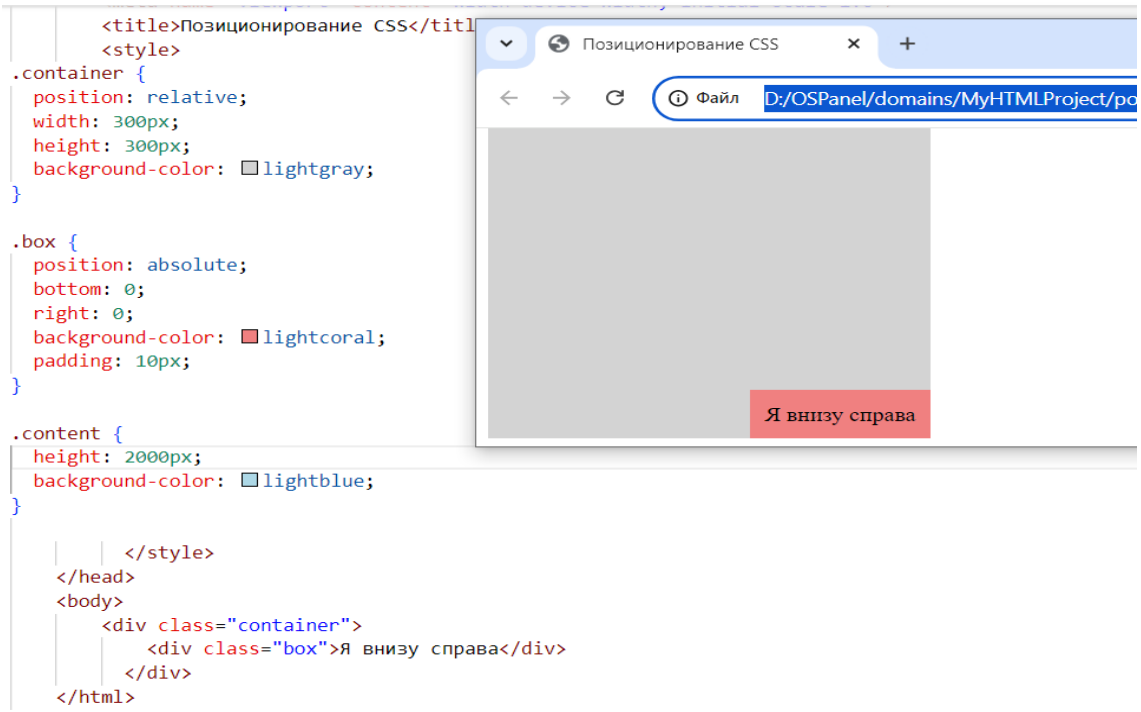


Рисунок 19 – Результат работы индивидуального задания 12.5

Варианты для задачи 1 (Центрирование элемента)

1. **Центрирование двух элементов**: Нужно разместить два элемента по центру страницы — один элемент по горизонтали и вертикали, а другой только по горизонтали (при этом оба элемента остаются независимыми друг от друга).
2. **Центрирование с учетом адаптивности**: Элемент должен оставаться центрированным как в больших, так и в маленьких окнах браузера, а также на мобильных устройствах, без использования фиксированных размеров (используя медиазапросы).
3. **Центрирование круга**: предлагается центрировать не прямоугольный элемент, а круг (элемент с радиусом, заданным через border-radius), и сохранить его центрирование при изменении размеров окна.

Варианты для задачи 2 (Закрепление элемента внизу экрана)

1. **Закрепление меню вверху экрана**: Создать навигационное меню, которое всегда остаётся прикреплённым в верхней части экрана при прокрутке страницы.
2. **Фиксированный блок с кнопкой «Вверх»**: нужно создать кнопку, которая будет всегда находиться в правом нижнем углу страницы и позволит возвращаться наверх при нажатии.
3. **Фиксированное уведомление**: Создать небольшое уведомление (блок с текстом), которое появляется в нижней части экрана при загрузке страницы и остаётся там до закрытия пользователем.

Варианты для задачи 3 (Обтекание текста вокруг изображения)

1. **Обтекание нескольких изображений**: Добавить несколько изображений (например, с float: left и float: right), чтобы текст обтекал их по обе стороны. Необходимо реализовать правильные отступы между изображениями и текстом.
2. **Обтекание с разными размерами изображений**: Необходимо разместить изображения с разными размерами, чтобы текст корректно обтекал каждое из них. Нужно учесть отступы между изображениями и текстом.
3. **Создание статьи с обтеканием нескольких блоков**: Представить статью с несколькими изображениями и текстом, где студенты должны расположить изображения слева и справа от текста так, чтобы структура страницы выглядела эстетично.

Варианты для задачи 4 (Липкий заголовок)

1. **Липкое меню с подзаголовками**: Нужно создать меню, которое «липнет» к верхней части экрана при прокрутке. При этом должны появляться разные подзаголовки в зависимости от секции, через которую прокручивается страница.
2. **Липкий блок с кнопками в боковой панели**: Нужно сделать панель с кнопками (например, соцсети) на боковой части экрана, которая остаётся видимой и «липкой» при прокрутке страницы.
3. **Липкий заголовок секции**: Разбить длинную страницу на несколько секций, каждая из которых имеет свой заголовок, который становится липким, когда пользователь прокручивает к этой секции. После прокрутки заголовок остаётся на экране, пока не будет достигнута следующая секция.

Варианты для задачи 5 (Абсолютное позиционирование)

1. **Позиционирование нескольких элементов**: Задание на размещение трёх элементов в разных углах родительского контейнера: один внизу справа, другой внизу слева, третий — вверху справа. Нужно следить за сохранением их позиционирования при изменении размера окна.
2. **Позиционирование относительно динамического родителя**: Задание на абсолютное позиционирование элемента внутри родительского контейнера, который изменяет свои размеры в зависимости от контента (например, контейнер с текстом, который растягивается).
3. **Плавающая кнопка «связаться»**: Создать кнопку для обратной связи, которая расположена в правом нижнем углу экрана и остаётся там при изменении размеров страницы.

Лабораторная работа №13

Тема: Стилизация CCS текста.

Цель: Изучить и научиться применять свойства CSS, предназначенные для стилизации текста, такие как изменение шрифта, размера, цвета, выравнивания, интервалов между строками и символами, а также декорирование текста

Краткий теоретический материал

Внешняя таблица стилей

Внешняя таблица стилей подключается к веб-странице с помощью элемента <link>, расположенного внутри раздела <head></head>. Такие стили работают для всех страниц сайта.

Внутренние стили

Внутренние стили встраиваются в раздел <head></head> HTML-документа и определяются внутри элемента <style></style>. Внутренние стили имеют приоритет над внешними, но уступают встроенным стилям (заданным через атрибут style).

Встроенные стили

Когда мы пишем встроенные стили, мы пишем CSS-код в HTML-файл, непосредственно внутри элемента с помощью атрибута style:

```
<p style="font-weight: bold; color: red;">Обратите внимание на этот текст.</p>
```

Такие стили действуют только на тот элемент, для которого они заданы.

Правило @import

Правило @import позволяет загружать внешние таблицы стилей. Чтобы директива @import работала, она должна располагаться в таблице стилей (внешней или внутренней) перед всеми остальными правилами.

Правило @import также используется для подключения веб-шрифтов:

exmp2.html | Exmp2.css

D: > MyHTMLProject > Exmp2.css

```
1  @import url(https://fonts.googleapis.com/css?family=Open+Sans&subset=latin,cyrillic);
```

Help | Search

exmp2.html

D: > MyHTMLProject > exmp2.html > html > body > p

```
1  <!DOCTYPE html>
2  <html lang="ru">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title> Мой документ</title>
7      <style>
8          @import url(mobile.css);
9          p {
10         font-size: 1.4em;
11         color: rgb(128, 128, 128);
12         }
13         </style>
14 </head>
15 <body>
16     <p style="font-weight: bold; color: red;">Обратите внимание на этот текст.</p>
17   <p> Расписание занятий
18   </p>
19 </body>
20 </html>
```

Рисунок 20 - Пример использования правила @import

Виды селекторов.

Универсальный селектор

Например, * {margin: 0;} обнулит внешние отступы для всех элементов сайта. Также селектор может использоваться в комбинации с псевдоклассом или псевдоэлементом: *:after {CSS-стили}, *:checked {CSS-стили}.

Селектор элемента, например,

h1 {font-family: Lobster, cursive;} задаст общий стиль форматирования всех заголовков h1.

Селектор класса

Например, для создания заголовка с классом headline необходимо добавить атрибут class со значением headline в открывающий тег <h1> и задать стиль для указанного класса. Стили, созданные с помощью класса, можно применять к другим элементам, не обязательно данного типа.

Селектор идентификатора

Селектор идентификатора позволяет форматировать один конкретный элемент. Значение id должно быть уникальным, на одной странице может встречаться только один раз и должно содержать хотя бы один символ. Значение не должно содержать пробелов.

Селектор потомка

Селекторы потомков применяют стили к элементам, расположенным внутри элемента-контейнера. Например, ul li {text-transform: uppercase;} — выберет все элементы li, являющиеся потомками всех элементов ul.
p.first a {color: green;} — данный стиль применится ко всем ссылкам, потомкам абзаца с классом first;
p .first a {color: green;} — если добавить пробел, то будут стилизованы ссылки, расположенные внутри любого элемента класса .first, который является потомком элемента <p>;
.first a {color: green;} — данный стиль применится к любой ссылке, расположенной внутри другого элемента, обозначенного классом .first.

Дочерний селектор

Дочерний элемент является прямым потомком содержащего его элемента.

Например, p > strong — выберет все элементы strong, являющиеся дочерними по отношению к элементу p.

Сестринский селектор

Сестринские отношения возникают между элементами, имеющими общего родителя.

h1 + p — выберет все первые абзацы, идущие непосредственно за любым элементом <h1>, не затрагивая остальные абзацы;

h1 ~ p — выберет все абзацы, являющиеся сестринскими по отношению к любому заголовку h1 и идущие сразу после него.

Селектор атрибута

Селекторы атрибутов выбирают элементы на основе имени атрибута или значения атрибута:

[атрибут] — все элементы, содержащие указанный атрибут, [alt] — все элементы, для которых задан атрибут alt;

селектор[атрибут] — элементы данного типа, содержащие указанный атрибут, img[alt] — только картинки, для которых задан атрибут alt;

селектор[атрибут="значение"] — элементы данного типа, содержащие указанный атрибут с конкретным значением, img[title="flower"] — все картинки, название которых содержит слово flower;

селектор[атрибут~="значение"] — элементы частично содержащие данное значение, например, если для элемента задано несколько классов через пробел, p[class~="feature"] — абзацы, имя класса которых содержит feature;

Селектор псевдокласса

Псевдоклассы — это классы, фактически не прикрепленные к HTML-элементам. Они позволяют применить CSS-правила к элементам при совершении события или подчиняющимся определенному правилу.

Псевдоклассы характеризуют элементы со следующими свойствами:

:link — не посещенная ссылка;

:visited — посещенная ссылка;

:hover — любой элемент, по которому проводят курсором мыши;

:focus — интерактивный элемент, к которому перешли с помощью клавиатуры или активировали посредством мыши;

:active — элемент, который был активизирован пользователем;

:valid — поля формы, содержимое которых прошло проверку в браузере на соответствие указанному типу данных;

:invalid — поля формы, содержимое которых не соответствует указанному типу данных;

:enabled — все активные поля форм;

Например , когда вы добавите этот код на страницу и взаимодействуете со ссылкой, её цвет будет изменяться в зависимости от состояния, что помогает пользователям лучше ориентироваться в том, какие ссылки они уже посещали и какие действия они совершают, Рисунки 21.1 и 21.2

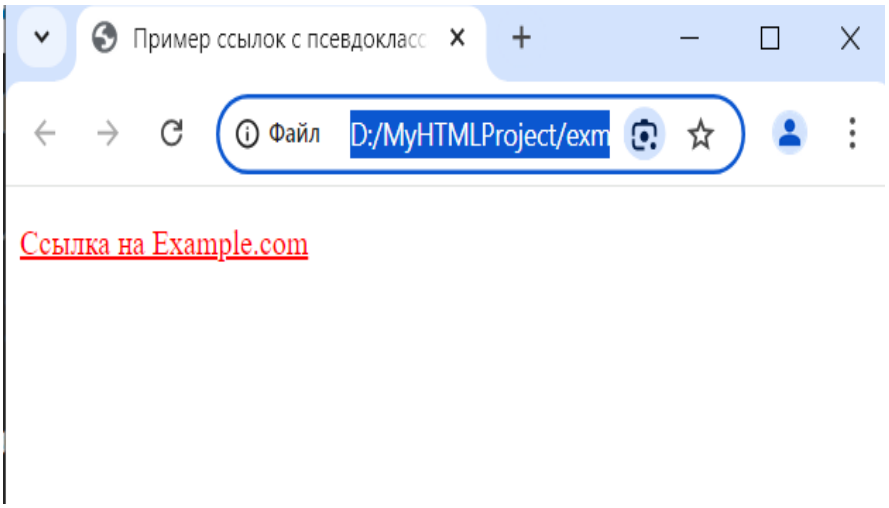


Рисунок 21.1 – Результат работы посещения ссылки

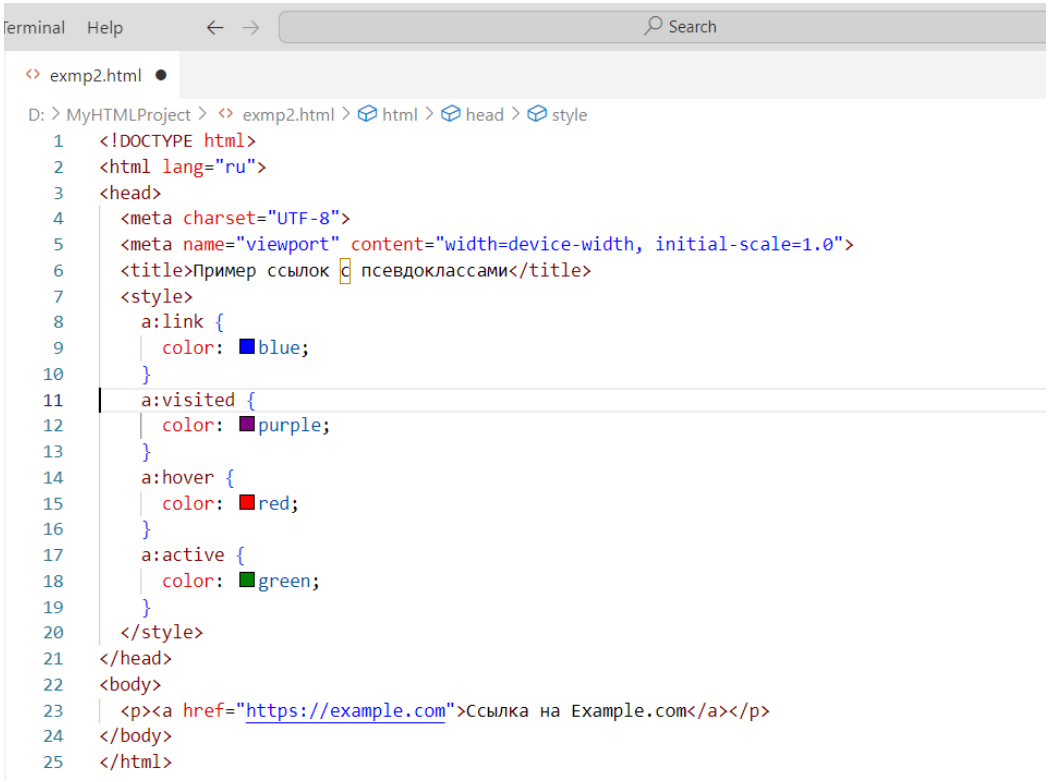


Рисунок 21.2 – Фрагмент кода с использование ccs

Селектор структурных псевдоклассов

Структурные псевдоклассы отбирают дочерние элементы в соответствии с параметром, указанным в круглых скобках:

:nth-child(odd) — нечётные дочерние элементы;

:nth-child(even) — чётные дочерние элементы;

:nth-child(3n) — каждый третий элемент среди дочерних;

:nth-child(3n+2) — выбирает каждый третий элемент, начиная со второго дочернего элемента (+2);

:nth-child(n+2) — выбирает все элементы, начиная со второго;

:nth-child(3) — выбирает третий дочерний элемент;

:nth-last-child() — в списке дочерних элементов выбирает элемент с указанным местоположением, аналогично с :nth-child(), но начиная с последнего, в обратную сторону;

Селектор структурных псевдоклассов типа

Указывают на конкретный тип дочернего элемента:

:nth-of-type() — выбирает элементы по аналогии с :nth-child(), при этом берёт во внимание только тип элемента;

:first-of-type — выбирает первый дочерний элемент данного типа;

:last-of-type — выбирает последний элемент данного типа;

:nth-last-of-type() — выбирает элемент заданного типа в списке элементов в соответствии с указанным местоположением, начиная с конца;

:only-of-type — выбирает единственный элемент указанного типа среди дочерних элементов родительского элемента.

Селектор псевдоэлемента

Псевдоэлементы используются для добавления содержимого, которое генерируется с помощью свойства content:

:first-letter — выбирает первую букву каждого абзаца, применяется только к блочным элементам;

:first-line — выбирает первую строку текста элемента, применяется только к блочным элементам;

:before — вставляет генерируемое содержимое перед элементом;

:after — добавляет генерируемое содержимое после элемента.

Наследование и каскад

Наследование и каскад — два фундаментальных понятия в CSS, которые тесно связаны между собой.

Наследование заключается в том, что элементы наследуют свойства от своего родителя (элемента, их содержащего).

Каскад проявляется в том, как разные виды таблиц стилей применяются к документу, и как конфликтующие правила переопределяют друг друга.

Спецификацией CSS предусмотрено наследование свойств, относящихся к текстовому содержимому страницы, таких как color, font, letter-spacing, line-height, list-style, text-align, text-indent, text-transform, visibility, white-space и word-spacing. Во многих случаях это удобно, так как не нужно задавать размер шрифта и семейство шрифтов для каждого элемента веб-страницы.

Свойства, относящиеся к форматированию блоков, не наследуются. Это background, border, display, float и clear, height и width, margin, min-max-height и -width, outline, overflow, padding, position, text-decoration, vertical-align и z-index.

Стилизация CCS текста

1. Преобразование регистра с `text-transform`, рисунок 22

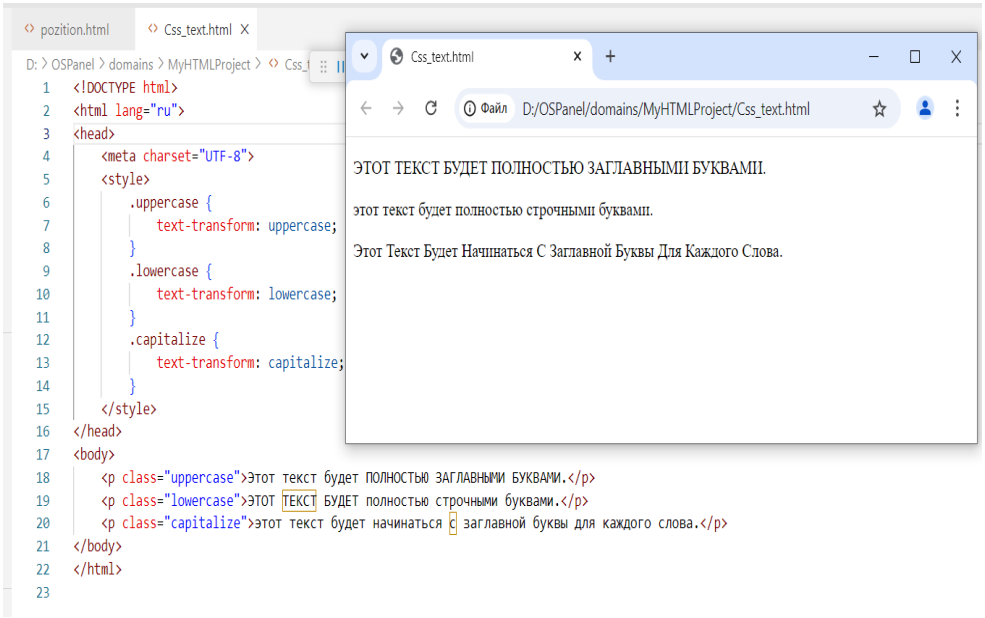


Рисунок 22 – пример изменения регистра с помощью `text-transform`

Результат:

- Первый абзац будет целиком в верхнем регистре.
- Второй абзац будет в нижнем регистре.
- В третьем абзаце каждое слово начнется с заглавной буквы.

2. Управление пробелами с `white-space`, рисунок 23

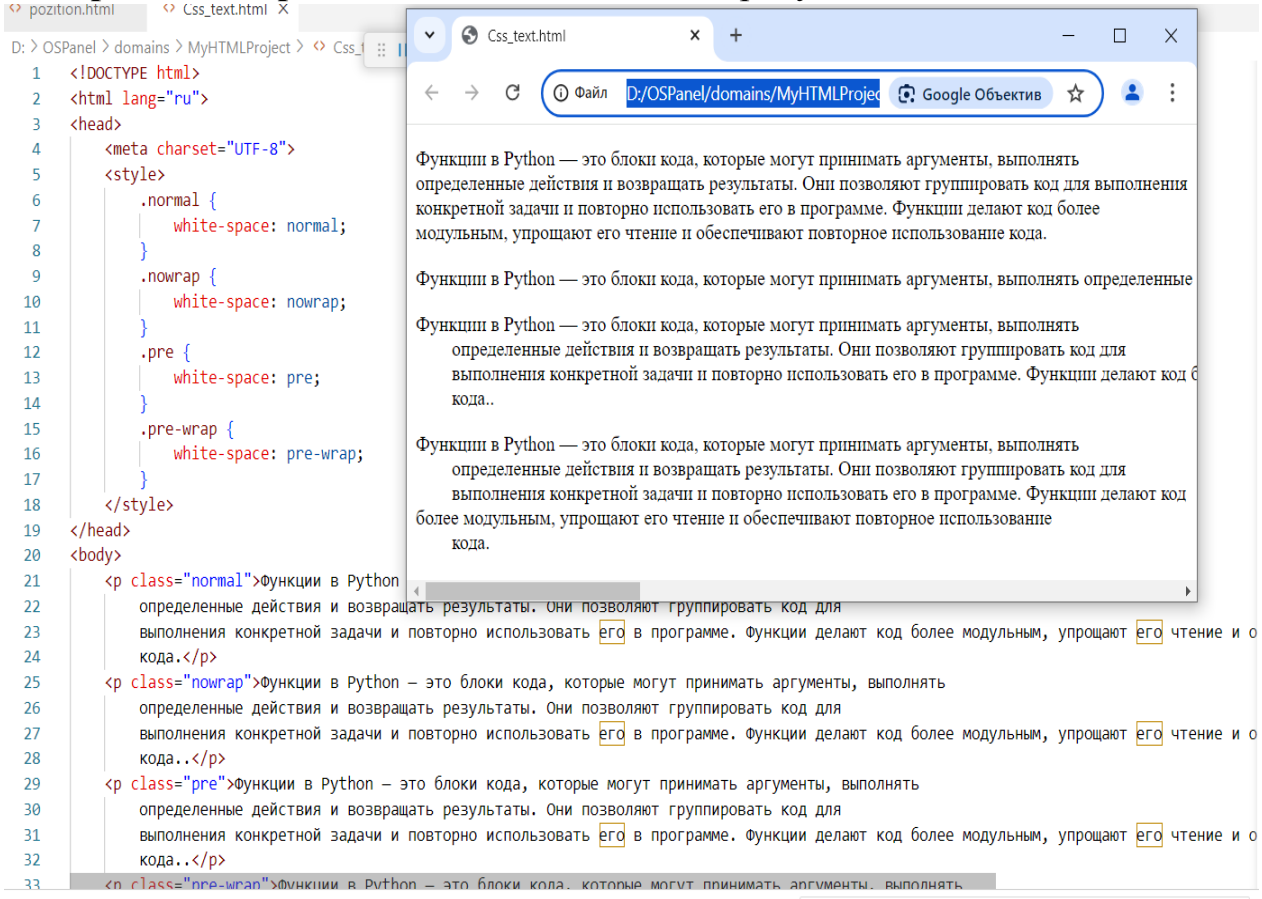


Рисунок 23 – Пример управление пробелами с `white-space`

Первый абзац (`normal`) сжимает все пробелы и игнорирует переносы строк.

Второй абзац (`nowrap`) не переносится на новую строку.

Третий абзац (`pre`) сохраняет все пробелы и переносы, как они заданы.

Четвертый абзац (`pre-wrap`) сохраняет пробелы и переносы, но текст может переноситься при необходимости.

3. Перенос слов с `overflow-wrap` и `hyphens`, рисунок 24

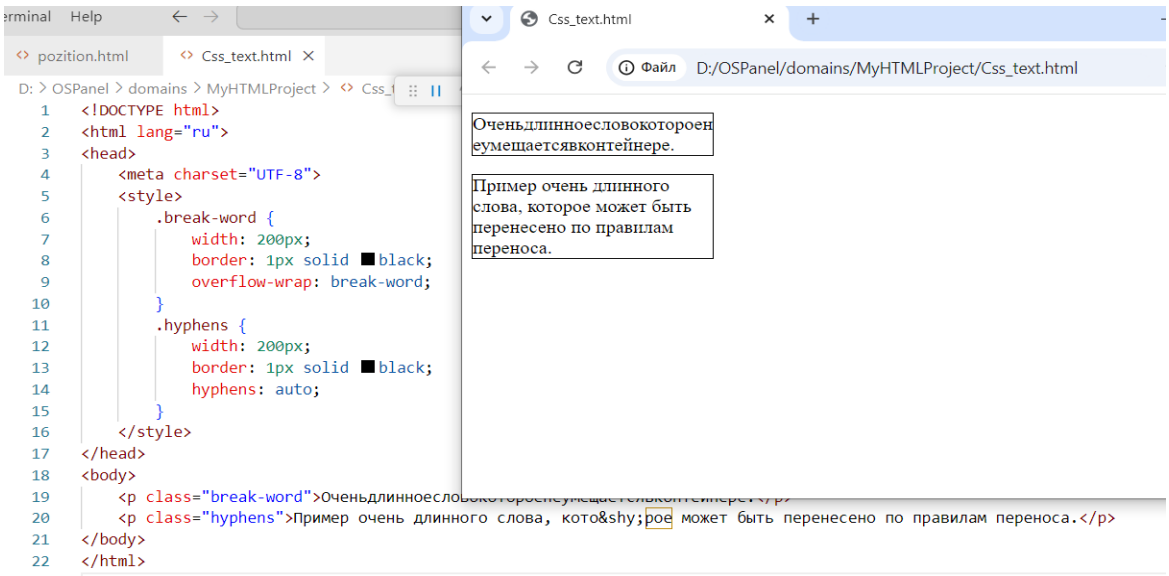


Рисунок 24 – Пример управления переносом слов слов с `overflow-wrap` и `hyphens`

Первый абзац (`break-word`) переносит длинное слово, которое не умещается в блоке, разрывая его.

Во втором абзаце (`hyphens`) длинное слово будет перенесено по правилам языка с использованием дефисов.

4. Выравнивание текста с `text-align`, рисунок 25

Первый абзац выровнен по левому краю.

Второй абзац выровнен по центру.

Третий абзац выровнен по правому краю.

Четвертый абзац выровнен по ширине, растягивая текст.

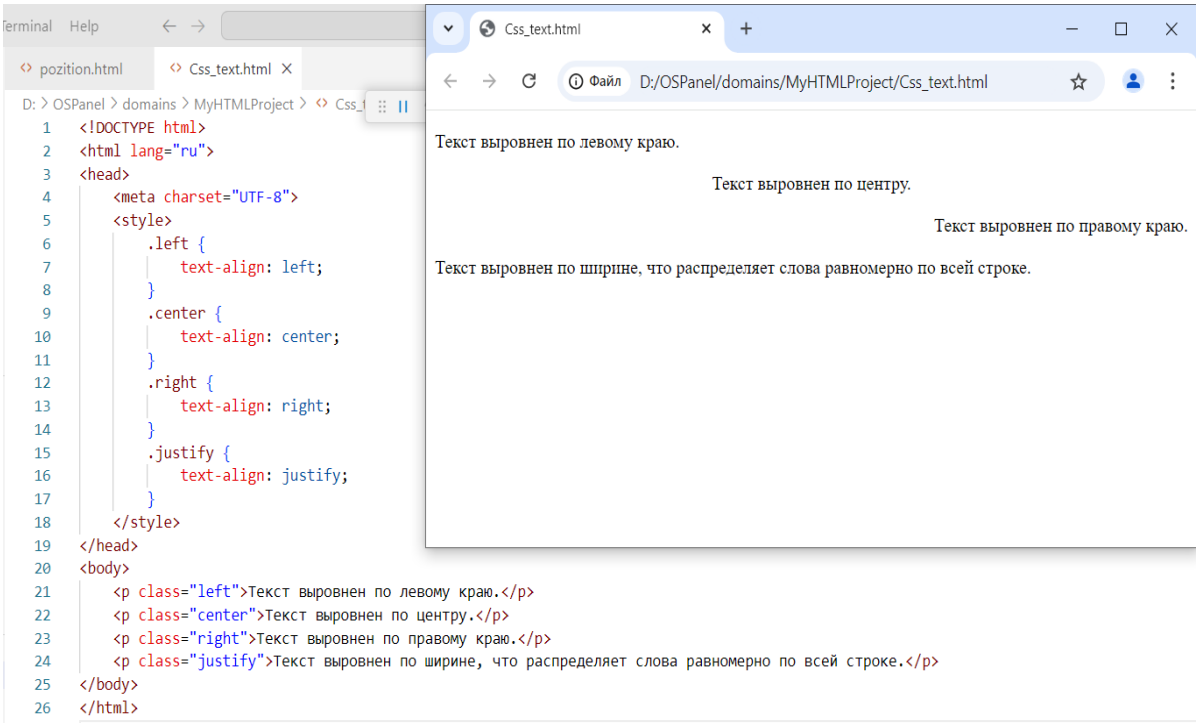


Рисунок 25 – Пример выравнивания текста с `text-align`

5. Интервалы между символами и строками, рисунок 26

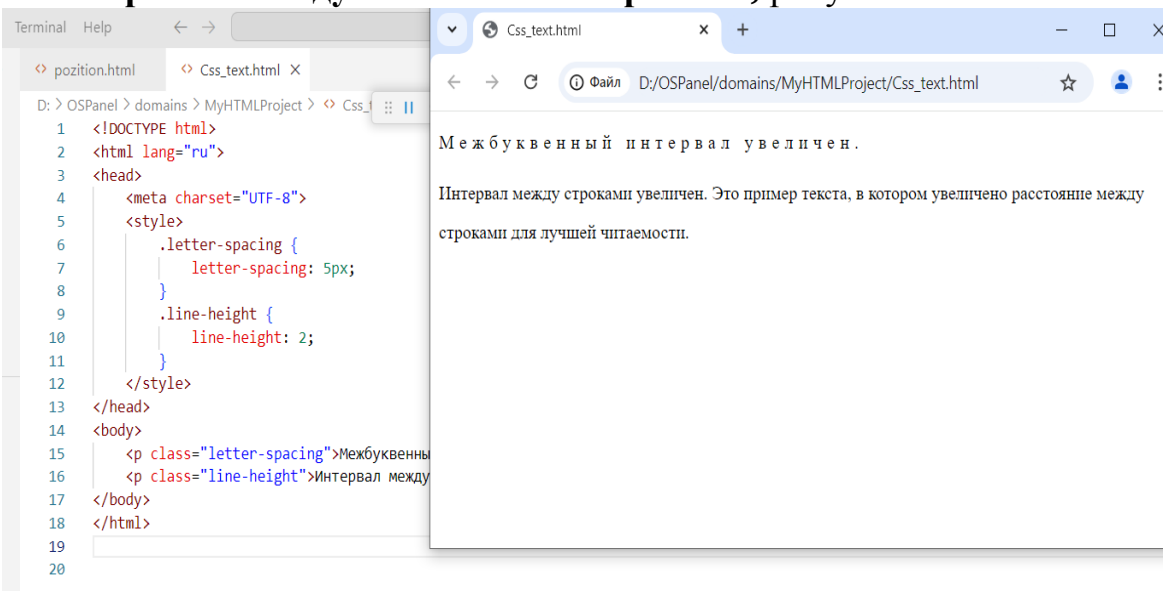


Рисунок 26 – Пример управления интервалами строками

Первый абзац имеет увеличенный интервал между символами (5 пикселей). Во втором абзаце увеличено расстояние между строками текста, что делает его более разреженным.

6. Отступ первой строки с `text-indent`, рисунок 27

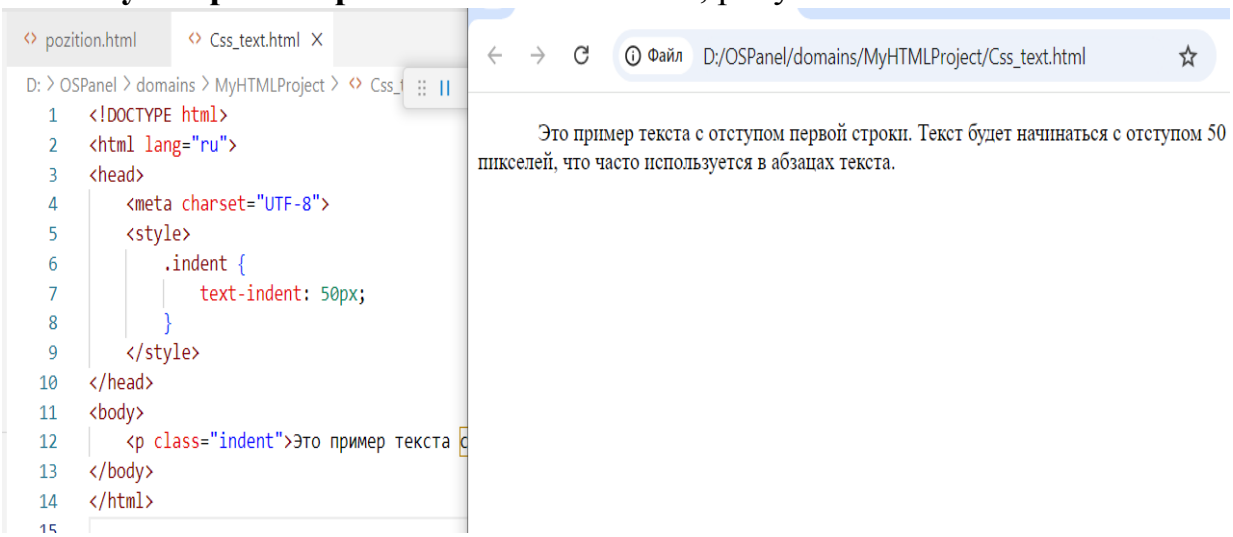


Рисунок 26 – Пример управления отступами строки с **с `text-indent`**

В этом абзаце первая строка будет начинаться с отступом 50 пикселей.

Индивидуальные задания лабораторной работы №13

Задание 1: Динамическая стилизация текста при наведении

Создайте страницу с тремя абзацами текста. Используя псевдоклассы и CSS-анимацию, сделайте следующее:

1. При наведении курсора на абзац, текст должен плавно трансформироваться в заглавные буквы (`text-transform: uppercase`) в течение 1 секунды.
2. При уходе курсора текст должен вернуться в исходный вид.

3. Текст должен также изменять цвет при наведении.

Подсказка: Используйте свойства `transition` и псевдокласс `:hover`.

Задание 2: Трёхуровневое выравнивание текста

Создайте страницу с тремя блоками текста, каждый из которых имеет различное выравнивание. Задача состоит в том, чтобы:

1. Первый блок текста был выровнен по ширине с добавлением отступов между словами.
2. Во втором блоке текста строки должны быть выровнены по центру, но каждая вторая строка (чётные строки) должна быть выровнена по правому краю.
3. В третьем блоке текста нужно использовать CSS-свойство `text-shadow` для создания эффекта лёгкой тени.

Задание 3: Перенос и ограничение длины текста

Создайте текстовый блок с длинными словами, но сделайте следующее:

1. Ограничьте ширину блока 300px.
2. При переполнении блока длинными словами они должны переноситься, но с ограничением на количество строк. Если текст выходит за пределы блока, он должен скрываться с добавлением троеточия (используйте `text-overflow: ellipsis`).
3. Для первого абзаца, который полностью помещается, примените свойство `line-clamp`, чтобы ограничить отображение только тремя строками.

Подсказка: Используйте комбинацию свойств `overflow`, `white-space`, `text-overflow`, и `-webkit-line-clamp`.

Задание 4: Многоязычный текст с разным направлением письма

Создайте страницу, которая содержит текст на нескольких языках, таких как английский, арабский и иврит. Задача:

1. Правильно отобразить текст на арабском и иврите, которые пишутся справа налево, используя свойства `direction` и `unicode-bidi`.
2. В тексте должны присутствовать как латинские, так и нелатинские символы, которые будут корректно смешаны в одном абзаце.

Задание 5: Адаптивная стилизация текста с помощью медиазапросов

Создайте адаптивную веб-страницу, где текст автоматически изменяет свой размер и межстрочный интервал в зависимости от ширины экрана:

1. Для экранов шириной менее 600px текст должен уменьшаться до 14px с межстрочным интервалом 1.2.
2. Для экранов шириной от 600px до 900px текст должен быть 16px с межстрочным интервалом 1.5.
3. Для экранов шириной более 900px текст должен увеличиваться до 18px с межстрочным интервалом 1.8.

Подсказка: Используйте медиазапросы для изменения стилей в зависимости от ширины экрана.

Задание 6: Комплексный макет с многоуровневой стилизацией текста

Создайте страницу, которая содержит многоуровневую структуру текста (заголовки, списки, абзацы). Стилизуйте текст так, чтобы:

1. Заголовки имели разные шрифты и размеры для каждого уровня (h1, h2, h3).
2. Списки имели стилизованные маркеры, а текст списков — отступы и межстрочные интервалы.
3. Используйте различные цвета для заголовков и текста в зависимости от уровня и важности информации.

Список литературы

1. Баранов, Р.Д. Практические аспекты разработки веб-ресурсов [Электронный ресурс]: учебное пособие/ Р.Д. Баранов, С.А. Иноземцева, А.А. Рябова. — Электрон. текстовые данные. — Саратов: Вузовское образование, 2018. — 121 c. — 978-5-4487-0263-1. — Режим доступа: http://www.iprbookshop.ru/75692.html
2. Лучанинов, Д.В. Основы разработки web-сайтов образовательного назначения [Электронный ресурс]: учебное пособие/ Д.В. Лучанинов. — Электрон. текстовые данные. — Саратов: Ай Пи Эр Медиа, 2018. — 105 c. — 978-5-4486-0174-3. — Режим доступа: http://www.iprbookshop.ru/70775.html
3. Торопова, О.А. Основы web-программирования. Технологии HTML, DHTML [Электронный ресурс]: учебное пособие/ О.А. Торопова, И.Ф. Сытник. — Электрон. текстовые данные. — Саратов: Саратовский государственный технический университет имени Ю.А. Гагарина, ЭБС АСВ, 2014. — 106 c. — 978-5-7433-2606-8. — Режим доступа: http://www.iprbookshop.ru/76493.html
4. Буренин, С.Н. Web-программирование и базы данных [Электронный ресурс]: учебный практикум/ С.Н. Буренин. — Электрон. текстовые данные. — М.: Московский гуманитарный университет, 2014. — 120 c. — 978-5-906768-17-9. — Режим доступа: http://www.iprbookshop.ru/39683.html
5. Зудилова, Т.В. Web-программирование HTML [Электронный ресурс]/ Т.В. Зудилова, М.Л. Буркова. — Электрон. текстовые данные. — СПб.: Университет ИТМО, 2012. — 70 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/65748.html
6. Зудилова, Т.В. Web-программирование JavaScript [Электронный ресурс]/ Т.В. Зудилова, М.Л. Буркова. — Электрон. текстовые данные. — СПб.: Университет ИТМО, 2012. — 68 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/65749.html
7. Мелькин, Н.В. Искусство продвижения сайта. Полный курс SEO [Электронный ресурс]: от идеи до первых клиентов/ Н.В. Мелькин, К.С. Горяев. — Электрон. текстовые данные. — М.: Инфра-Инженерия, 2017. — 268 c. — 978-5-9729-0139-5. — Режим доступа: http://www.iprbookshop.ru/68990.html
8. Шапошников, И. Web-сайт своими руками [Текст]: учеб. пособие/ Шапошников И.- СПб.: БХВ-Петербург, 2000.- 224 с.
9. Кожемякин, А. А. «HTML и CSS в примерах. Создание Web-страниц»/ А. А. Кожемякин.– Изд-во: Альтекс-А, 2014.– 416 с.
10. Шмитт, К. Создание WEB-страниц средствами CSS/ К. Шмит. – Издательство: КУДИЦ-Образ, 2003. – 368 с.

КОЧКАРОВ Ахмат Магомедович
СЕЛИМСУЛТАНОВА Рита Ильясовна

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ В ИНТЕРНЕТ

Лабораторный практикум
для обучающихся 3 курса направления подготовки
01.03.02 Прикладная математика и информатика

Корректор Джукаев У.М.
Редактор Джукаев У.М.

Сдано в набор 23.04.2025 г.
Формат 60x84/16
Бумага офсетная.
Печать офсетная.
Усл. печ. л. 5,34.
Заказ № 5084
Тираж 100 экз.

Оригинал-макет подготовлен
в Библиотечно-издательском центре СКГА
369000, г. Черкесск, ул. Ставропольская, 36