Geekbrains

Специальность: Frontend-программист Цифровые профессии

Тема дипломного проекта: «Инструкция по HTML и СSS»

ФИО студента: Дьячков Антон Геннадьевич

г. Северодвинск

2023

Содержание дипломного проекта

Введение 4

1. HTML 5

1.1 Основные понятия HTML 5

1.2 Работа с атрибутами 8

1.3 Работа с текстом 9

1.4 Работа со ссылками 11

1.5 Работа с изображениями 14

1.6 Работа с таблицами 15

1.7 Работа со списками 18

1.8 Полезные ссылки 19

2. CSS 20

2.1 Основы CSS 20

2.2 Работа с текстом

2.3 Блочные и строчные элементы

2.4 Работа с полями

2.5 Позиционирование

2.6 Работа с display: flex

2.7 Работа с CSS Grid

3. Заключение

4. Список использованной литературы и ресурсов

5. Приложения

6. Презентация к дипломному проекту

Введение

Данный дипломный проект - небольшая практическая работа, в которой предполагается использовать все полученные сведения и закрепить теорию практикой верстки небольшого сайта. Защита проекта является обязательной итоговой аттестацией, после прохождения планируемых технологических специализаций (Frontend-программист. Цифровые профессии) для получения итогового документа об обучении после его окончания.

Тема проекта была выбрана автором самостоятельно с целью объединить знания о самых основных и минимально необходимых инструментах верстки сайтов, полученных на курсе, интересы и некоторый практический опыт верстки сайтов с соблюдением следующих условий: тема актуальна, реализуема и практична.

Используемые инструменты - HTML, CSS, Visual Studio Code.

Проект представляет из себя сжатую инструкцию по использованию основных инструментов, необходимых для верстки сайтов – HTML и CSS для непосредственной верстки сайта.

Цель выбранной темы - дать самое общее представление о вышеуказанных инструментах и возможностях использования в практической работе имеющихся сведений для верстки сайтов.

Задачи данного дипломного проекта:

Во-первых, рассмотреть основные понятия HTML, возможности работы с атрибутами, с текстом, со ссылками, с изображениями, с таблицами, списками и формами.

Во-вторых, рассмотреть основные понятия CSS, возможности работы с текстом, блочными и строчными элементами, с полями, позиционирование и работа с «display: flex» и CSS Grid.

Если рассматривать каждый инструмент обстоятельно и подробно, то общий объем данного проекта выйдет далеко за свои рамки. Для более глубокого и подробного понимания предлагается использовать список литературы и источников, приведенных в Разделе 4.

1. HTML

1.1. Понятия HTML

HTML (HyperText Markup Language) - это основной или стандартный язык разметки документов сети INTERNET. В данном разделе используем правила языка HTML, описание стандартной структуры страницы HTML, отношения в общей структуре документа между его элементами. Данный язык «читается» браузерами, а затем полученная и обработанная ими информация выдается на экран монитора ПК, ноутбука, планшета или смартфона.

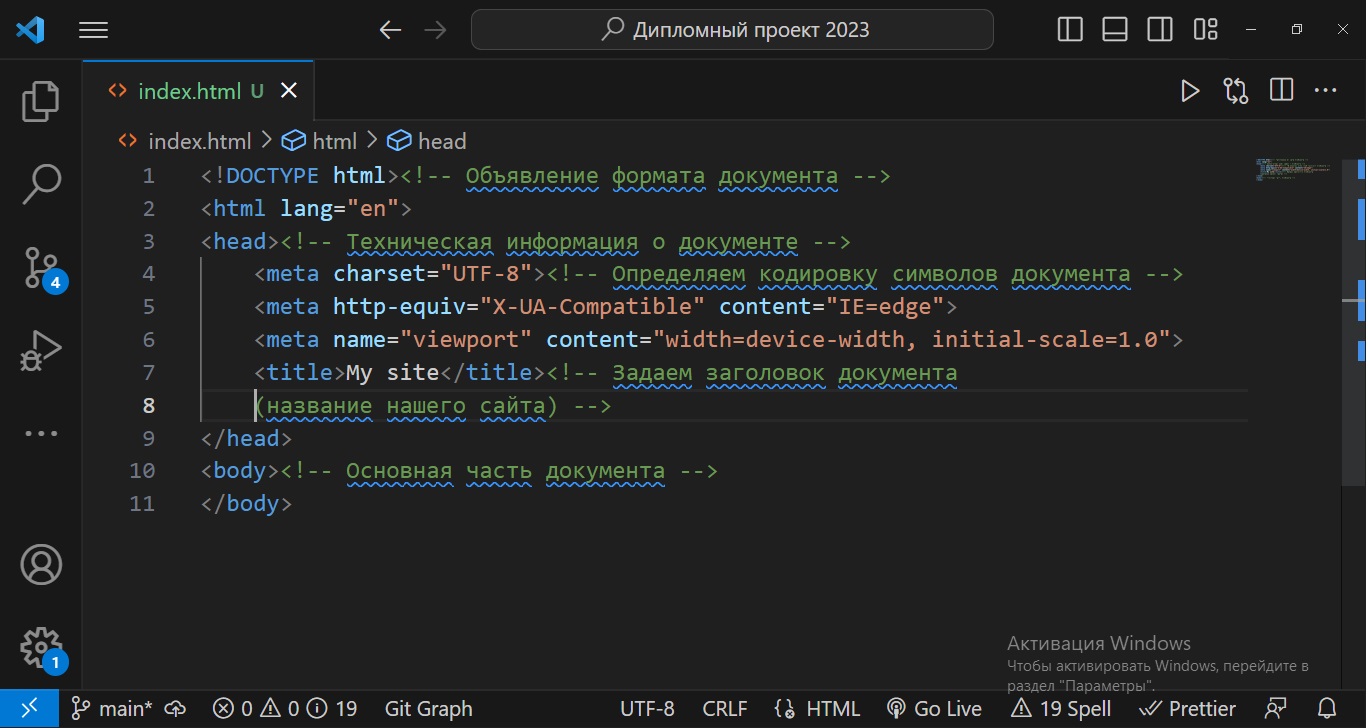
Гипертекст – система текстовых страниц, соединенных между собой ссылками, а сам документ в формате HTML – просто текстовый документ, который можно создать в любом обычном текстовом редакторе (Блокнот, Word) или в редакторе кода (Notepad++, Visual Studio Code и т.п.). Документ формата HTML имеет расширение «.html» в своем названии. Создав на своем ПК, ноутбуке, или планшете файл с таким расширением, можно открыть его через любой веб браузер, просто кликнув на него как на обычный файл на своем устройстве.

В рамках данного дипломного проекта для написания кода будет использоваться редактор кода VSC (Visual Studio Code) как наиболее подходящий по своим возможностям.

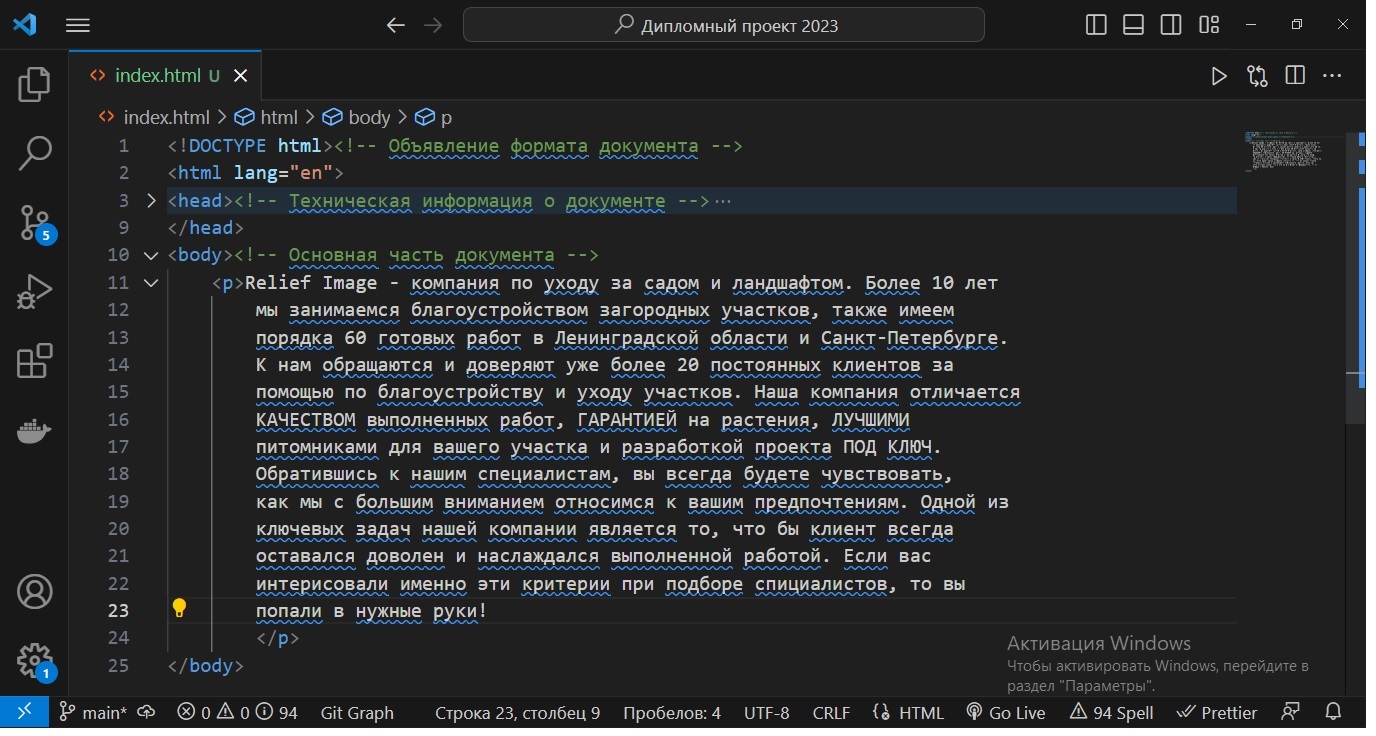
Создадим файл с рабочим названием «index.html» (название файла до наименования его расширения может быть любым) в данном редакторе кода и сформируем самую базовую структуру html-документа нашего нового сайта как представлено на Рисунке 1.1.

Любая запись в представленном коде, начинающая с символа «<!—» и заканчивающаяся на «-->», является комментарием и не отображается непосредственно на странице веб сайта и служит пояснением для самого программиста или других разработчиков.

Созданный нами HTML-документ состоит из дерева множества HTML-элементов и текста или иным другим содержимым. А каждый элемент обозначается в исходном документе начальным (открывающим) и конечным (закрывающим) тегами (с исключениями, когда закрывающийся тег не требуется).

Рисунок 1.1

На Рисунке 1.2 начальный тег «<p>» показывает, где начинается элемент, конечный «</p>» - где заканчивается. Браузер при этом выведет только все содержимое между двумя тегами.

Рисунок 1.2

На Рисунке 1.2 содержимым является текст. В качестве контента в зависимости от наименования тэга будет также название компании (продукта), заголовок статьи или раздела сайта, картинка, видео, аудиозапись и т.д.

Именно элементы, находящиеся внутри элемента «<html>», образуют дерево (модель) документа. При этом сам элемент «<html>» является корневым элементом.

Как видно из представленных рисунков почти все элементы можно вкладывать друг в друга и чаще всего именно такое вкладывание необходимо. Браузер просматривает («читает») код документа, выстраивая его структуру и отображая ее в соответствии со инструкциями стилей, включенными в этот файл (т.е. CSS, который будет рассмотрен позднее). Если разметка верная, то в окне браузера будет отображена страница сайта, содержащая HTML-элементы – текст, заголовки, таблицы, изображения и т.д.

Процесс «чтения» начинается прежде, чем веб-страница полностью загружена в браузер. Браузеры интерпретируют документы HTML последовательно, с самого начала, обрабатывая CSS и соотнося таблицы стилей с элементами самой страницы. Любой HTML-документ состоит из двух разделов: заголовок (содержимое элемента <head>) и содержательная часть (содержимое элемента <body>).

Взаимодействие всех элементов между собой в нашей веб-странице, происходит в отношениях между элементами, представленных в Таблице 1.1.

Таблица 1.1

|  |  |  |
| --- | --- | --- |
| Наименование  элемента | Описание взаимоотношений | Пример  представления |
| Предок | Элемент, включающий в себя другие элементы | «<html>» для «<body>» |
| Потомок | Элемент, сам находящийся внутри одного или более элементов | «<body>» для «<html>» |
| Родитель | Элемент, связанный с другими элементами более низкого уровня, и находящийся в дереве документа выше | «<html>» для «<head>» и «<body>» |
| Дочерний элемент | Элемент, подчиненный другому элементу более высокого уровня | «<р>» для «<body>» |
| Сестринский элемент | Элемент, имеющий общего родителя с рассматриваемым элементом (то есть элементы одного уровня | «<head>» и «<body>» для «<html>» |

Таким образом кроме элемента «html» в нашем (и любом другом) HTML-документе обязательно присутствуют:

- «head» («<head>…</head>») – здесь расположена техническая информация о нашей странице, и она не отображается в окне браузера, но указывает браузеру, как следует интерпретировать страницу;

- «title» («<title>…</title>») – здесь отображается текст в строке заголовка веб-браузера и содержится максимально полное описание содержимого веб-страницы т.е. по сути это название страницы сайта, на котором мы находимся (каждая страница сайта должна иметь собственное название, которое и отображается в данных тегах);

- «meta» («<meta>») – здесь содержится описание содержимого страницы и ключевые слова для поисковых машин, автора документа и прочие свойства метаданных и при этом может содержаться несколько подобных элементов, поскольку в зависимости от используемых атрибутов они несут различную информацию т.е. может иметь множество различных атрибутов;

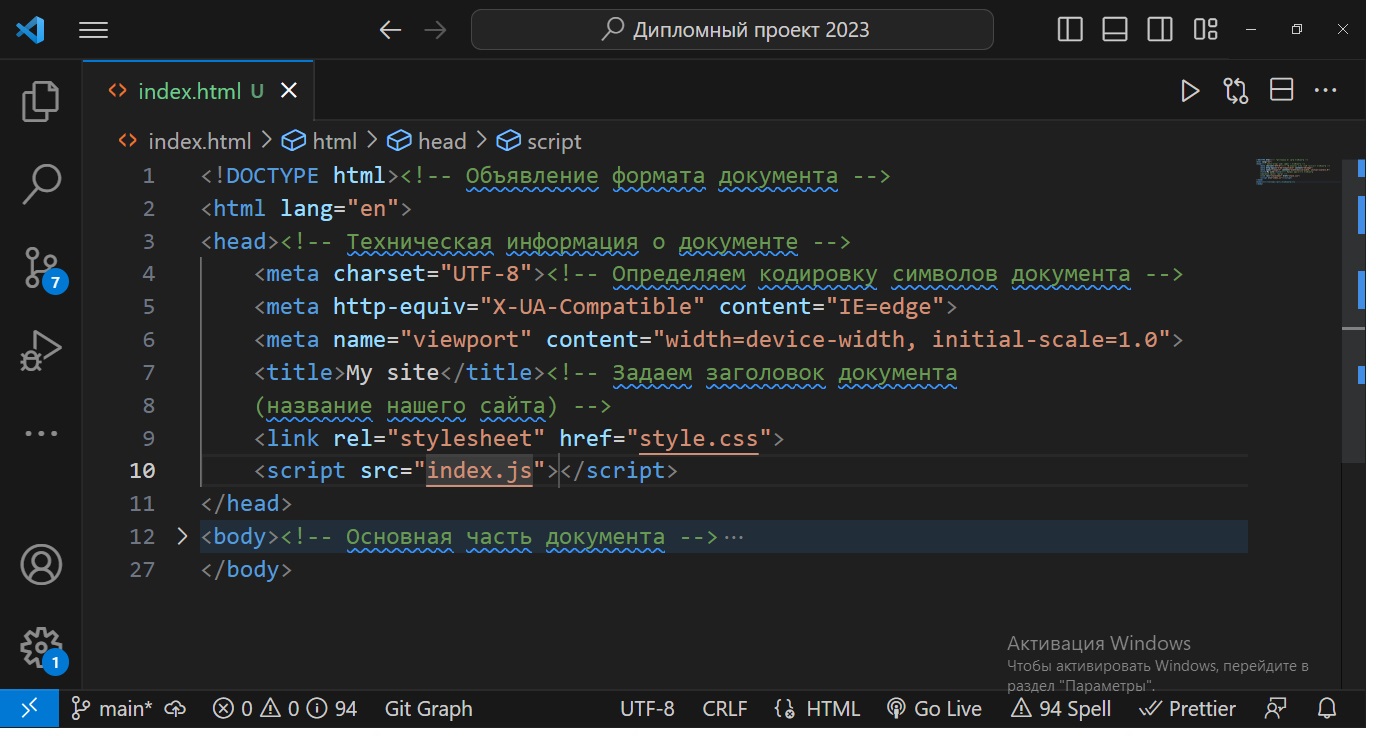
- «style («<style>…</style>») - здесь находятся стили, которые будут использоваться на странице, но в HTML-документе лучше всего использовать все же внешний подключаемый файл CSS (с помощью элемента, указанного ниже);

- «link» («<link>») - подключение к html-документу других исполняемых файлов – файл со стилями style.css, где href является основным атрибутом элемента, где в качестве значения выступает путь к файлу со стилями;

- «script» («<script>») - присоединяет к документу различные сценарии, написанные на языке программирования. Текст сценария может располагаться по аналогии с CSS либо внутри этого элемента, либо во внешнем файле;

- «body» («<body>…</body>») - здесь располагается все основное содержимое документа или его контент, ради чего и создается страница.

Все вышеперечисленные элементы представлены на Рисунке 1.3

Рисунок 1.3

В «body» все содержимое разделяется ка правило на «header» («шапка» сайта, в котором находятся элементы навигации по самому сайту, поисковый сервис, контент с названием организации и логотипа), «footer» («подвал» сайта, в котором могут располагаться контактные данные организации) и основное, видимое в браузере, содержимое.

Это лишь часть основных элементов стандартной html-страницы.

Благодаря использованию в настоящем дипломном проекте Visual Studio Code можно не прописывать все элементы в разделе «head», а развернуть автоматически («!» + Tab) и начать проектировать наш сайт в разделе «body».

1.2 Работа с атрибутами

HTML-элементы на нашем сайте могут (и должны) иметь атрибуты (применяемые вообще для всех HTML-элементов, и собственные). Атрибуты помогают изменять свойства и поведение элемента, для которого они заданы.

Многие из них повторяются в различных тегах («title» - добавляет всплывающую подсказку и используется в большинстве HTML тегов).

HTML-атрибуты — специальные слова, управляющие поведением элемента, к которому они прописаны. Они добавляют дополнительную функциональность, либо меняют поведение элемента по умолчанию. Атрибуты элемента выражаются внутри начального тега элемента.

Некоторые из самых необходимых атрибутов, которые будут нужны в нашем проекте:

- «class» («class="название класса"») - список классов элемента с учетом регистра (позволяют CSS и Javascript выбирать и получать доступ к отдельным элементам с помощью селекторов классов или функций);

- «id» («id="название идентификатора"») - уникальный идентификатор элемента для идентификации элемента;

- «lang» («lang="en "») - указывает основной язык для содержимого элемента и для любого из атрибутов элемента, содержащих текст;

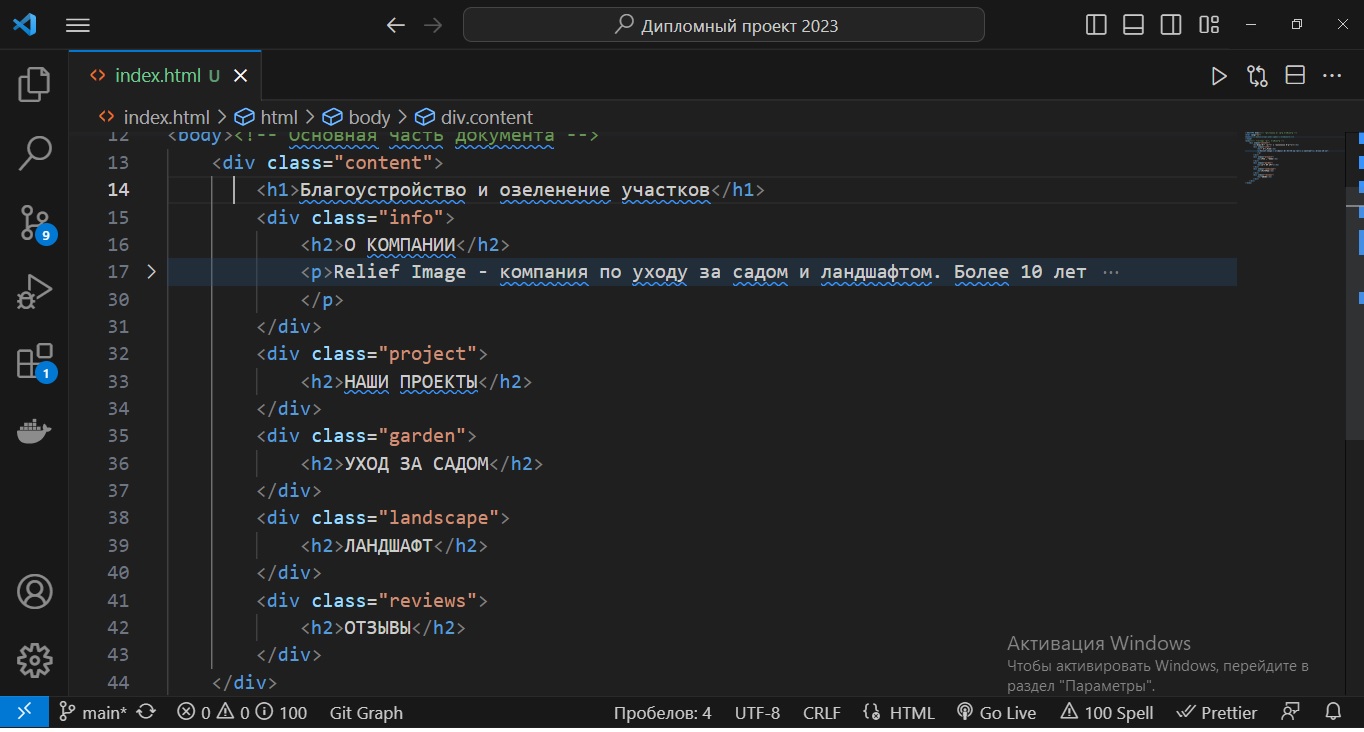
- «style» («style="color: blue; background: transparent"») - содержит объявления стилей CSS, применяемые к конкретному элементу (имеется также тег style, который задает стиль для всей html страницы);

- «title» («title="Hypertext Transport Protocol"») - содержит дополнительную информацию об элементе, задавая всплывающую подсказку для страницы.

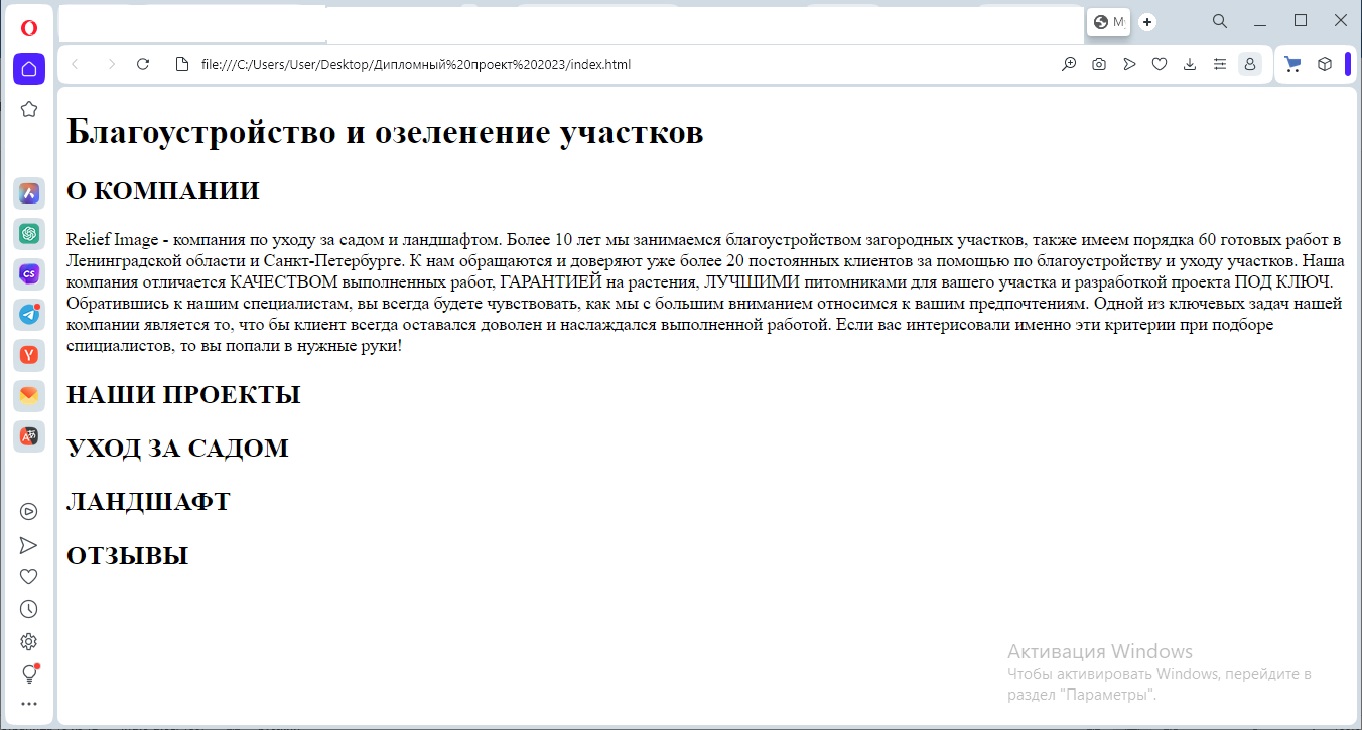
1.3 Работа с текстом

Текст является одним из основных типов контента (содержимого) сайта и сделать его ярче и лучше для восприятия можно за счет добавления к нему специальных HTML тегов.

В HTML можно создавать заголовки за счет тегов «h1», «h2», «h3», «h4», «h5», «h6». Они дают возможность создать заголовки разных размеров (от большего к меньшему). Добавим к нашему проекту один элемент «h1» (должен быть один на странице) и несколько элементов «h2» (элементов «h2» - «h6» может быть сколько необходимо) как показано на Рисунке 1.4.

Рисунок 1.4

Визуальное же отображение представленного кода непосредственно в браузере выглядит как на Рисунке 1.5.

Рисунок 1.5

Заголовки упорядочивают текстовую часть страницы, формируя его визуальную структуру. Элементы «h1» - «h6» должны использоваться только для выделения заголовков нового раздела или подраздела.

Основным тегом для разделения текста является тег абзаца «<p>». В него можно поместить любое количество текста и все будет выделено в формате абзаца с отступами.

Дополнительно в HTML можно делать текст более разнообразным при помощи специальных тегов (жирный текст при помощи тега «<b>», курсивом при помощи тега «<em>»). Но в рамках данного проекта и в целом исходя из сложившейся практики такие элементы при форматировании текста несут лишь смысловую нагрузку и задают для текста, заключенного внутрь, стилевое оформление, выделяяя текст жирным начертанием или отображают его шрифтом другого семейства (свойство font-family).

Все HTML-элементы для работы с текстом представлены в Таблице 1.2.

Таблица 1.2

|  |  |
| --- | --- |
| Наименование элемента | Описание элемента |
| Элементы для форматирования текста | |
| <b> | Задаётся полужирное начертания шрифта текста. Выделяется текст без акцента на его важность |
| <em> | Отображается шрифт текста курсивом, придавая тексту значимость |
| <i> | Отображается шрифт курсивом |
| <small> | Уменьшается размер шрифта на единицу по отношению к обычному тексту |
| <strong> | Задаётся полужирное начертание шрифта, относится к элементам логической разметки, указывая браузеру на важность текста |
| <sub> | Используется для создания нижних индексов, сдвигая текст ниже уровня строки и уменьшая его размер |
| <sup> | Используется для создания степеней, сдвигая текст выше уровня строки и уменьшая его размер |
| <ins> | Выделяется текст в новой версии документа, подчёркивая его |
| <del> | Перечёркивается текст. Используется для выделения текста, удаленного из документа |
| <mark> | Применяется для выделения фрагментов текста в справочных целях, окрашивая блок символов желтым цветом |
| Абзацы, средства переноса текста | |
| <p> | Разбивается текст на отдельные абзацы, отделяя друг от друга пустой строкой |
| <br> | Переносится текст на следующую строку, создавая разрыв строки |
| <hr> | Используется для разделения контента на веб-странице. Отображается в виде горизонтальной линии |

Из всего представленного списка элементов для работы с текстом понадобился лишь элемент «<p>».

1.4 Работа со ссылками

Ссылка - перенаправление пользователя сайта не только на новые страницы, но и на определенное место на текущей странице. Ссылка будет являться активной и отображаться в формате ссылки только при условии прописанного атрибута «href». Если не добавить значения в данный атрибут, то при клике на ссылку страница будет перезагружаться.

Ссылки создаются с помощью элемента «<a>». Внутрь помещается текст, который будет отображаться на веб-странице. Текст ссылки отображается в браузере с подчёркиванием, цвет шрифта - синий, при наведении на ссылку курсор мыши меняет вид.

Обязательным параметром элемента «<a>» является уже упомянутый атрибут href, который задает URl-адрес веб-страницы.

Ссылка состоит из двух частей - указателя («Мы в социальной сети ВК») и адресной части (https://vk.com).

Ссылки бывают:

- относительные;

- абсолютные;

- якоря.

Относительные ссылки используются для перемещения внутри документа или сайта (по страницам сайта).

Абсолютные ссылки используются для перехода на страницы внешнего сайта в сети Интернет. Для этого в значении атрибута href нужно указать полный путь до той страницы, на которую мы хотим перейти, включая и тип протокола.

Когда же в ссылке указывается только имя файла, браузер предполагает, что файл находится в той же папке, что и документ, содержащий ссылку. То это ссылка относительная. Нередко веб-сайты содержат множество документов, которые размещают в отдельные папки, чтобы ими было легче управлять. Создадим отдельный файл Чтобы создать ссылку на файл, находящийся вне папки, содержащей текущий документ, необходимо указать расположение файла или путь: абсолютный или относительный.

Абсолютный путь указывает точное местоположение файла в пределах всей структуры папок на компьютере (сервере). Он даёт доступ к файлу со сторонних ресурсов и содержит различные компоненты (протокол, домен, папка, файл).

Когда необходимо сослаться на страницу на другом сайте, то можно использовать только полный абсолютный путь.

Если файл находится в корневой папке, то путь к нему - http://mysite.ru/index.html.

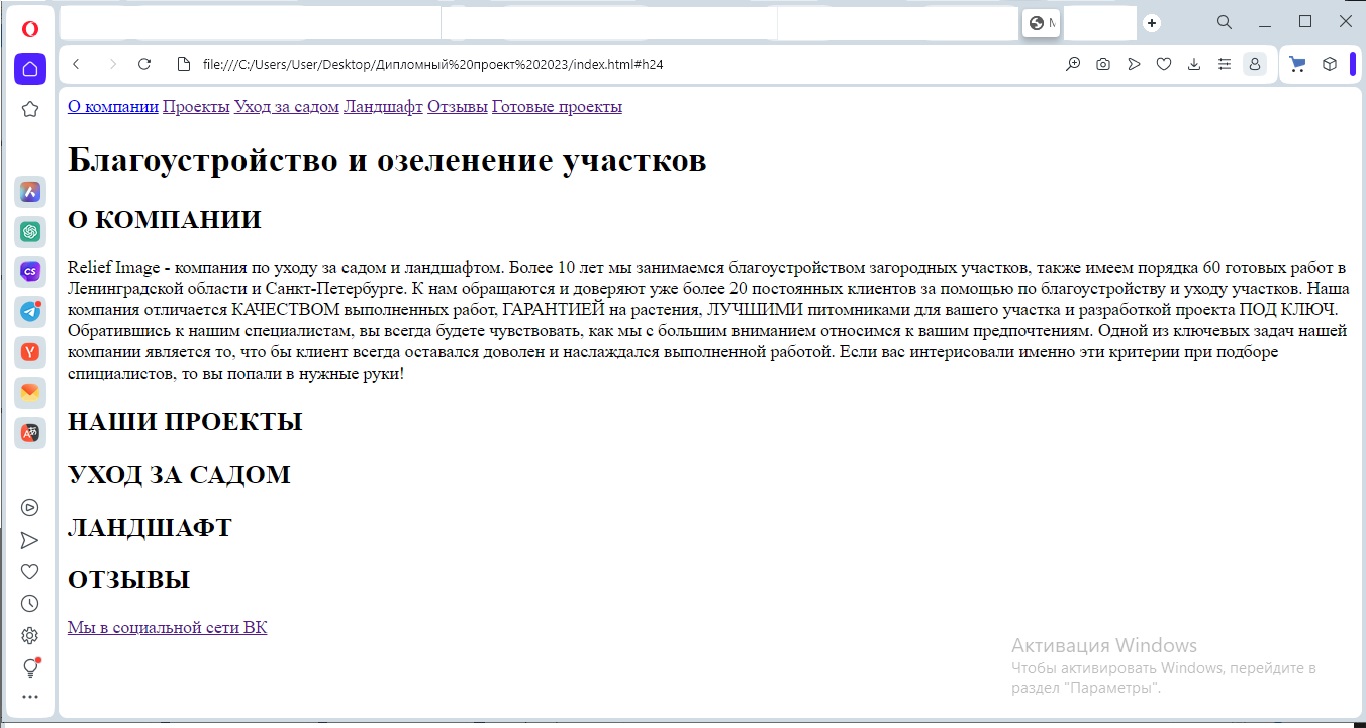
Обычно в качестве индексного файла выступает документ с именем index.html.

Относительный путь описывает путь к указанному документу относительно текущего. Он определяется с учётом местоположения веб-страницы, на которой находится ссылка. Относительные ссылки используются при создании ссылок на другие документы на одном и том же сайте.

Главное отличие относительного пути от абсолютного в том, что относительный путь не содержит имени корневой папки и родительских папок, что делает адрес короче, и в случае переезда с одного домена на другой не нужно прописывать новый абсолютный адрес. Но если сторонний ресурс будет ссылаться на файлы (изображения, видео, другой контент) с относительными адресами, то они не будут отображаться на другом сайте.

Якорь - это внутренняя ссылка, создающая переход на различные разделы текущей веб-страницы, позволяя быстро перемещаться между разделами. Это очень удобно в случае, когда на странице слишком много текста. Внутренние ссылки также создаются при помощи элемента <a> с разницей в том, что атрибут href содержит имя указателя, а не URl-адрес и перед именем указателя всегда ставится знак «#» как показано на Рисунке 1.7, где также представлены ссылки относительная и абсолютная. При нажатии на любую созданную ссылку можно сразу перейти к нужному разделу сайта или другой страницы сайта, или сторонний ресурс. Визуально страница сайта браузером отображается как представлено на Рисунке 1.8.

Рисунок 1.7.

Рисунок 1.8

1.5. Работа с изображениями

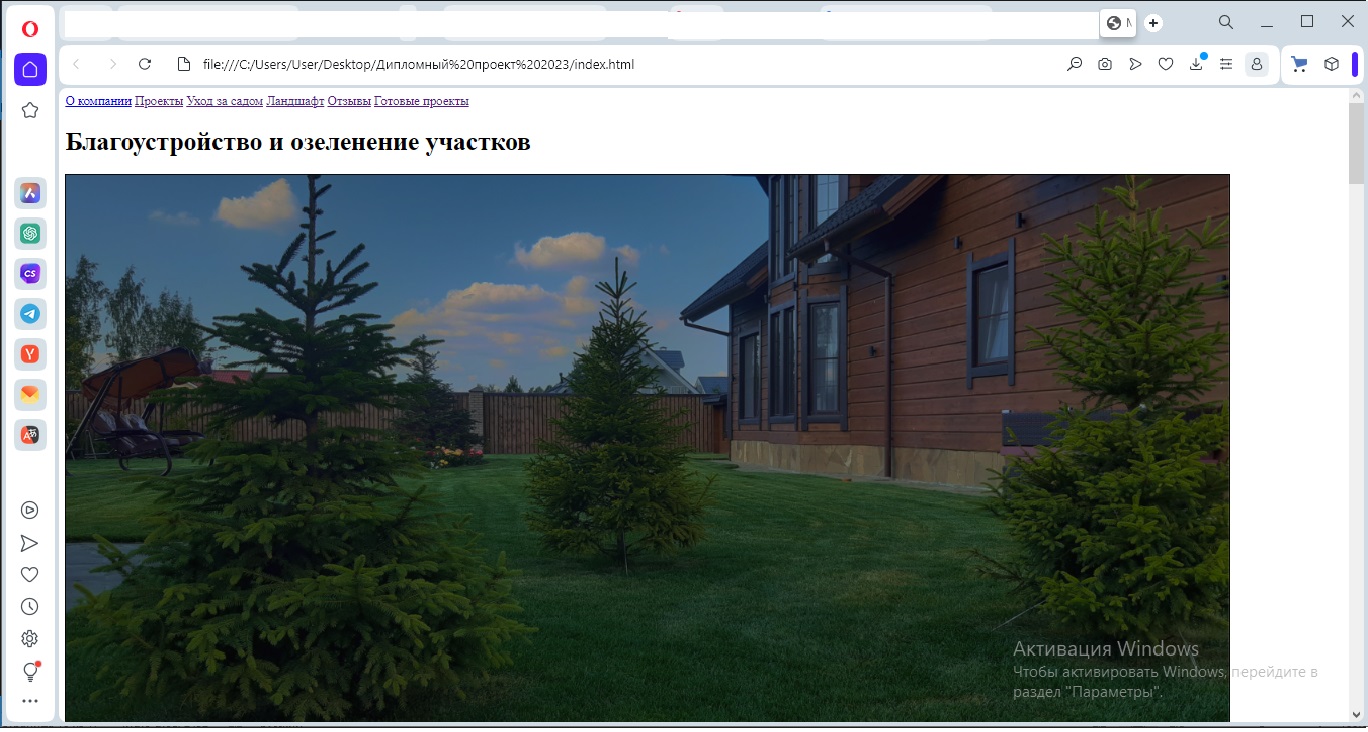
Изображения добавляются на страницы сайта с помощью одинарного тега «<img>» (не требует закрывающего тега).

Вообще использование какой-либо графики делает веб-страницы значительно привлекательнее и позволяет лучше передать суть и содержание веб-документа. Добавим в код нашего проекта несколько элементов «<img>» как представлено на Рисунке 1.9.

Рисунок 1.9

Как видно для тега «<img>» доступны основные атрибуты «src» (задает путь к изображению либо полный к картинке, либо относительный, что находится на сервере или же в одной папке с файлом index.html.) и «alt» (добавляется альтернативный текст для изображения и выводится на месте появления изображения до его загрузки или при отключенной графике). Все изображения любых форматов необходимо размещать в отдельной папке файлов проекта (например, создадим папку img) и уже с помощью атрибута «src» элемента «img» добавлять в код.

Внешний вид веб-страницы при обработке браузером представлен на Рисунке 1.10.

 Рисунок 1.10

Кроме вышеперечисленных используются и другие атрибуты для задания высоты изображения, его размера и ширины.

Возможные форматы изображений для их загрузки на сайт – JPEG (подходит для больших изображений, фотографий), GIF (анимация), SVG (походит для изображений – иконок социальных сетей, почтовых сервисов и других мелких изображений) и многие другие.

1.6. Работа с таблицами

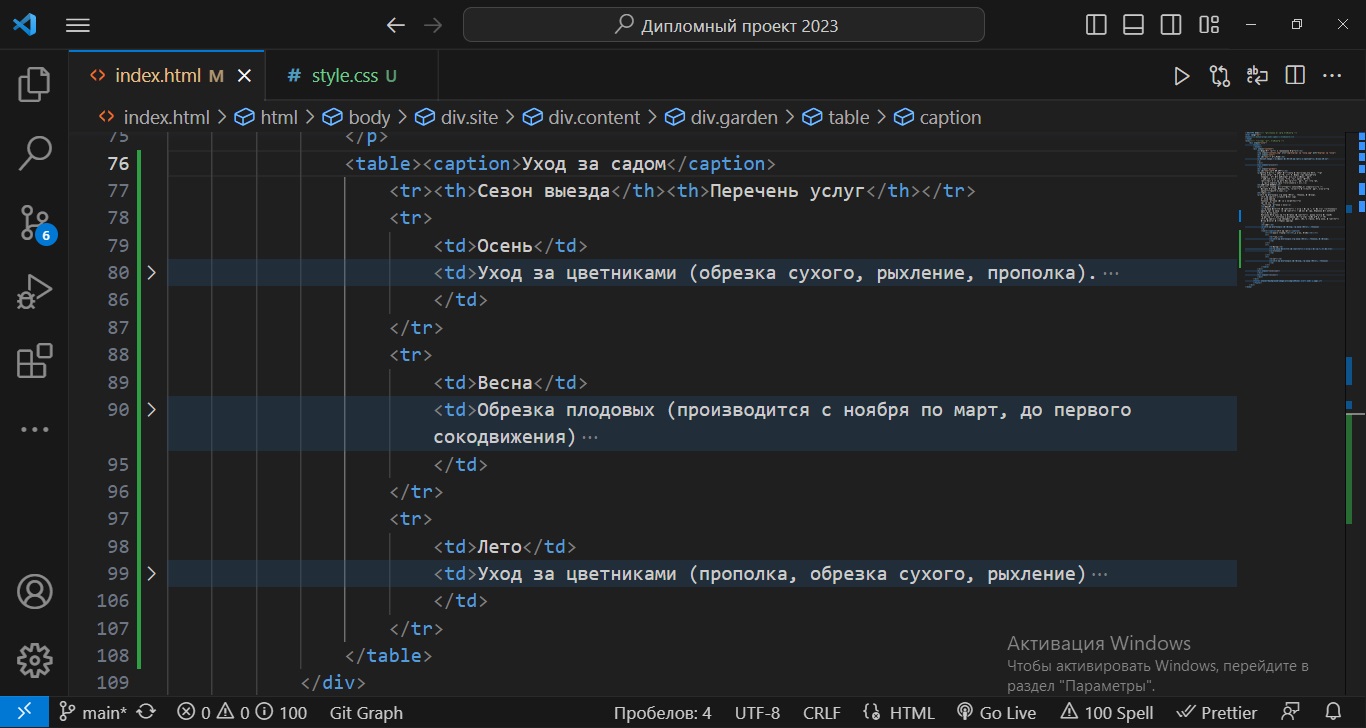
Таблицы упорядочивают и выводят на экран данные с помощью строк или столбцов и состоят из ячеек, образующихся при пересечении строк и столбцов.

Ячейки таблиц могут содержать любые элементы, такие как заголовки, списки, текст, изображения, элементы форм, а также другие таблицы.

К каждой таблице можно добавить связанный с ней заголовок, расположив его перед таблицей или после неё.

Таблицы больше не используются для вёрстки веб-страниц и компоновки отдельных элементов поскольку такой приём не обеспечивает гибкость структуры и адаптивность сайта, значительно увеличивая HTML-разметку. Однако в данном дипломном проекте используем элементы для создания таблицы в качестве примера.

Создадим таблицу в разделе «Уход за садом» с помощью элемента «<table></table>», который является контейнером для элементов таблицы и все элементы должны находиться внутри него. Код представлен на Рисунке 1.11.

Рисунок 1.11

По умолчанию таблица и ячейки не имеют видимых границ, и они задаются с помощью свойств CSS в файле style.css (в данном проекте заданы стили для элементов «th» и «td»).

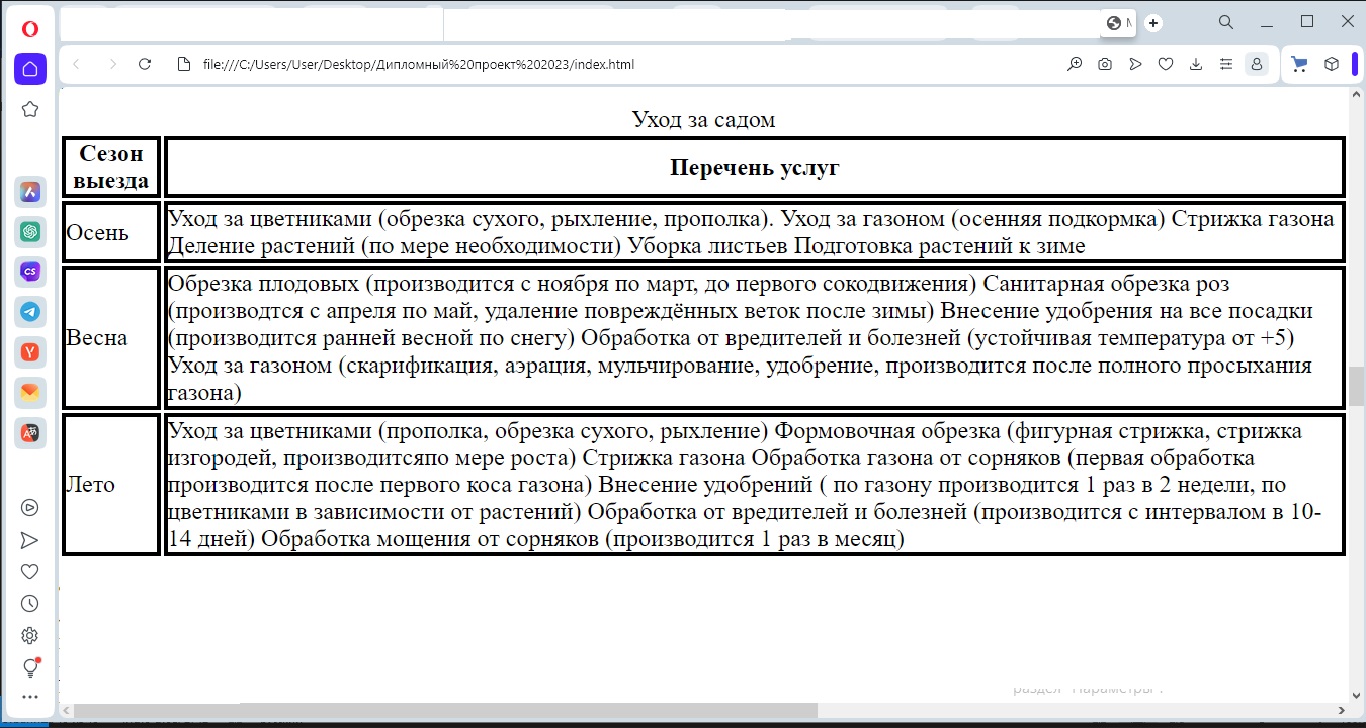
Строки или ряды таблицы создаются с помощью элемента «<tr>». Количество горизонтальных строк таблицы определяется количеством элементов «<tr></tr>».

Элемент «<th>» создаёт заголовок столбца - специальную ячейку, текст в которой выделяется полужирным. Количество ячеек заголовка определяется количеством элементов «<th></th>». Для элемента доступны атрибуты «colspan» (устанавливает число ячеек, которые должны быть объединены по горизонтали), «rowspan» (устанавливает число ячеек, которые должны быть объединены по вертикали), «headers» (позволяет связать ячейки таблицы с заголовками).

Элемент «<td>» создаёт ячейки таблицы, внутрь которых помещаются данные таблицы. Элементы «<td></td>», расположенные в одном ряду, определяют количество ячеек в строке таблицы. Количество пар ячеек «<td>» должно быть равно количеству пар ячеек «<th>». Для элемента также доступны вышеназванные атрибуты «colspan», «rowspan», headers.

Элемент <caption> создает подпись таблицы. Добавляется непосредственно после тега <table>, вне строки или ячейки.

Отображение кода браузером представлено на Рисунке 1.12.

Рисунок 1.12

1.7. Работа со списками

В проекте сайта в разделе «Уход за садом» имеется перечень услуг в зависимости от сезона. Ранее для его отображения использовались варианты отображения как текста, как таблицы, но лучшим решением для привлекательного внешнего вида будут использоваться элементы списков.

Для создания списков используются теги «ol» и «ul», которые создают пронумерованный (каждый элемент отмечается цифрой) и маркированный (каждый элемент отмечается маркером) список соответственно. Для создания конкретного пункта в списке потребуется использовать тег «li». Если нужно вписать один список внутрь другого, то необходимо использовать следующую конструкцию, представленную на Рисунке 1.13.

Рисунок 1.13

Каждый список представляет собой контейнер, внутри которого располагаются элементы списка или пары термин-определение.

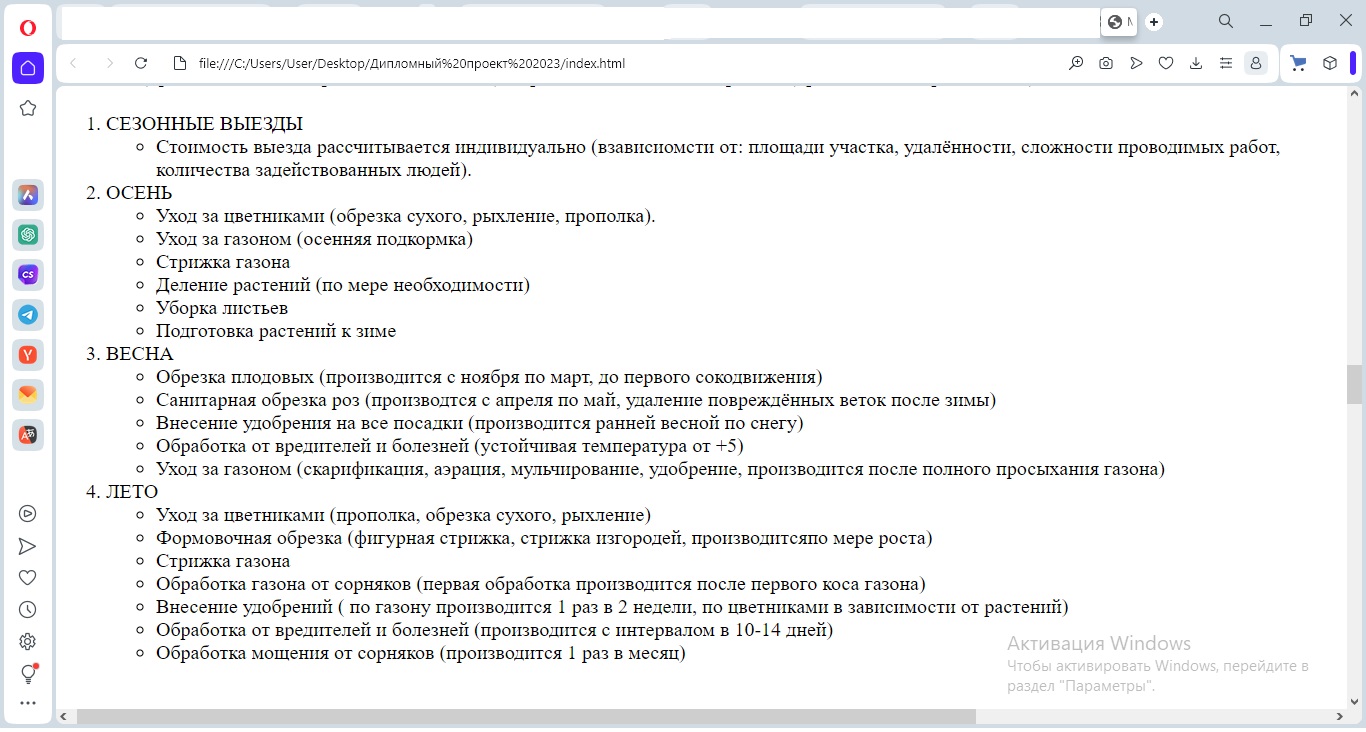
С помощью различных атрибутов можно задать различные виды отображения списков:

- «reversed» (задает отображение списка в обратном порядке);

- «start» (задает начальное значение, от которого пойдет отсчет нумерации);

- «type» (задает вид маркера для использования в списке (в виде букв или цифр).

Визуальное отображение созданного кода браузером представлено на Рисунке 1.14

Рисунок 1.14

2. CSS

2.1 Основные понятия CSS

CSS (каскадные таблицы стилей) – формальный язык описания внешнего вида документа, написанного с использованием языка разметки (HTML). Связка двух технологий HTML/CSS помогает создать внешний вид нашего сайта, HTML это контент нашего сайта, CSS это стили для данного контента. В данном разделе рассмотрим основные понятия CSS и селекторов.

CSS (Cascading Style Sheets) - язык таблиц стилей, позволяющий прикреплять стиль (шрифты, цвет и др.) к элементам веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML, но также могут быть применены к любому виду XML-документа, в том числе XML, SVG и XUL.

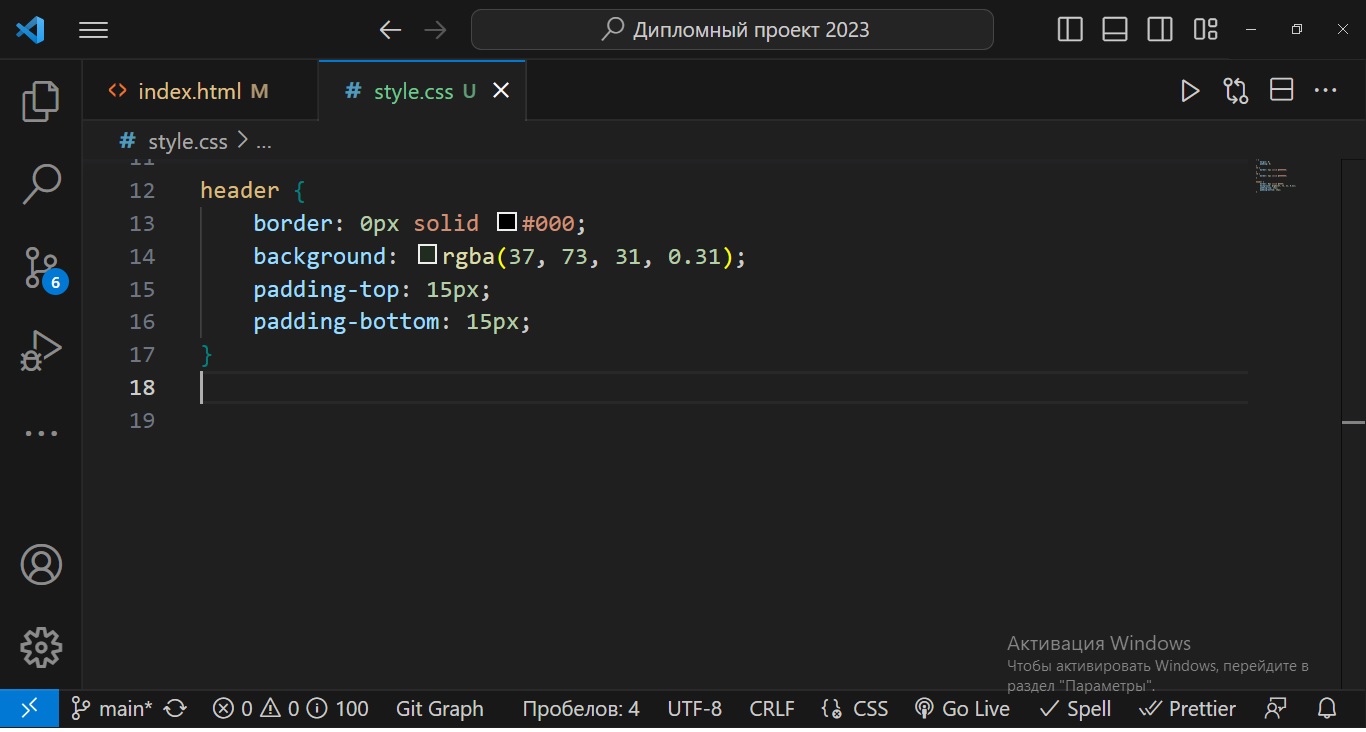
Отделяя стиль представления документов от содержимого документов, CSS упрощает создание веб-страниц и обслуживание сайтов.

Чистый HTML-документ выглядит однообразно без использования стилей.

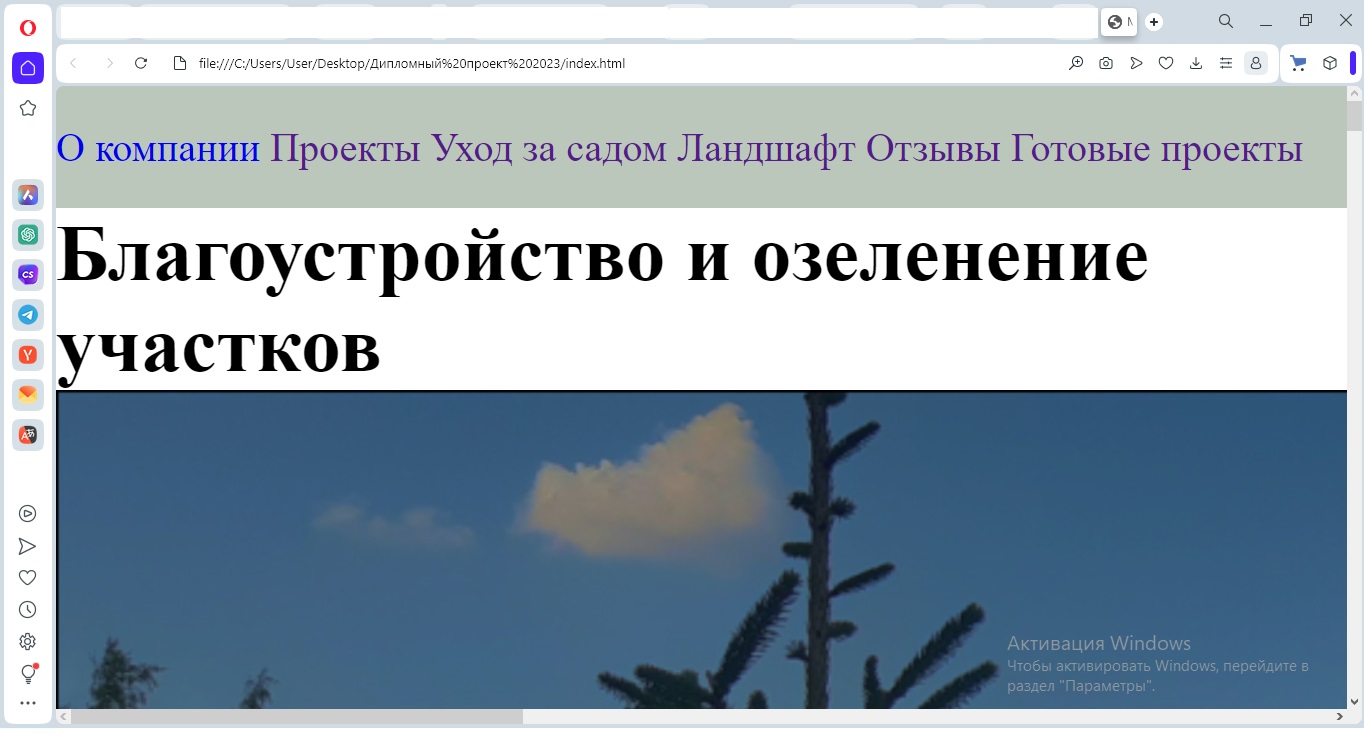
CSS-стиль - это определённое правило, что подсказывает пользователю правила форматирования для каждого элемента. Под форматированием подразумевается: изменение цвета текста, фона элемента, шрифта, теней, позиции на экране и прочее.

Таблица стилей составлена из одной и более инструкций (называемых правилами или наборами правил), которые описывают, как элемент или группа элементов должны отображаться. Чтобы создать рабочий CSS-стиль, нужно правильно указать 2 основные части правила - селектор (целевой элемент, которому назначается стиль) и блок стилей (правила форматирования).

На Рисунке 2.1 селектором выступает тег «header». Это означает, что все стили в блоке будут применены к каждому header-элементу на странице (однако в настоящем проекте он всего лишь один и поэтому ему можно не присваивать class).

Рисунок 2.1

В фигурных скобках всегда указываются стили. Они должны стоять строго в скобках, а не за их пределами. В представленном примере на выходе получится «header» как представлен она Рисунке 2.2 – цветной блок c выравненными ссылками-разделами нового сайта.

Рисунок 2.2

Все команды имеют свойство, а за ним идёт значение т.е. background – свойство, а rgba(37, 73, 31, 0.31) – значение этого свойства и указывается после двоеточия. По завершению команды всегда ставится точка с запятой. Этот символ является делителем, который отделяет между собой команды.

Список различных стилей и принято называть таблицей стилей или CSS. Есть несколько разных методов указания стилей.

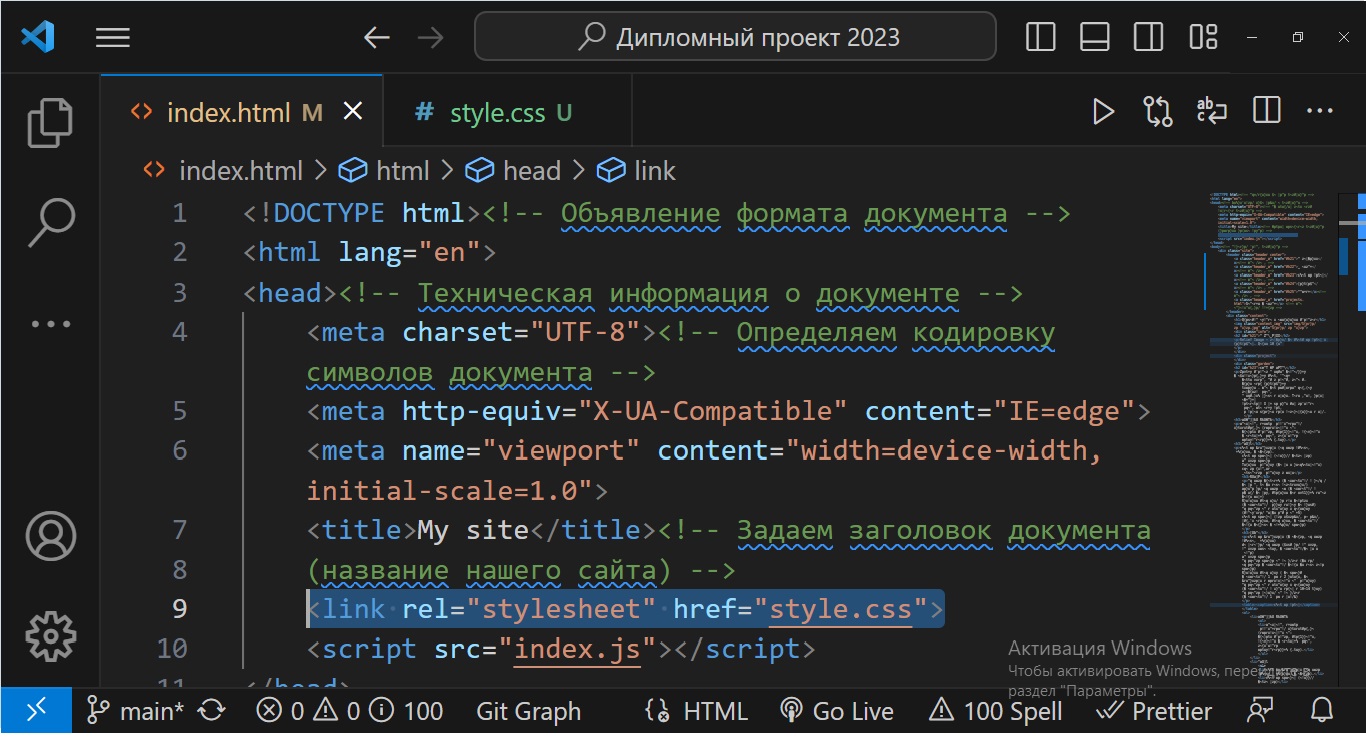
Большинство HTML-элементов поддерживают установку стилей посредством атрибутов. Отдельным элементам есть возможность указать атрибуты width и height (ширина с высотой). Атрибуты указываются прямо в HTML-файле. Это удобный способ указать стили для конкретного блока, все прочие теги, подпадающие под общие селекторы, будут иметь стили с CSS-файла, но это же портит HTML-код, он перестаёт быть чистым и в случае редактирования затруднительно находить и менять стили по разным файлам, появляется путаница. Поэтому применяется внешняя таблица стилей - текстовый файл с расширением .css (ранее в папке нашего проекта уже был создан файл style.css), в котором находится набор CSS-стилей элементов. Файл создаётся в редакторе кода, также, как и HTML-страница. Внутри файла могут содержатся только стили, без HTML-разметки.

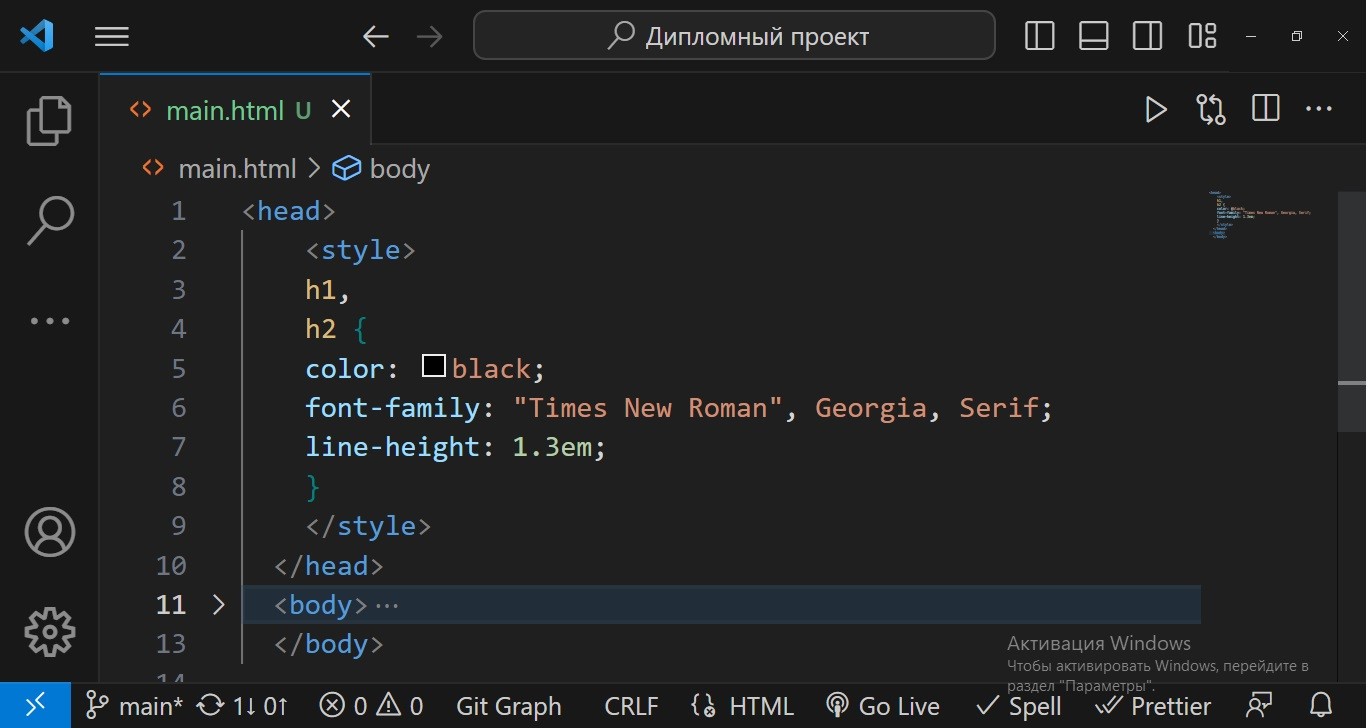
Внешняя таблица стилей подключена к веб-странице с помощью элемента «<link>», расположенного внутри раздела «<head></head>». Такие стили работают для всех страниц сайта.

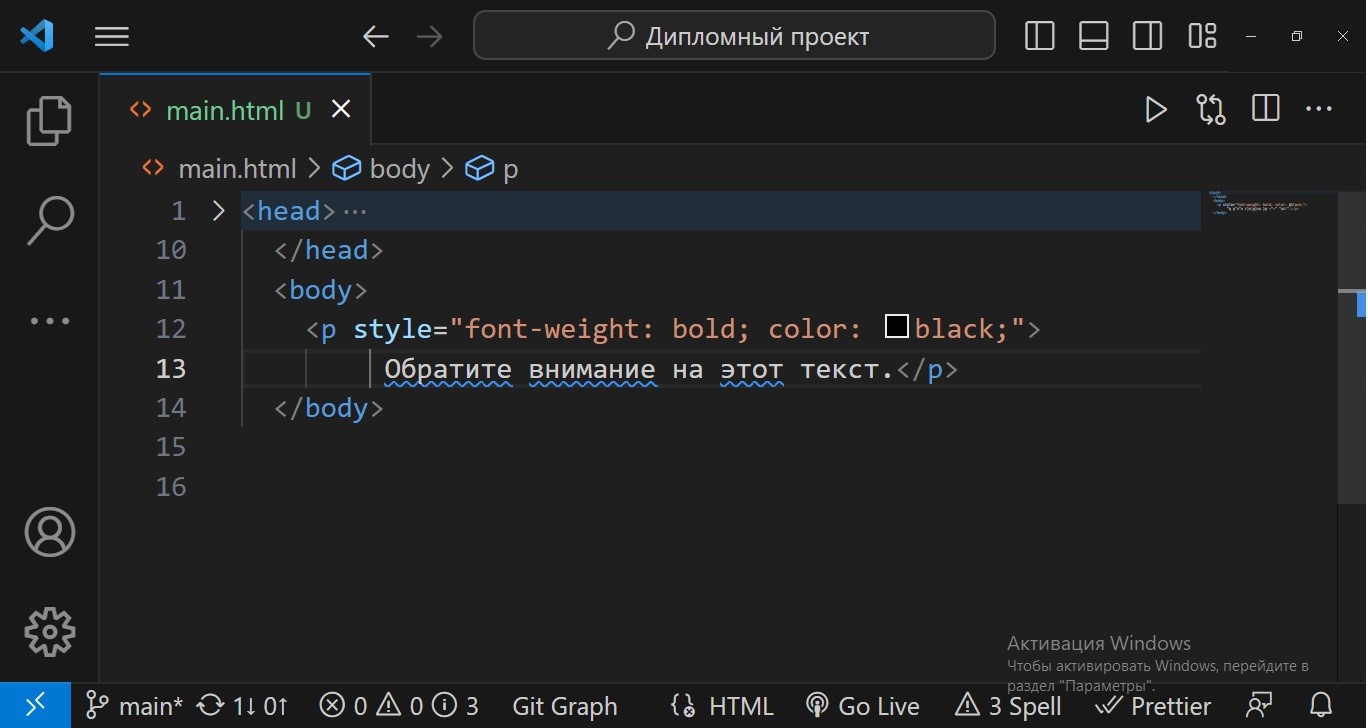
Внутренние же стили встраиваются в раздел <head></head> HTML-документа и определяются внутри элемента <style></style>.

Кроме перечисленных внешних и внутренних стилей есть также и встроенные, когда CSS-код пишется в HTML-файле, непосредственно внутри элемента с помощью атрибута «style».

Все представленные способы подключения стилей отображены соответственно на Рисунке 2.3, Рисунке 2.4, Рисунке 2.5. В рамках данного проекта будем использовать подключение внешнего файла со стилями.

Рисунок 2.3

Рисунок 2.4

Рисунок 2.5

Такие стили действуют только на тот элемент, для которого они заданы (элемент <p>).

Селекторы представляют структуру веб-страницы. С их помощью создаются правила для форматирования элементов веб-страницы. Селекторы могут быть:

- селекторы элементов - селектор, позволяющий форматировать все элементы данного типа на всех страницах сайта;

- селекторы класса - селектор, задающий стили для одного и более элементов с одинаковым именем класса,

- селектор идентификатора - селектор, форматирующий один конкретный элемент (значение id является уникальным и на одной странице может встречаться только один раз);

- селектор потомков – селектор, применяющий стили к элементам, расположенным внутри элемента-контейнера (например, ul li {text-transform: uppercase;} - выберет все элементы li, являющиеся потомками всех элементов ul);

- дочерний селектор – селектор, позволяющий применить стили только, если дочерний элемент идёт сразу за родительским элементом и между ними нет других элементов, то есть дочерний элемент больше ни во что не вложен;

- селекторы сестринских элементов – селекторы, позволяющие выбрать элементы из группы элементов одного уровня (h1 + p - выберет все первые абзацы, идущие непосредственно за любым элементом <h1>, не затрагивая остальные абзацы, а h1 ~ p - выберет все абзацы, являющиеся сестринскими по отношению к любому заголовку h1 и идущие сразу после него);

- селектор атрибута - выбираются элементы на основе имени атрибута или значения атрибута;

- cелектор псевдокласса – селектор классов, не прикрепленных к HTML-элементам и позволяющий применить CSS-правила к элементам при совершении события или подчиняющимся определенному правилу (не посещенная ссылка, посещенная ссылка, любой элемент, по которому проводят курсором мыши и многие другие);

- селектор структурных псевдоклассов - селектор, отбирающий дочерние элементы в соответствии с параметром, указанным в круглых скобках;

- селектор структурных псевдоклассов типа – селектор, указывающий на конкретный тип дочернего элемента;

- селектор псевдоэлемента – селектор, добавляющий содержимое.

Проверка, какие стили применяются, может осуществляться в режиме разработчика браузера (в браузере необходимо щёлкнуть правой кнопкой мыши на любое место на странице и выбрать пункт «Посмотреть код»).

Некоторые из перечисленных селекторов представлены на Рисунке 2.6

Рисунок 2.6

2.2 Работа с текстом

Значительное количество содержимого сайтов составляет текстовый контент поэтому работа с текстом является одной из самых важных элементов в профессиональной верстке со стороны как HTML, так и CSS.

Основные моменты при работе с текстом:

- выбор шрифтов текста (свойство font-family);

- насыщенность шрифта (свойство font-weight);

- ширина шрифта (свойство font-stretch);

- начертание шрифта (свойство font-style);

- цвет шрифта (свойство color);

- размер шрифта (свойство font-size);

- выравнивание текста (свойство text-align);

- межбуквенный и межсимвольный интервал (свойства word-spacing и letter-spacing);

- межстрочный интервал (свойство line-height);

- преобразование текста (свойство text-transform);

- обработка пробелов и переносы строк (свойство white-space);

- настройка табуляции (свойство tab-size);

- разрыв для букв (свойство word-break);

- разрыв строки (свойство line-break);

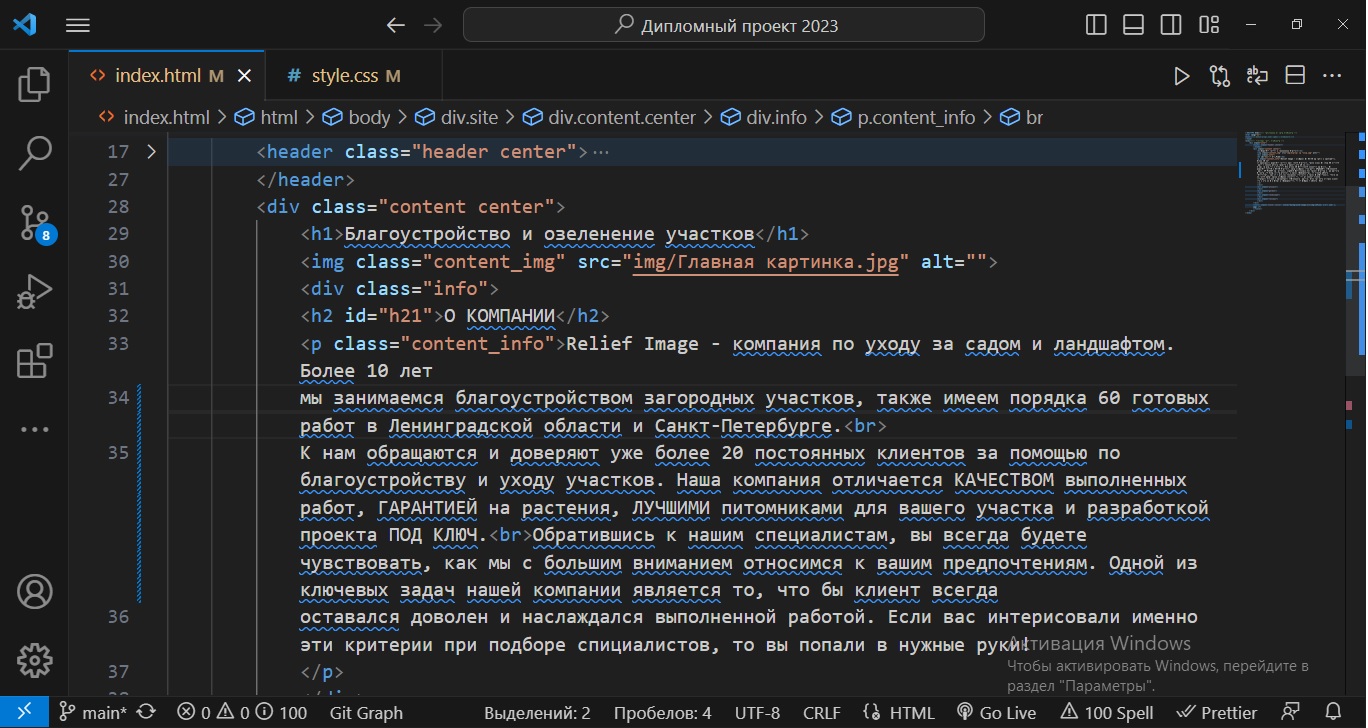
- расстановка переносов (свойство hyphens);

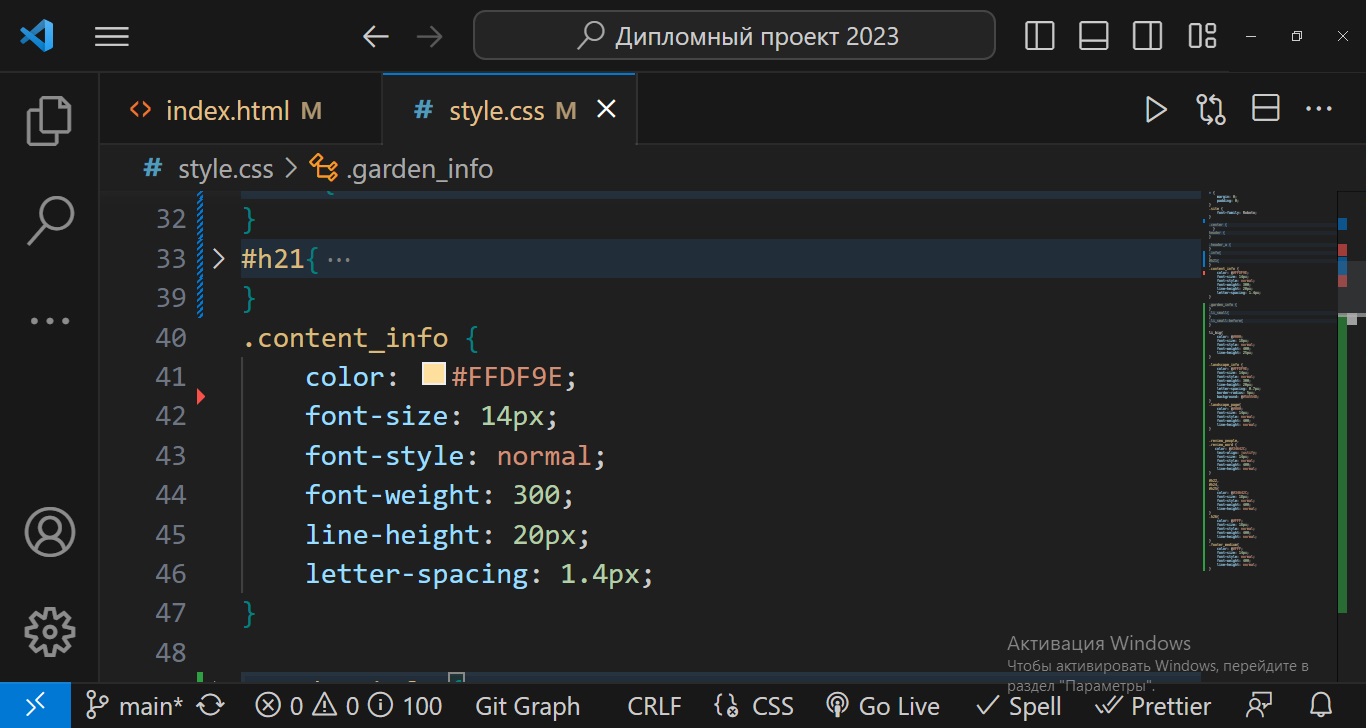
- переполнение блока-обертки (свойство overflow-wrap/word-wrap);

- цвета фона (свойство background-color) и изображения (свойство background-image).

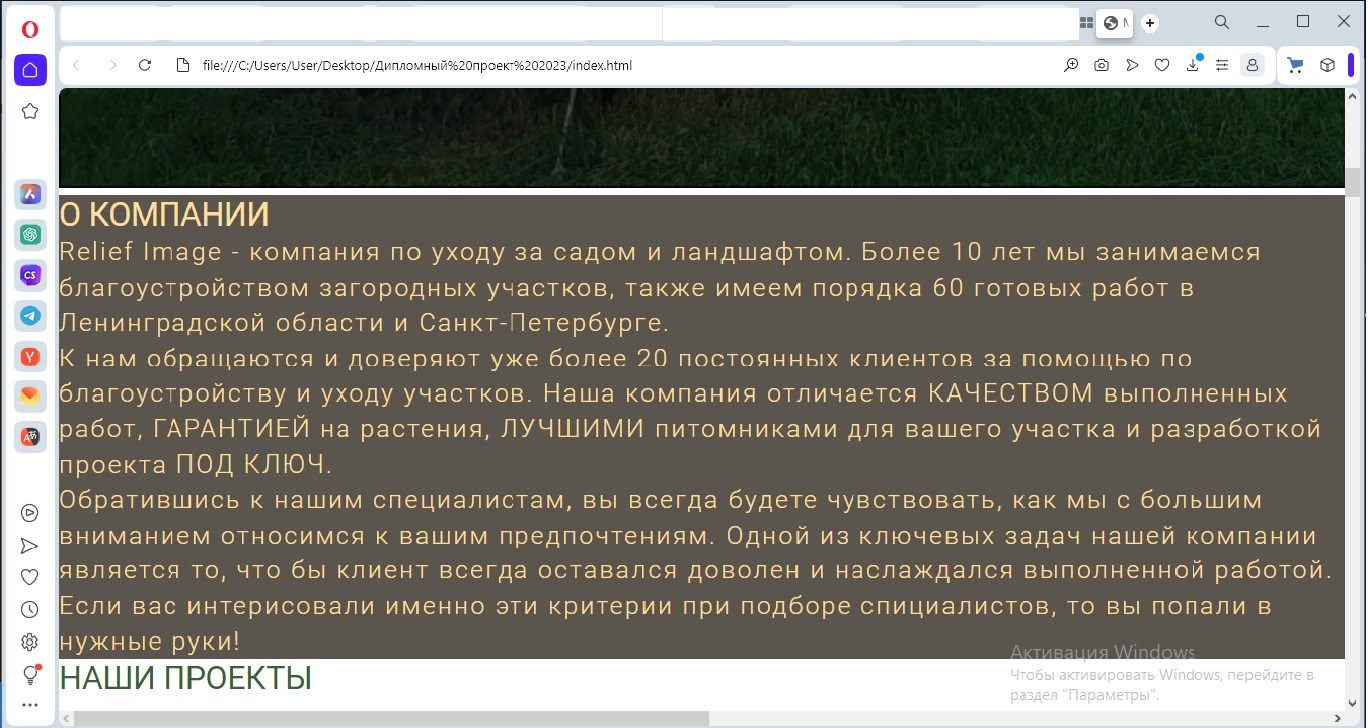
Из представленного списка свойств особенно стоит обратить внимание на свойство font-family, подключаемое аналогично внешнему файлу со стилями с ресурса - <https://fonts.google.com>. В большинстве случаев вариантов такого свойства не может быть много на веб-странице и поэтому непосредственно в таблице стилей прописывается вообще для всех текстовых элементов одновременно (не требуется прописывать для каждого элемента и достаточно применить или объявить сразу ко всему проекту).

Для текстовой части («p» с классом «content\_info») раздела сайта «О компании» (Рисунок 2.7), применим стили как показано на Рисунке 2.8.

Рисунок 2.7

Рисунок 2.8

Получим более привлекательный внешний вид данного раздела (Рисунок 2.9). При этом перечень свойств для всех текстовых элементов в нашем проекте примерно одинаков, различаясь лишь значениями данных свойств.

Рисунок 2.9

2.3. Блочные и строчные элементы

В создаваемом проекте (верстке) сайта можно выделить две основные группы HTML-элементов, которые соответствуют типам их содержимого и поведению в структуре веб-страницы - блочные и строчные элементы.

С помощью блочных элементов можно создавать структуру веб-страницы, строчные элементы используются для форматирования текстовых фрагментов (за исключением элементов <area> и <img>).

Любой html-элемент создает на веб-странице прямоугольный контейнер. Все видимое на экране состоит из контейнеров разных типов.

Обычно блочные элементы создают структурные блоки и выводятся вертикально один над другим, занимая по ширине всю ширину блока, в котором находятся.

Строковые контейнеры же создают строковые блоки и выводятся в строке горизонтально и при этом ширина строковых элементов равна ширине их содержимого.

Строчно-блочный элемент также создает строку текста, где низ элемента располагается на базовой линии строки текста и не разрывает строку.

Содержимое элемента форматируется так же, как и для блочных элементов, а ширина блока равна ширине содержимого.

С помощью свойства «display» можно задать следующее:

- block - элемент занимает всю доступную ширину;

- inline элемент действует как обычный текст;

- inline-block соединение блочного и строчного поведения;

- list-item – элемент занимает всю доступную ширину, но показывает дополнительный маркер пункта списка.

В нашем проекте сайта строчными является лишь элемент «img», а блочными – «div», «p» «h1» - «h3». Особенно заметно свойство блочных элементов создавать прямоугольный элемент и возможность вкладываться в друг в друга видно на рисунках 2.7 – 2.9.

2.4. Работа с полями

Как говорилось в предыдущем разделе любой сайт состоит из блоков. Каждый блок может быть выделен тегом «div» или же каким-либо специальным тегом по типу: «header», «footer», «aside», «main» и так далее. Модуль CSS описывает свойства padding и margin, которые создают поля внутри и отступы снаружи CSS блока соответственно. Размеры блока также могут быть увеличены за счет рамок.

Каждый блок имеет прямоугольную область содержимого в центре, поля вокруг содержимого, рамку вокруг полей и отступ за пределами рамки. Размеры этих областей определяют свойства padding и его подсвойства - padding-left, padding-top и т.д., border и его подсвойства, margin и его подсвойства.

Поля, рамка и отступы могут быть разбиты на верхний, правый, нижний и левый сегменты, каждый из которых независимо управляется своим соответствующим свойством.

Фон области содержимого, полей и рамки блока определяется свойствами фона. Область рамки может быть дополнительно окрашена с помощью свойства border. Отступы элемента всегда прозрачны, что позволяет показывать фон родительского элемента.

Так как поля и отступы элемента не являются обязательными, по умолчанию их значение равно нулю. Тем не менее, некоторые браузеры добавляют этим свойствам положительные значения по умолчанию на основе своих таблиц стилей. Ранее мы уже очистили стили браузеров для всех элементов при помощи универсального селектора (\* {margin: 0; padding: 0;}), что рекомендуется делать при создании любого нового проекта (это первые строки кода в файле style.css).

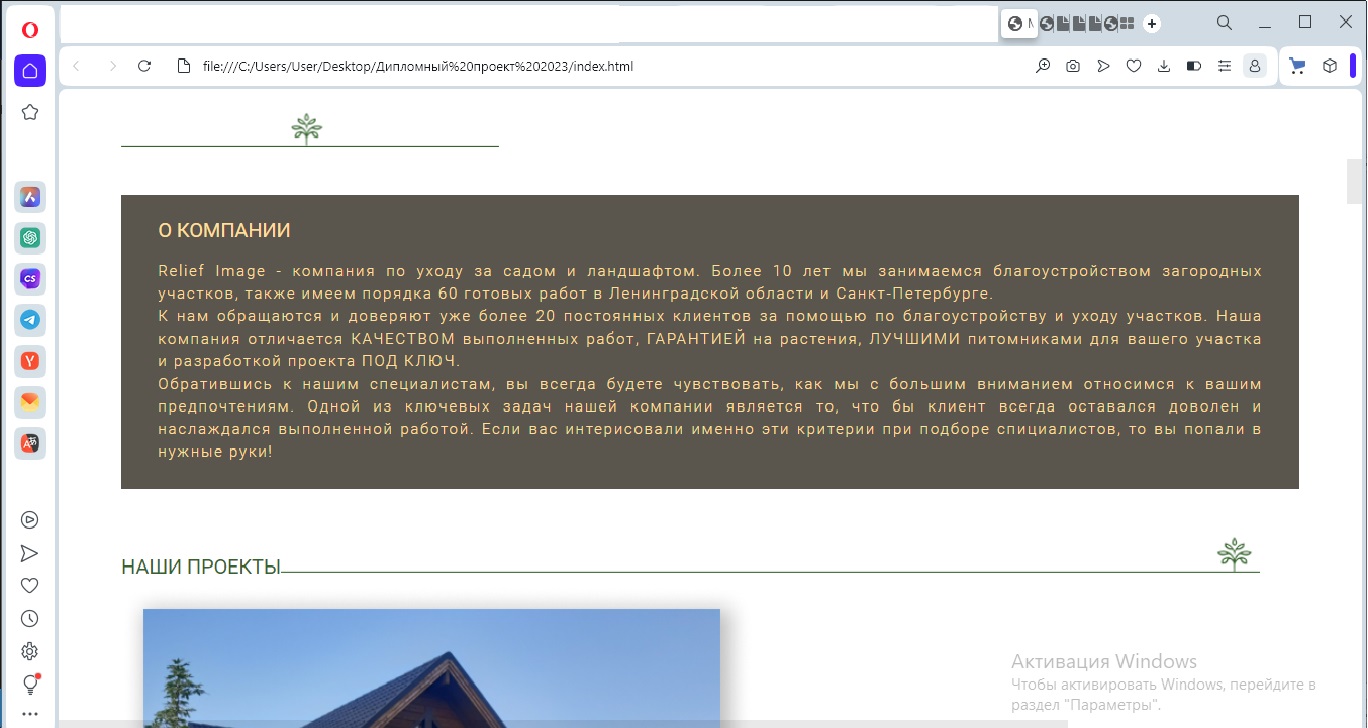
Свойство margin является сокращенным свойством для установки margin-top, margin-right, margin-bottom и margin-left в одном объявлении. Если существует только одно значение, оно применяется ко всем сторонам. Если два - верхний и нижний отступы устанавливаются на первое значение, а правый и левый - устанавливаются 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.

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.

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.

Расставляем все необходимые отступы блоков веб-страницы с учетом того факта, что в подавляющем большинстве случаев используются padding-bottom и padding-top для расставления внутренних отступов между блочными элементами, а margin-bottom и margin-top – для внешних. Так ранее код и его отображение браузером, представленные на рисунках 2.7 – 2.9, теперь выглядит как показано на Рисунках 2.10 и 2.11, где также можно заметить отступы в блоке сайта «О компании» между заголовком и основным текстом, а не только между текстовым содержимым и всем блоком. Отсуп между заголовком и текстом выставляется в стилях самого заголовка как padding-bottom, но имеется вариант его «обнуления» и выставления ненулевого значения уже для padding-top текстового содержимого.

Рисунок 2.10

Рисунок 2.11

2.5 Позиционирование

CSS рассматривает макет html-документа как дерево элементов. Позиционирование в CSS описывает, как любой из элементов может быть размещен независимо от порядка документа.

В CSS блок элемента может быть расположен по следующим схемам позиционирования:

- нормальный поток (блочный и строчный (встроенный) контексты форматирования, и относительное и «липкое» позиционирование элементов уровня блока и строки);

- обтекание (блок удаляется из нормального потока и позиционируется влево или вправо с float: left и float: right соответственно);

- абсолютное позиционирование (блок полностью удаляется из нормального потока и ему присваивается позиция относительно содержащего блока с помощью значений position: absolute; и position: fixed;).

Float — это свойство, определяет по какой стороне будет выравниваться элемент, а остальные элементы будут обтекать объект со данным свойством с правой стороны (right), либо с левой (left), либо вовсе отменить позиционирование (float: none). Уместнее всего использовать float для обтекания картинки текстом.

Свойство position определяет, какой из алгоритмов позиционирования используется для вычисления положения блока:

- static (блок располагается в соответствии с нормальным потоком);

- relative (положение блока рассчитывается по нормальному потоку и затем блок смещается относительно его нормального положения и не влияет на положение любых следующих блоков);

- absolute (положение блока задается с помощью свойств top, right, bottom и left и они определяют явное смещение относительно его содержащего блока);

- sticky (положение блока рассчитывается в соответствии с нормальным потоком, а затем блок смещается относительно своего ближайшего предка с прокруткой или окна просмотра, если ни у одного из предков нет прокрутки);

- fixed (фиксированное позиционирование аналогично абсолютному позиционированию, но содержащим блоком устанавливается окно просмотра);

- initial (значение свойства в значение по умолчанию);

- inherit (наследует значение свойства от родительского элемента).

Свойство z-index позволяет изменить порядок наложения позиционированных элементов в случае, когда они накладываются друг на друга.

2.6 Работа с «display: flex»

В CSS flexbox (Flexible Box Layout Module) – это модуль макета гибкого контейнера, то есть способ компоновки элементов, в основе которой лежит идея оси. Flexbox состоит из гибкого контейнера (flex container) и гибких элементов (flex items). Гибкие элементы могут выстраиваться в строку или столбик, а оставшееся свободное пространство распределяется между ними различными способами.

Flexbox позволяет:

- располагать элементы в одном из четырех направлений: слева направо, справа налево, сверху-вниз или снизу-вверх;

- переопределять порядок отображения элементов;

- автоматически определять размеры всех элементов, чтобы они вписывались в доступное пространство;

- производить горизонтальное и вертикальное центрирование;

- переносить элементы внутри контейнера, не допуская его переполнения;

- создавать колонки контента одинаковой высоты;

- создавать прижатый к низу страницы подвал («footer») сайта.

Порядок (переупорядочение) отображения содержимого flex-элементов и ориентация (переопределение направления) с помощью «flex-direction», «flex-wrap», «flex-flow и «order» с описанием и возможными значениями представлены в Таблице 2.1.

Таблица 2.1

|  |  |
| --- | --- |
| Наименование flex-элемента | Описание flex-элемента |
| flex-direction | Управление направлением главной оси, вдоль которой укладываются flex-элементы, в соответствии с определённым значением:   * row (в строку слева направо); * row-reverse (в строку справа налево); * column (вертикально сверху-вниз); * column-reverse (вертикально снизу-вверх); * inherit (наследуемый порядок); * initial (по умолчанию) |
| flex-wrap | Определение, будет ли flex-контейнер однострочным или многострочным, и направление поперечной оси, определяющее направление укладки новых линий flex-контейнера:   * nowrap (значение по умолчанию); * wrap (элементы переносятся, располагаясь в несколько горизонтальных рядов (если не помещаются в один ряд) слева направо); * wrap-reverse (элементы переносятся на новые линии, располагаясь в обратном порядке слева-направо); * initial (значение свойства в значение по умолчанию); * inherit (наследуемое значение) |
| flex-flow | Определение направления главной и поперечной осей, а также возможность переноса flex-элементов при необходимости на несколько строк (принимает значения от flex-wrap и flex-direction одновременно) |
| order | Определение порядка, в котором flex-элементы отображаются и располагаются внутри flex-контейнера:   * число (значение по умолчанию 0); * initial (значение по умолчанию); * inherit (наследуемое значение) |

Гибкость flex-элементов – это возможность «сгибать» flex-элементы, изменяя их ширину / высоту (в зависимости от того, какой размер находится на главной оси), чтобы заполнить доступное пространство в основном измерении (Таблица 2.2).

Таблица 2.2

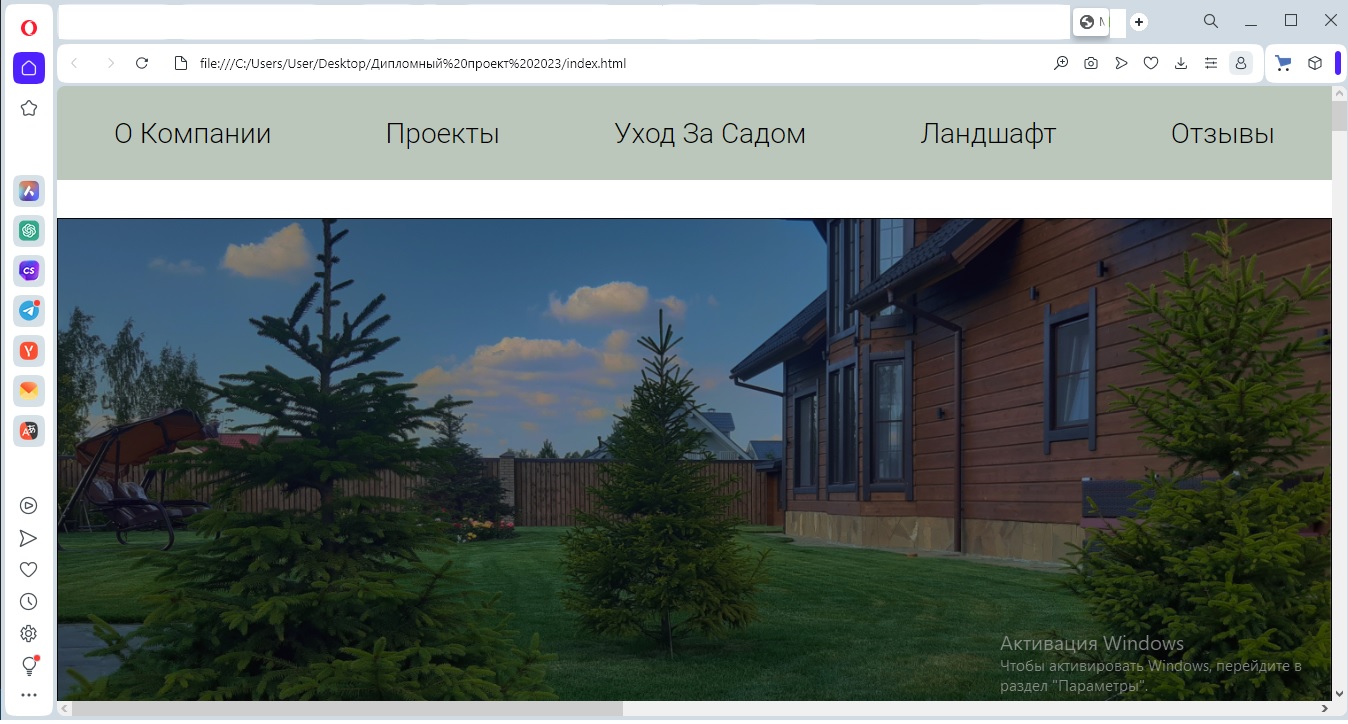
|  |  |
| --- | --- |
| Наименование flex-элемента | Описание flex-элемента |
| flex-basis | Установление начального основного размера flex-элемента до распределения свободного пространства в соответствии с коэффициентами гибкости:   * auto (значение по умолчанию);   - content (значение в зависимости от содержимого flex-элемента т.е. контента);  - длина (значение задается в единицах длины);  - initial (значение по умолчанию);  - inherit (наследуемое значение) |
| flex-grow | Определение коэффициента роста одного flex-элемента относительно других flex-элементов в flex-контейнере при распределении положительного свободного пространства:   * число (значение по умолчанию 0); * initial (значение по умолчанию);   - inherit (наследуемое значение) |
| flex-shrink | Определение, насколько блоку можно уменьшиться в размере:  - число (положительное целое или дробное число, значение по умолчанию 1).  - initial (значение умолчанию);  - inherit (наследуемое значение) |

Свойство выравнивания flex-элементы по главной оси flex-контейнера, распределяя свободное пространство, незанятое flex-элементами (Таблица 2.3).

Таблица 2.3

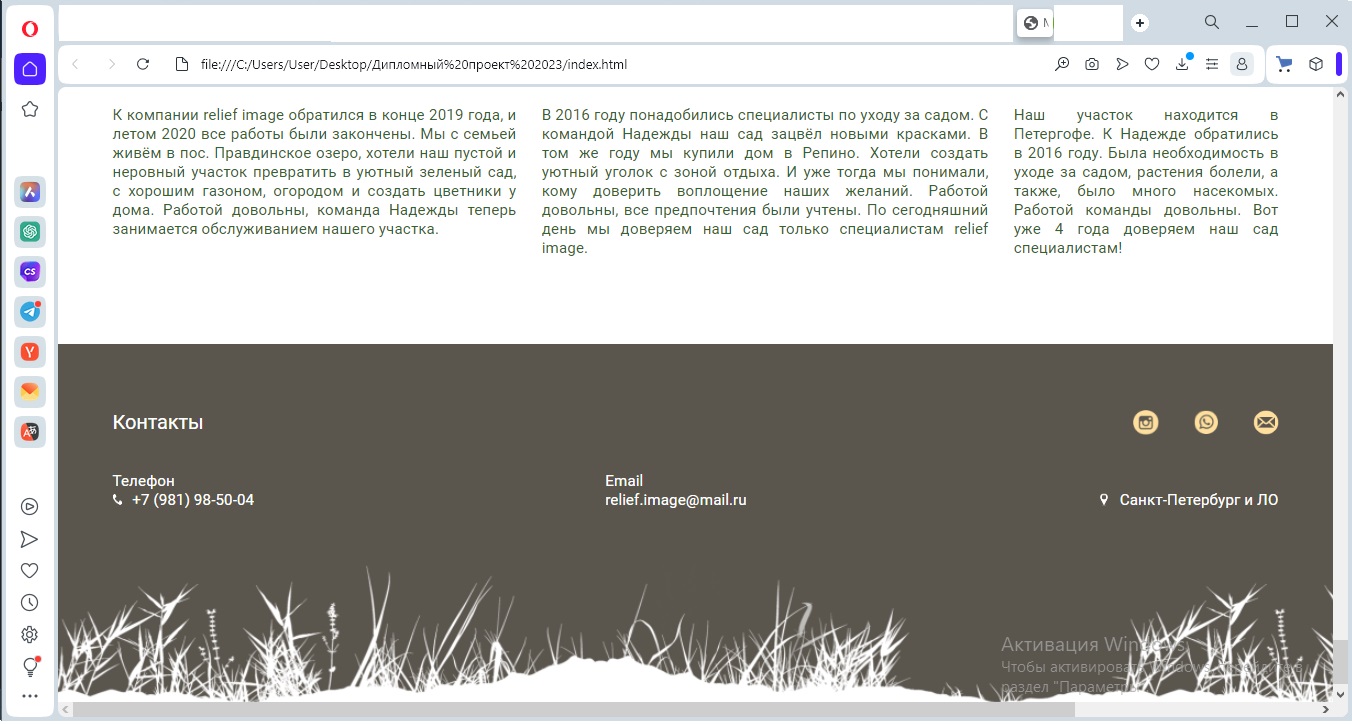
|  |  |
| --- | --- |
| Наименование flex-элемента | Описание flex-элемента |
| align-content | Выравнивание строки в flex-контейнере на поперечной оси:  - flex-start (значение по умолчанию - элементы укладываются в направлении, идущем от начала flex-контейнера);  - flex-end (элементы размещаются в конце flex-контейнера);  - center (flex-элементы выравниваются по центру flex-контейнера);  - space-between (flex-элементы равномерно распределяются по линии, где первый элемент помещается вровень с краем начальной линии, последний - вровень с краем конечной линии, а остальные элементы на линии распределяются так, чтобы расстояние между любыми двумя соседними элементами было одинаковым);  - space-around (flex-элементы на линии распределяются так, чтобы расстояние между любыми двумя смежными flex-элементами было одинаковым, а расстояние между первым / последним flex-элементами и краями flex-контейнера составляло половину от расстояния между элементами);  - initial (значение по умолчанию);  - inherit (наследуемое значение) |
| justify-content | Выравнивание flex-элементов по главной оси flex-контейнера, распределяя свободное пространство, незанятое flex-элементами):   * flex-start (значение по умолчанию, когда элементы выкладываются в направлении, идущем от начала flex-контейнера); * flex-end (flex-элементы размещаются в конце flex-контейнера); * center (flex-элементы выравниваются по центру flex-контейнера); * space-between (flex-элементы равномерно распределяются по линии, когда первый элемент помещается вровень с краем начальной линии, последний - вровень с краем конечной линии, а остальные flex-элементы на линии распределяются так, чтобы расстояние между любыми двумя соседними элементами было одинаковым); * space-around (flex-элементы на линии распределяются так, чтобы расстояние между любыми двумя смежными flex-элементами было одинаковым, а расстояние между первым / последним flex-элементами и краями flex-контейнера составляло половину от расстояния между flex-элементами); * initial (значение по умолчанию); * inherit (наследуемое значение) |
| align-items | Выравнивание по поперечной оси текущей строки flex-контейнера для всех элементов flex-контейнера)   * flex-start (значение по умолчанию, когда элементы выкладываются в направлении, идущем от начала flex-контейнера); * flex-end (элементы в конце flex-контейнера); * center (flex-элементы выравниваются по центру flex-контейнера); * space-between (flex-элементы равномерно распределяются по линии, где первый flex-элемент помещается вровень с краем начальной линии, последний - вровень с краем конечной линии, а остальные flex-элементы на линии распределяются так, чтобы расстояние между любыми двумя соседними элементами было одинаковым); * - space-around (flex-элементы на линии распределяются так, чтобы расстояние между любыми двумя смежными flex-элементами было одинаковым, а расстояние между первым / последним flex-элементами и краями flex-контейнера составляло половину от расстояния между flex-элементами); * initial (значение по умолчанию); * inherit (наследуемое значение) |
| align-self | Выравнивание отдельно взятого flex-элемента по высоте flex-контейнера, переопределяя выравнивание, заданное align-items:  - flex-start (значение по умолчанию, когда элементы выкладываются в направлении, идущем от начала flex-контейнера);  - flex-end (flex-элементы размещаются в конце flex-контейнера);  - center (flex-элементы выравниваются по центру flex-контейнера);  - space-between (flex-элементы равномерно распределяются по линии, когда первый flex-элемент помещается вровень с краем начальной линии, последний - вровень с краем конечной линии, а остальные flex-элементы на линии распределяются так, чтобы расстояние между любыми двумя соседними элементами было одинаковым);  - space-around (flex-элементы на линии распределяются так, чтобы расстояние между любыми двумя смежными flex-элементами было одинаковым, а расстояние между первым / последним flex-элементами и краями flex-контейнера составляло половину от расстояния между flex-элементами);  - initial (значение по умолчанию);  - inherit (наследуемое значение). |

Для использования каких-либо flex-элементов необходимо объявить правило «display: flex». Далее используя свойства «justify-content: space-around» и «align-items: center» выровняем элементы шапки сайта (меню сайта) блока «header» получим визуальное отображение браузером как представлено на Рисунке 2.12.

Рисунок 2.12

Для всех блоков, содержащих в себе большой объем текста, подойдет свойство «justify-content: center» (выравнивание вдоль главной оси по центру).

Для элементов блока «footer» (заголовок, иконки соцсетей, номер телефона, местонахождение и др.) выставим «display: flex; justify-content: space-between» (Рисунок 2.13) поскольку для данной области более уместно «прижатие» к границам области сайта («шапка» сайта более центрирована).

Рисунок 2.13

2.7 CSS Grid

CSS Grid - это система двумерного макета для дизайна пользовательского интерфейса. Главная идея - разделении веб-страницы на столбцы и строки. В образовавшиеся области сетки можно помещать элементы сетки, а управлять их размерами и расположением можно с помощью специальных свойств.

Grid обеспечивает двухмерное выравнивание, использует нисходящий подход к макету, допускает явное перекрытие элементов, а Flexbox фокусируется на распределении пространства по оси, использует более простой восходящий подход к макету, может использовать систему переноса строк на основе размера контента для управления своей вторичной осью и опирается на базовую иерархию разметки для построения более сложных макетов.

Сетка (grid) представляет собой набор пересекающихся горизонтальных и вертикальных линий, делящих пространство grid-контейнера на области сетки, в которые могут быть помещены элементы сетки.

Линии сетки (grid lines) — это невидимые горизонтальные и вертикальные разделительные линии, они существуют по обе стороны от строки и столбца. На них можно ссылаться по числовому индексу (используя свойства grid-column-start, grid-column-end, grid-row-start и grid-row-end) или имени, заданному в CSS-коде. Числовые индексы сетки зависят от стиля языка, поэтому первым столбцом может быть как самый левый, так и самый правый столбец.

Выделяют две группы линий сетки: одна группа определяет столбцы, которые проходят вдоль оси блока (ось столбцов), и перпендикулярная группа, определяющая строки, простирающиеся вдоль линейной оси (ось строк), в соответствии с CSS3 режимом записи.

Дорожка сетки (grid track) — пространство между двумя соседними линиями сетки, используется для определения либо столбца, либо строки сетки. Дорожка идет от одного края контейнера к другому, размер зависит от расположения линий сетки, которые ее определяют. Дорожки сетки аналогичны столбцам и строкам таблицы. По умолчанию смежные дорожки плотно прилегают друг к другу, задать расстояние между ними можно с помощью свойств row-gap, column-gap и gap.

Ячейка сетки (grid cell) — пространство, ограниченное четырьмя линиями сетки, аналогично ячейке таблицы. Ячейка сетки — это область, в которой можно разместить контент. Это наименьшая единица сетки, на которую можно ссылаться при позиционировании элементов сетки. К ячейкам сетки нельзя обращаться напрямую с помощью CSS-свойств.

Область сетки (grid area) — прямоугольная область, ограниченная четырьмя линиями сетки и состоящая из одной или нескольких соседних ячеек. Область может быть такой же маленькой, как одна ячейка, или такой же большой, как все ячейки сетки. Область сетки может быть задана явно с помощью свойства grid-template-areas, по умолчанию на нее ссылаются ограничивающие линии сетки.

Элементы сетки (grid items) — отдельные элементы, которые назначаются области сетки (или ячейке сетки). Каждый контейнер-сетка включает ноль и более элементов сетки; каждый дочерний элемент контейнера-сетки автоматически становится элементом сетки.

Дорожки, ячейки и области сетки построены из линий сетки. Тем не менее не требуется, чтобы все области сетки были заполнены элементами, вполне возможно, что некоторые или даже большинство ячеек сетки будут пустыми от любого содержимого. Также возможно, что элементы сетки будут перекрывать друг друга, либо определять перекрывающиеся области сетки.

2. Создание контейнера-сетки: значения grid и inline-grid свойства display

Для создания макета на основе сетки необходимо определить контейнер-сетку.

Контейнер-сетка (grid container) — это блок, который устанавливает контекст форматирования по типу сетки, то есть создает область с сеткой, а дочерние элементы располагаются в соответствии с правилами компоновки сетки, а не блочной компоновки. Когда вы определяете контейнер сетки с помощью display: grid или display: inline-grid, вы создаете новый контекст форматирования для содержимого этого контейнера, который влияет только на дочерние элементы сетки.

Контейнер-сетка бывает двух видов: обычный display: grid и встроенный display: inline-grid. Первый генерирует grid-контейнер уровня блока, второй — grid-контейнер уровня строки. Контейнеры-сетки не являются блочными контейнерами, поэтому некоторые CSS-свойства не работают в контексте макета сетки:

float и clear игнорируются элементами сетки (но не самим контейнером-сеткой).

vertical-align не влияет на элементы сетки.

Псевдоэлементы ::first-line и ::first-letter не применяются к контейнеру-сетке и его потомкам.

Если контейнер-сетка является контейнером уровня строки display: inline-grid и для него заданы обтекание или абсолютное позиционирование, то вычисляемое значение свойства display будет grid.

3. Определение сетки

Когда вы создаете контейнер-сетку, сетка по умолчанию имеет один столбец и одну строку, которые занимают полный размер контейнера. Для разделения контейнера-сетки на столбцы и строки используются свойства grid-template-columns, grid-template-rows и grid-template-areas. С помощью этих свойств можно определить сетку явно.

Окончательная сетка может оказаться больше из-за элементов сетки, размещенных вне явной сетки; в этом случае будут созданы неявные дорожки, размер этих неявных дорожек будет определяться свойствами grid-auto-rows и grid-auto-columns.

Свойства grid и grid-template — это сокращенные обозначения, которые можно использовать для одновременной установки всех трех явных свойств сетки grid-template-columns, grid-template-rows и grid-template-areas. grid сбрасывает свойства, управляющие неявной сеткой, тогда как свойство grid-template оставляет их без изменений.

3.1. Как задать количество строк и столбцов: свойства grid-template-rows и grid-template-columns

Количество строк / столбцов задается с помощью свойств grid-template-rows и grid-template-columns.

Свойства не наследуются.

3.1.1. Относительные, абсолютные единицы и процентные значения для определения дорожек сетки (длина)

Размеры дорожек сетки можно задавать с помощью положительных значений, используя относительные единицы длины — например, em, vh, vw; абсолютные единицы длины — px; и проценты %. Размеры в % вычисляются от ширины или высоты контейнера-сетки.

3.1.2. Гибкие размеры дорожек: единица измерения fr

fr — единица длины, которая позволяет создавать гибкие дорожки. Не является единицей измерения в обычном ее понимании, поэтому не может быть представлена или объединена с другими типами единиц в выражениях calc(). Общий размер фиксированных строк или столбцов вычитается из доступного пространства контейнера-сетки. Оставшееся пространство делится между строками и столбцами с гибкими размерами пропорционально их коэффициенту, например:

Если сумма всех гибких размеров дорожек меньше 1, они будут занимать только соответствующую долю оставшегося пространства, а не расширяться, чтобы заполнить его полностью.

Если доступное пространство бесконечно (то есть, ширина или высота контейнера-сетки не заданы), дорожки сетки гибкого размера масштабируются по своему содержимому, сохраняя при этом их соответствующие пропорции.

3.1.3. Минимальные и максимальные размеры дорожек

Ключевое слово max-content устанавливает для дорожки размер, который занимает максимально необходимое пространство с учетом содержимого элемента сетки.

min-content позволяет занимать минимальное пространство, необходимое для этого содержимого, при этом ширина элемента ориентируется на самое длинное слово или на самое широкое изображение.

Функция minmax(min,max) определяет диапазон размеров, больше или равный min и меньше или равный max. Если max < min, то max игнорируется, а minmax(min,max) обрабатывается как min. Значения в fr можно устанавливать только как максимальное.

3.1.4. Автоматические размеры

Значение auto ориентируется на содержимое элементов сетки одной дорожки. Как минимум, рассматривается как минимальный размер элемента сетки, как определено min-width или min-height. Как максимум, обрабатывается так же, как и max-content. Может растягиваться за счет свойств align-content и justify-content.

3.1.5. Соответствие содержимому

Размеры дорожек можно задавать с помощью значения fit-content(длина или %), представляющее собой формулу min(maximum size, max(minimum size, argument)), которая вычисляется как minmax(auto, max-content), то есть auto. При этом, размер дорожки ограничивается значением, указанным в скобках, и если оно больше, чем автоматический минимум.

3.1.6. Повтор строк и столбцов

Нотация repeat() представляет повторяющийся фрагмент списка дорожек, что позволяет записать в более компактной форме большое количество одинаковых по размерам столбцов или строк. Общая форма синтаксиса следующая:

Первый аргумент задает количество повторений, которое может быть задано с помощью положительного целого числа или ключевых слов. Второй аргумент - размер повторяющейся дорожки. Однако, существуют некоторые ограничения:

Нотация repeat() не может быть вложенной.

Значения auto-fill или auto-fit не могут быть совмещены с min-content, max-content, auto, fit-content() или fr.

Синтаксис repeat() имеет несколько форм:

Используя значение auto-fill, вы всегда получите хотя бы один столбец, даже если он по какой-то причине не помещается в контейнер-сетку. Если вы используете auto-fit, то дорожки, которые не содержат элементы сетки, будут сброшены.

3.2. Именованные области: свойство grid-template-areas

Свойство grid-template-areas определяет именованные области сетки, которые не связаны с каким-либо конкретным элементом сетки, но на которые можно ссылаться из свойств размещения сетки. Синтаксис свойства обеспечивает визуализацию структуры сетки, облегчая понимание общего макета контейнера-сетки.

С Каждый идентификатор сетки в значении grid-template-areas соответствует ячейке сетки. Как только все ячейки идентифицированы, браузер объединяет все смежные ячейки с одинаковыми именами в одну область, которая охватывает все их, при условии, что они описывают область прямоугольной формы. Если вы попытаетесь настроить более сложные области, весь шаблон будет недействительным и области сетки не будут определены.

Все строки должны содержать одинаковое количество столбцов. Если вы хотите определить только некоторые ячейки как часть области сетки, вы можете использовать одну или несколько . для заполнения этих безымянных ячеек. При определении областей сетки идентификаторы можно перечислять через единичный пробел, без разрыва строки. Или же выровнять с помощью пробелов/табуляции и перевода строки для большей наглядности.

Области сетки полезны для определения семантических отношений между различными частями макета страницы, позволяя указать, какая часть страницы включает в себя верхний колонтитул, боковую панель, область содержимого и нижний колонтитул.

После того, как вы создали области сетки, элементы сетки могут быть назначены непосредственно, чтобы занимать эти области, используя свойство grid-area.войство не наследуется.

3.3. Сокращенная запись явного объявления сетки: свойство grid-template

Свойство grid-template является сокращением для установки свойств grid-template-rows, grid-template-columns и grid-template-areas в одном объявлении.

Свойство не наследуется.

Функция repeat() не разрешена для определения списка дорожек в этом свойстве, если используются именованные области сетки (сетка просто не будет отрисована).

3.4. Неявная сетка

3.4.1. Автоматическое создание дорожек сетки: свойства grid-auto-rows и grid-auto-columns

Если элемент сетки расположен в строке или столбце, размер которых не определен явно grid-template-rows или grid-template-columns, создаются неявные дорожки сетки для его хранения. Это может произойти в случае, если строка или столбец оказались за пределами установленных размеров сетки.

По умолчанию эти автоматически добавляемые дорожки имеют минимальный необходимый размер. Свойства grid-auto-rows и grid-auto-columns позволяют контролировать размер неявных дорожек сетки. Если дано несколько размеров дорожек, шаблон повторяется по мере необходимости, чтобы найти размер неявных дорожек. Первая неявная дорожка сетки после явной сетки получает первый заданный размер и так далее.

Свойства не наследуются.

3.4.2. Автоматическое размещение: свойство grid-auto-flow

Элементы сетки, которые не размещены явно, автоматически помещаются в незанятое пространство в контейнере-сетке с помощью алгоритма автоматического размещения. Свойство grid-auto-flow управляет автоматическим размещением элементов сетки без явного положения. После заполнения явной сетки (или если явной сетки нет) автоматическое размещение также приведет к генерации неявных дорожек сетки.

Свойство не наследуется.

Свойство будет полезным при создании компактных галерей, если для изображений не задан порядок, в котором они должны быть расположены. Для каждого элемента сетки браузер сканирует всю сетку в заданном направлении потока (строка или столбец), начиная от начальной точки потока (верхний левый угол, на языках LTR - слева направо), пока не найдет место, куда поместится этот элемент сетки.

3.5. Сокращенная запись сетки: свойство grid

Свойство grid задает все явные grid-template-rows, grid-template-columns и grid-template-areas и все неявные свойства сетки grid-auto-flow, grid-auto-rows и grid-auto-columns в одном объявлении. Оно не сбрасывает свойства row-gap/column-gap. Его синтаксис соответствует grid-template, а также дополнительной синтаксической форме для определения автоматического размещения элементов сетки:

Путем явного задания дорожек на одной оси (устанавливая grid-template-rows или grid-template-columns и задавая другим значение none), и задавая, как автоматически повторять дорожки на другой оси (устанавливая grid-auto-rows или grid-auto-columns и задавая другим auto).

Для grid-auto-flow также устанавливается одно из трех допустимых значений. Все остальные подсвойства grid сбрасываются к своим начальным значениям.

4. Элементы сетки

Контейнер-сетка устанавливает новый контекст форматирования для элементов сетки, который обуславливает следующие особенности:

Для элементов сетки блокируется их значение свойства display. Значение display: inline-block вычисляется в display: block, анонимные блоки текста также занимают всю ширину контейнера и образуют разрыв строки.

Размер элемента сетки в пределах содержащего блока определяется его областью сетки.

Расчеты элементов сетки для width: auto и height: auto зависят от их значений align-self:

align-self: normal; - незамещаемые элементы заполняют область сетки, замещаемые элементы используют собственные размеры;

align-self: stretch; - обе категории элементов заполняют область сетки;

align-self: start/center и т.д. - незамещаемые элементы устанавливают размеры в соответствии со своим содержимым, замещаемые элементы используют собственные размеры.

Поскольку соседние элементы сетки находятся в независимых областях сетки, то поля соседних элементов сетки margin не схлопываются.

Браузеры по-разному обрабатывают процентные значения свойств margin и padding, поэтому не рекомендуется использовать их при задании значений этих свойств.

Поля margin: auto; расширяются, поглощая свободное пространство в соответствующем измерении, поэтому могут использоваться для выравнивания элемента.

5. Размещение и переупорядочивание элементов сетки

Свойства размещения позволяют свободно упорядочивать и переупорядочивать содержимое сетки таким образом, что визуальное представление может значительно отличаться от порядка элементов в html-документе.

5.1. Размещение с помощью линий сетки: свойства grid-row-start, grid-column-start, grid-row-end и grid-column-end

Каждый элемент сетки связан с областью сетки, которая определяет содержащий блок для элемента сетки. Положение элементов сетки определяется расположением линий сетки и диапазоном сетки - количеством занимаемых дорожек сетки. По умолчанию элемент сетки занимает одну дорожку на каждой оси. Поэтому можно опустить значение grid-column-end или grid-row-end.

Свойства размещения на сетке - grid-row-start, grid-row-end, grid-column-start и grid-column-end и их краткая запись grid-row, grid-column и grid-area позволяют определить размещение элемента сетки, предоставив любую (или ноль) из следующих шести частей информации:

5.1.1. Именованные линии сетки

Хотя на линии сетки можно ссылаться по их числовому индексу, именованные линии облегчают понимание и использование свойств размещения сетки. Линии могут быть названы явно в свойствах grid-template-rows и grid-template-columns или неявно путем создания именованных областей сетки в свойстве grid-template-areas.

Имя линии может быть любым, при указании в значении свойства оно заключается в квадратные скобки. В качестве имени линии нельзя использовать слово span.

Имена линий добавляются к неявным именам линий сетки, созданным свойством grid-template-areas, принимая вид name-start и name-end. Имена линий сетки никогда не заменяют другие имена линий сетки. Вместо этого они просто накапливаются.

5.2. Сокращенная запись свойств размещения элементов сетки: свойства grid-row, grid-column и grid-area

Свойства grid-row и grid-column являются сокращенными именами для свойств grid-row-start/grid-row-end и grid-column-start/grid-column-end соответственно.

Если заданы два значения, первое (до косой черты) устанавливается для параметра grid-row-start/grid-column-start, второе - для grid-row-end/grid-column-end. Если второе значение опущено, а первое указано в формате пользовательского идентификатора, то grid-row-end/grid-column-end также устанавливается в пользовательское имя сетки. В противном случае, оно вычисляется в auto.

Для свойства grid-area если указано четыре значения, первое устанавливается для grid-row-start, второе - для grid-column-start, третье - для grid-row-end, четвертое - для grid-column-end.

Если grid-column-end/grid-row-end не указан, а grid-column-start/grid-row-start указан в форме пользовательского имени, то для grid-column-end/grid-row-end также устанавливается значение пользовательского имени линии; в противном случае он установлен на auto.

Когда grid-column-start опущен, а значение grid-row-start указан в форме пользовательского имени, оно устанавливается для всех четырех значений. В противном случае оно устанавливается на auto.

5.3. Переупорядочивание элементов сетки

Свойство order также применяется к элементам сетки. Это влияет на их автоматическое размещение и порядок отрисовки. Свойство должно использоваться только для визуального, а не логического переупорядочения контента.

6. Выравнивание элементов сетки и промежутки между элементами

Для выравнивания элементов сетки можно использовать свойство margin, аналогично, как работает это свойство для блочных элементов.

По умолчанию элементы сетки растягиваются, чтобы заполнить свою область сетки. Тем не менее, если justify-self или align-self вычисляют значение, отличное от stretch или задано margin: auto, элементы сетки будут автоматически изменяться в соответствии с их содержимым.

6.1. Выравнивание с помощью margin: auto

При расчете размеров дорожек сетки margin: auto обрабатываются как 0. Они поглощают положительное свободное пространство, предшествующее выравниванием с помощью свойств выравнивания. Переполняющиеся элементы игнорируют свои автоматические поля и переполнение, как указано в их свойствах выравнивания блоков.

6.2. Выравнивание по оси строки

Элементы сетки могут быть выровнены в направлении оси строки (по горизонтали для LTR-языков) с помощью свойства justify-self или свойства justify-items (заданного для контейнера-сетки).

6.3. Выравнивание по оси столбца

Элементы сетки могут выровнены в направлении, перпендикулярном оси строки с помощью свойства align-self или свойства align-items, заданного для контейнера-сетки.

6.4. Промежутки между элементами сетки: свойства row-gap, column-gap и gap

Свойства row-gap и column-gap (и их сокращенная запись gap), если они указаны в контейнере сетки, определяют промежутки между строками и столбцами сетки. При определении размера дорожки каждый промежуток рассматривается как дополнительная пустая дорожка указанного размера. Дополнительный промежуток также может быть добавлен между дорожками за счет свойств justify-content и align-content.

Промежутки добавляются только между двумя дорожками сетки, то есть они не добавляются перед первой и после последней дорожки.

Свойства не наследуются.