Geekbrains

Специальность: Frontend-программист Цифровые профессии

Тема дипломного проекта: «Профессиональная верстка: HTML и СSS»

ФИО студента: Дьячков Антон Геннадьевич

г. Северодвинск

2023

Содержание дипломного проекта

Введение 4

1. HTML 5
   1. Основы HTML 5
   2. Работа с атрибутами 8
   3. Работа с текстом 9
   4. Работа со ссылками 11
   5. Работа с изображениями 14
   6. Работа с таблицами 15
   7. Работа со списками 18
   8. Полезные ссылки 19
2. CSS 20
   1. Основы CSS 20 20
3. JavaScript
4. Заключение
5. Список использованной литературы и ресурсов
6. Приложения
7. Презентацию к дипломному проекту

Введение

Данный дипломный проект - небольшая практическая работа. В ней предполагается использовать все полученные знания и закрепить теорию практикой, а результат положить в портфолио. Защита дипломного проекта является обязательной итоговой аттестацией, после прохождения технологических специализаций (Frontend-программист. Цифровые профессии) для получения диплома после обучения.

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

Используемые инструменты - HTML, CSS, Git, Visual Studio Code.

Проект представляет из себя сжатую инструкцию по использованию основных инструментов, необходимых для верстки сайтов - HTML, CSS. Выбранная тема призвана дать самое общее представление о вышеуказанных инструментах и возможность использования в практической работе имеющихся сведений для верстки сайтов. Если же 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 5.

1. HTML

1.1. Основы HTML

HTML (от англ. HyperText Markup Language - «язык гипертекстовой разметки») это как известно основной язык Всемирной сети INTERNET. В данном разделе содержатся основные правила языка HTML, описание структуры HTML-страницы, отношения в структуре документа между его элементами.

Гипертекст – система текстовых страниц, соединенных между собой ссылками.

HTML-документ - это текстовый документ, который можно создать в обычном текстовом редакторе (Блокнот) или в специализированном, с подсветкой кода (Notepad++, Visual Studio Code и т.п.). HTML-документ имеет расширение .html. Создав файл с таким расширением можно открыть его через любой веб браузер.

HTML-документ состоит из дерева HTML-элементов и текста. Каждый элемент обозначается в исходном документе начальным (открывающим) и конечным (закрывающим) тегом (с редкими исключениями).

Начальный тег показывает, где начинается элемент, конечный - где заканчивается. Например, как представлено на Рисунке 1.1.

Рисунок 1.1

<p> Какой то текст</p>

Где <p> является открывающим тегом, а </p> - закрывающим. Браузер при этом выведет только все содержимое или контент (текст, название, картинка, видео и т.д.).

Элементы могут вкладываться друг в друга по принципу «матрёшки».

HTML-элементы могут иметь атрибуты (применяемые для всех HTML-элементов, и собственные). Атрибуты позволяют изменять свойства и поведение элемента, для которого они заданы.

Каждому элементу можно присвоить несколько значений class и только одно значение id. Множественные значения class записываются через пробел, <div class="nav top">. Значения class и id должны состоять только из букв, цифр, дефисов и нижних подчеркиваний и должны начинаться только с букв или цифр.

Браузер просматривает (интерпретирует) HTML-документ, выстраивая его структуру и отображая ее в соответствии с инструкциями, включенными в этот файл (CSS). Если разметка правильная, то в окне браузера будет отображена HTML-страница, содержащая HTML-элементы - заголовки, таблицы, изображения и т.д.

Процесс интерпретации начинается прежде, чем веб-страница полностью загружена в браузер. Браузеры обрабатывают HTML-документы последовательно, с самого начала, при этом обрабатывая CSS и соотнося таблицы стилей с элементами страницы.

HTML-документ состоит из двух разделов - заголовка - содержимое элемента <head> и содержательной части - содержимое <body>.

Создадим файл index.html в редакторе кода Visual Studio Code и сформируем самую базовую структуру html-документа как представлено на Рисунке 1.2

Рисунок 1.2

<!DOCTYPE html><!-- Объявление формата документа -->

<html lang="en">

<head><!-- Техническая информация о документе -->

    <meta charset="UTF-8"><!-- Определяем кодировку символов документа -->

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

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

    <title>Document</title><!-- Задаем заголовок документа -->

    <link rel="stylesheet" type="text/css" href="style.css"> <!-- Подключаем внешнюю таблицу стилей (CSS) -->

    <script src="script.js"></script> <!-- Подключаем сценарии JavaScript -->

</head>

<body><!-- Основная часть документа -->

</body>

</html>

Запись «<!-- Объявление формата документа -->» является комментарием и не отображается непосредственно на странице веб сайта и служит пояснением для самого программиста или других разработчиков кода.

Элементы, находящиеся внутри элемента <html>, образуют дерево (**модель) документа**. При этом сам элемент <html> является корневым элементом.

Взаимодействие элементов веб-страницы, происходит в отношениях между элементами, подразделяющиеся на родительские, дочерние и сестринские:

- предок - элемент, который заключает в себе другие элементы (на Рисунке 1.2 предком для всех элементов является <html> и в то же время элемент <body> является предком для всех содержащихся в нем самом элементов);

- потомок - элемент, расположенный внутри одного или более типов элементов (<body> является потомком <html>, а все элементы в нем являются потомками одновременно для <body> и <html>);

- родительский элемент - элемент, связанный с другими элементами более низкого уровня, и находящийся на дереве выше их (на рисунке 1.1  <html> является родительским только для <head> и <body>);

- дочерний элемент - элемент, непосредственно подчиненный другому элементу более высокого уровня (на Рисунке 1.2 все элементы внутри <body> являются дочерними по отношению к нему);

- сестринский элемент - элемент, имеющий общий родительский элемент с рассматриваемым, так называемые элементы одного уровня (на рисунке 1.1 <head> и <body> являются элементами одного уровня).

Кроме элемента html в любом HTML-документе также обязательно присутствуют:

- head (<head>…</head>) – в нем содержится техническая информация о странице: заголовок, описание, ключевые слова для поисковых машин, кодировку и т.д., а введенная в нем информация не отображается в окне браузера, но указывает браузеру, как следует обрабатывать страницу;

- title (<title>…</title>) – отображает текст в строке заголовка веб-браузера и содержит максимально полное описание содержимого веб-страницы;

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

- style (<style>…</style>) - здесь задаются стили, которые используются на странице, но в HTML-документе лучше всего использовать все же CSS;

- link (<link>) - подключение к html-документу других исполняемых файлов – например 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 либо внутри этого элемента, либо во внешнем файле;

- body (<body>…</body>) - здесь располагается все содержимое документа или контент.

Это лишь часть основных элементов стандартной html-страницы.

1.2. Работа с атрибутами

В HTML существует значительное количество различных атрибутов. Многие из них дублируются в различных тегах (например, title - добавляет всплывающую подсказку и используется в большинстве HTML тегов) и запоминаются по мере изучения HTML.

HTML-атрибуты это специальные слова, которые управляют поведением HTML-элемента. Они добавляют дополнительную функциональность, либо меняют поведение элемента по умолчанию. Атрибуты элемента выражаются внутри начального тега элемента.

Некоторые из самых необходимых атрибутов:

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

- id (id="…") - уникальный идентификатор элемента для идентификации элемента;

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

- style (style="color: blue; background: transparent") - содержит объявления стилей CSS, которые следует применить 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-h6. Они позволяют создать заголовки разных размеров (от большего к меньшему) как показано на Рисунке 1.3.

Рисунок 1.3

<h1>Заголовок 1 уровня</h1>

<h2>Заголовок 2 уровня</h2>

<h3>Заголовок 3 уровня</h3>

<h4>Заголовок 4 уровня</h4>

<h5>Заголовок 5 уровня</h5>

<h6>Заголовок 6 уровня</h6>

Напоминают разделы и подразделы в данном дипломном проекте. Они упорядочивают текст, формируя его визуальную структуру. Элементы должны использоваться только для выделения заголовков нового раздела или подраздела. От самого высокого до самого меньшего.

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 как на Рисунке 1.4

Рисунок 1.4

<p> Элемент р - Разбивает текст на отдельные абзацы, отделяя друг от друга пустой строкой.</p>

Дополнительно в HTML можно делать текст более разнообразным при помощи специальных тегов. Можно выделить текст жирным при помощи тега <b> или же сделать его курсивом при помощи тега <em>

HTML-текст представлен в спецификации элементами для форматирования и группировки текста. Данные элементы являются контейнерами для текста и не имеют визуального отображения.

Элементы для форматирования текста несут смысловую нагрузку и обычно задают для текста, заключенного внутрь, стилевое оформление, например, выделяют текст жирным начертанием или отображают его шрифтом другого семейства (свойство font-family).

Грамотно отформатированный текст дает понять поисковым системам, какие слова несут важную смысловую нагрузку, по каким из них предпочтительно ранжировать веб-страницу в поисковой выдаче. Вся текстовая информация, отображаемая на сайте, размещается внутри элемента <body>.

Все HTML-элементы для работы с текстом представлены в Таблице 1.1

Таблица 1.1

|  |  |
| --- | --- |
| Наименование элемента | Описание элемента |
| Элементы для форматирования текста | |
| <b> | Задаёт полужирное начертания шрифта текста. Выделяет текст без акцента на его важность |
| <em> | Отображает шрифт текста курсивом, придавая тексту значимость |
| <i> | Отображает шрифт курсивом |
| <small> | Уменьшает размер шрифта на единицу по отношению к обычному тексту |
| <strong> | Задаёт полужирное начертание шрифта, относится к элементам логической разметки, указывая браузеру на важность текста |
| <sub> | Используется для создания нижних индексов. Сдвигает текст ниже уровня строки, уменьшая его размер |
| <sup> | Используется для создания степеней. Сдвигает текст выше уровня строки, уменьшая его размер |
| <ins> | Выделяет текст в новой версии документа, подчёркивая его |
| <del> | Перечёркивает 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> | Разбивает текст на отдельные абзацы, отделяя друг от друга пустой строкой |
| <br> | Переносит текст на следующую строку, создавая разрыв строки |
| <hr> | Используется для разделения контента на веб-странице. Отображается в виде горизонтальной линии |

1.4. Работа со ссылками

Ссылка - перенаправление пользователя не только на новые страницы, но также на определенное место на странице. Ссылка будет являться активной и отображаться в формате ссылки только при условии прописанного атрибута href. Если не добавить значение в атрибут «href», то при клике на ссылку страница будет перезагружаться. Ссылки создаются с помощью элементов <a>, <area> и <link>.

Ссылки разделяются на ссылки на внешние ресурсы - создаются с помощью элемента <link> и используются для расширения возможностей текущего документа при обработке браузером и гиперссылки - ссылки на другие ресурсы, которые пользователь может посетить или загрузить.

Гиперссылки создаются с помощью элемента <a></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 атрибут href, который задает URl-адрес веб-страницы.

Код, представленный на Рисунке 1.5 даст вид при загрузке браузером - Это сайт Geekbrains и при нажатии на него переадресует пользователя на сайт gb.ru.

Рисунок 1.5

<a href="https://gb.ru ">Это сайт Geekbrains</a>

Ссылка состоит из двух частей -  указателя («Это сайт Geekbrains») и адресной части (https://gb.ru).

Когда же в ссылке указывается только имя файла, браузер предполагает, что файл находится в той же папке, что и документ, содержащий гиперссылку. На практике веб-сайты содержат множество документов, которые размещают в отдельные папки, чтобы ими было легче управлять. 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 ссыла, 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 1.6.

Рисунок 1.6

<a href="#p1">HTML</a> <!--создаём якорь, указав #id элемента-->

<a href="#p2">CSS</a>

<p id="p1">...</p> <!--добавляем соответствующий id элементу-->

<p id="p2">...</p>

При нажатии на созданную ссылку можно сразу перейти к параграфу текста или разделу, к которому он был привязан.

Если же нужно сделать ссылку с одной страницы сайта на определенный раздел другой страницы, то необходимо задать id для этого раздела страницы, а затем добавить его к абсолютному адресу ссылки как представлено на Рисунке 1.7.

Рисунок 1.7

<p id="about me">About me</з>

<a href="https://mysite.ru/css/#about me" class="site" target="\_blank">color</a>

Аналогичным образом можно создать ссылку из изображения поместив элемент <img> внутрь элемента <a>.

По клику можно не только переходить на другие страницы и скачивать файлы, но и совершать звонки на телефоны, отправлять сообщения или звонить по скайпу (Рисунок 1.8).

Рисунок 1.8

ссылка на телефонный номер

<a href="tel:+74951234567">+7 (495) 123-45-67</a>

ссылка на адрес электронной почты

<a href="mailto:example@mail.ru">example@mail.ru</a>

ссылка на скайп (позвонить)

<a href="skype:имя-пользователя?call">Skype</a>

ссылка на скайп (открыть чат)

<a href="skype:имя-пользователя?chat">Skype</a>

ссылка на скайп (добавить в список контактов)

<a href="skype:имя-пользователя?add">Skype</a>

ссылка на скайп (отправить файл)

<a href="skype:имя-пользователя?sendfile">Skype</a>

1.5. Работа с изображениями

Изображения добавляются на веб-страницы с помощью одинарного тега <img>. Вообще использование какой-либо графики делает веб-страницы значительно привлекательнее и позволяет лучше передать суть и содержание веб-документа (Рисунок 1.9).

Рисунок 1.9

<img src="image.png" alt="Какая то картинка">

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).

Кроме них используется и другие атрибуты для задания высоты изображения, его размера и ширины.

Возможные форматы изображений для их загрузки на сайт – JPEG, GIF, PNG, APNG, **SVG, BMP, ICO.**

# 1.6. Работа с таблицами

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

Ячейки таблиц могут содержать любые HTML-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, потому что такой приём не обеспечивает гибкость структуры и адаптивность сайта, существенно увеличивая HTML-разметку.

Таблица создаётся при помощи элемента <table></table>, который является контейнером для элементов таблицы и все элементы должны находиться внутри него (Рисунок 1.10).

Рисунок 1.10

<table>

    <tr><th>HTML</th><th>CSS</th></tr>

    <tr><td>Теги</td><td>Таблицы стилей</td></tr>

</table

По умолчанию таблица и ячейки не имеют видимых границ, и они задаются с помощью свойств CSS в файле style.css (Рисунок 1.12).

Рисунок 1.12

/\* внешние границы таблицы серого цвета толщиной 1px \*/

table {

    border: 1px solid grey;

 }

/\* границы ячеек первого ряда таблицы \*/

th {

    border: 1px solid grey;

 }

 /\* границы ячеек тела таблицы \*/

 td {

    border: 1px solid grey;

 }

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, rowspan, headers.

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 атрибуты colspan (задает количество ячеек, объединенных по горизонтали), rowspan (задает количество ячеек, объединенных по вертикали), headers.

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

Код и отображение кода браузером представлены на Рисунке 1.13 и Рисунке 1.14 соответственно.

Рисунок 1.13

<table>

    <caption>Перечень изделий</caption>

      <tr>

        <th>№ п/п</th>

        <th>Наименование изделия</th>

        <th>Ед. изм.</th>

        <th>Количество</th>

        <th>Цена за ед. изм., руб.</th>

        <th>Стоимость, руб.</th>

      </tr>

      <tr>

        <td>1.</td>

        <td>Изделие № 1</td><td>кг</td><td>15</td><td>10</td><td>150</td>

      </tr>

      <tr>

        <td>2.</td>

        <td>Изделие № 2</td><td>кг</td><td>2</td><td>50</td><td>100</td>

      </tr>

      <tr>

        <td colspan="5" style="text-align:right">ИТОГО:</td><td>250</td>

      </tr>

    </table>

Рисунок 1.14

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Перечень изделий | | | | | |
| **№ п/п** | **Наименование изделия** | **Ед. изм.** | **Количество** | **Цена за ед. изм., руб.** | **Стоимость, руб.** |
| 1. | Изделие № 1 | кг | 15 | 1 | 150 |
| 2. | Изделие № 2 | кг | 2 | 50 | 100 |
| ИТОГО: | | | | | 250 |

# 1.7. Работа со списками

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 элемента в списке потребуется использовать тег li. Если мы хотим вписать один список внутрь другого, то необходимо использовать следующую конструкцию:

<ol>

    <li>HTML</li>

    <li>CSS

        <ul>

            <li>Стили таблиц</li>

            <li>Стили текста</li>

        </ul>

    </li>

    <li>Javascript</li>

</ol>

Получим следующее:

1. HTML
2. CSS
   * Стили таблиц
   * Стили текста
3. Javascript

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)/

1.8 Полезные ссылки:

1. <https://stepik.org/course/100971/syllabus>;
2. <https://html5book.ru/html-html5/>;
3. [http://htmlbook.ru](https://vk.com/away.php?to=http%3A%2F%2Fhtmlbook.ru&post=11193554_493&cc_key=);
4. <https://htmlacademy.ru>;
5. https://code-basics.com/ru/languages/html/

2. CSS

2.1 Основы CSS

В данном разделе рассмотрим основы CSS и понятие селекторов.

CSS (Cascading Style Sheets) - язык таблиц стилей, который позволяет 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 тому прочее. Чтобы создать рабочий CSS-стиль, нужно правильно указать 2 основные части - селектор (целевой элемент, которому назначается стиль) и блок стилей (правила форматирования).

На Рисунке 2.1 селектором выступает тег div. Это означает, что все стили в блоке будут применены к каждому div-элементу на странице.

Рисунок 2.1

div {

   background-color:black;

   width: 200px;

   height: 100px;

}

В фигурных скобках всегда указываются стили. Они должны стоять строго в скобках, а не за их пределами. В представленном примере на выходе получится [div](https://itproger.com/spravka/html5/div) блок c черным фоном, шириной 200 и высотой 100 пикселей.

Все команды имеют свойство, а за ним идёт значение т.е. background-color – свойство, а black – 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. Например, 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 (по умолчанию в папке любого проекта должен быть файл style.css), в котором находится набор CSS-стилей элементов. Файл создаётся в редакторе кода, так же как и HTML-страница. Внутри файла могут содержатся только стили, без HTML-разметки.

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

К каждой веб-странице можно присоединить несколько таблиц стилей, добавляя последовательно несколько элементов <link> (Рисунок 2.2).

Рисунок 2.2

<head>

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

      <link rel="stylesheet" href="css/assets.css" media="all">

</head>

При этом атрибут type="text/css" не является обязательным по стандарту HTML5, поэтому его можно не указывать. Если атрибут отсутствует, по умолчанию используется значение type="text/css".

**Внутренние стили** встраиваются в раздел <head></head> HTML-документа и определяются внутри элемента <style></style> (Рисунок 2.3).

Рисунок 2.3

<head>

  <style>

  h1,

  h2 {

  color: black;

  font-family: "Times New Roman", Georgia, Serif;

  line-height: 1.3em;

  }

  </style>

</head>

<body>

  ...

</body>

Кроме перечисленных внешних и внутренних стилей есть также и встроенные, когда CSS-код пишется в HTML-файле, непосредственно внутри элемента с помощью атрибута style (Рисунок 2.4).

Рисунок 2.4

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

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

Рассмотрим понятие селекторов более подробно.

**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);

- селекторы атрибутов - 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.

Все перечисленные селекторы для более точного отбора элементов для форматирования могут использоваться в комбинации селекторов (a[href][title] - выберет все ссылки, для которых заданы атрибуты href и title)или один и тот же стиль одновременно применить к нескольким элементам (перечисляются через запятую).

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»).

# 2.2. Работа с полями

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. Очистить стили браузеров для всех элементов можно при помощи универсального селектора (\* {margin: 0; padding: 0;}), что рекомендуется делать при создании любого нового проекта.

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.

# 2.3. Блочные и строчные элементы

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>).

### 1. Модель визуального форматирования

HTML-документ организован в виде дерева элементов и текстовых узлов. Модель визуального форматирования CSS представляет собой алгоритм, который обрабатывает HTML-документ и выводит его на экран устройства.

Каждый блок в дереве представляет соответствующий элемент или псевдоэлемент, а текст (буквы, цифры, пробелы), находящийся между открывающим и закрывающим тегами, представляет содержимое текстовых узлов.

Чтобы создать дерево блоков, CSS сначала использует каскадирование и наследование, позволяющие назначить вычисленное значение для каждого css-свойства каждому элементу и текстовому узлу в исходном дереве.

Затем для каждого элемента CSS генерирует ноль или более блоков в соответствии со значением свойства display этого элемента. Как правило, элемент генерирует один основной блок, который представляет самого себя и содержит свое содержимое. Некоторые значение свойства display, например, display: list-item;, генерируют блок основного блока и блок дочернего маркера. Другие, например, display: none;, приводят к тому, что элемент и/или его потомки вообще не генерируют блоки.

Положение блоков на странице определяется следующими факторами:

* размером элемента (с учётом того, заданы они явно или нет);
* типом элемента (строчный или блочный);
* схемой позиционирования (нормальный поток, позиционированные или плавающие элементы);
* отношениями между элементами в DOM (родительский — дочерний элемент);
* внутренними размерами содержащихся изображений;
* внешней информацией (например, размеры окна браузера).

### 2. Блочные элементы и блочные контейнеры

**Блочные элементы** — элементы высшего уровня, которые форматируются визуально как блоки, располагаясь на странице в окне браузера вертикально. Значения свойства display, такие как block, list-item и table делают элементы блочными. Блочные элементы генерируют основной блок, который содержит только блок элемента. Элементы со значением display: list-item генерируют дополнительные блоки для маркеров, которые позиционируются относительно основного блока.

<address>, <article>, <aside>,  
<blockquote>,  
<dd>, <div>, <dl>, <dt>, <details>,  
<fieldset>, <figcaption>, <figure>, <footer>, <form>,  
<h1>-<h6>, <header>, <hr>,  
<li>, <legend>,  
<nav>, <noscript>,  
<ol>, <output>, <optgroup>, <option>,  
<p>, <pre>,  
<section>, <summary>,  
<table>,  
<ul>

Блочные элементы могут размещаться непосредственно внутри элемента <body>. Они создают разрыв строки перед элементом и после него, образуя прямоугольную область, по ширине занимающую всю ширину веб-страницы или блока-родителя.

Блочные элементы могут содержать как строчные, так и блочные элементы, но не оба типа элементов сразу. При необходимости, строки текста, принадлежащие блочному контейнеру, могут быть обёрнуты анонимными контейнерами, которые будут вести себя внутри блока как элементы со значением display: block;, а строчные элементы обёрнуты элементом <p>. Блочные элементы могут содержаться только в пределах блочных элементов.

Элемент <p> относится к блочным элементам, но он не должен содержать внутри себя другой элемент <p>, а также любой другой блочный элемент.

#### Анонимные блоки уровня блока

Как говорилось выше, блочные элементы могут содержать только блочные или только строчные элементы. В случае смешанного контента, когда блочный элемент одновременно содержит текстовое содержимое и другой блочный элемент, алгоритм визуального форматирования добавляет дополнительную обёртку для текстового содержимого — так называемый анонимный блок. Анонимный блок не связан ни с одним элементом. Поскольку такой контейнер не имеет названия, то к нему нельзя применить CSS-стили для оформления. Анонимные блоки наследуют свойства окружающего блока, а не наследуемые свойства принимают первоначальное значение.

### 3. Строчные элементы и строчные контейнеры

**Встроенные (строчные) элементы** генерируют внутристрочные контейнеры. Они не формируют новые блоки контента. Значения свойства display, такие как inline и inline-table делают элементы строчными.

<a>, <area>,  
<b>, <bdo>, <bdi>,  
<cite>, <code>,  
<dfn>, <del>,  
<em>,  
<i>, <iframe>, <img>, <ins>,  
<kbd>,  
<label>,  
<map>, <mark>,  
<s>, <samp>, <small>, <span>, <strong>, <sub>, <sup>,  
<time>,  
<q>,  
<ruby>,  
<u>,  
<var>

Строчные элементы могут содержать только данные и другие строчные элементы. Исключение составляет элемент <a>, который согласно спецификации HTML5 может оборачивать целые абзацы, списки, таблицы, заголовки и целые разделы при условии, что они не содержат другие интерактивные элементы — другие ссылки и кнопки.

#### Анонимные блоки уровня строки

Любой текст, содержащийся непосредственно внутри блочного элемента и не внутри строчного элемента, рассматривается как анонимный строчный элемент. Также как и анонимные блоки уровня блока они наследуют свойства родительского блока, а не наследуемые свойства принимают первоначальное значение.

РИС. 3. АНОНИМНЫЙ БЛОК УРОВНЯ СТРОКИ

### 4. Строчно-блочные элементы

Существует еще одна группа элементов, которые браузер обрабатывает как строчно-блочные {display: inline-block;}. Такие элементы являются встроенным, но для них можно задавать поля, отступы, ширину и высоту.  
<audio>,  
<button>,  
<canvas>,  
<embed>,  
<input>,  
<keygen>,  
<meter>,  
<object>,  
<progress>,  
<select>,  
<textarea>,  
<video>.

### 5. Ширина содержимого: свойство width

Свойство width определяет ширину содержимого блока.

Это свойство не применяется к незамещаемым строчным элементам display: inline;. Ширина содержимого встроенных блоков определяется шириной отображаемого содержимого внутри них. Встроенные блоки сливаются в линейные блоки. Ширина линейных блоков определяется шириной содержащего блока, но может быть уменьшена из-за наличия свойства float.

Отрицательные значения не допускаются.

Свойство не наследуется.

|  |  |
| --- | --- |
| **width** | |
| Значения: |  |
| длина | Ширина элемента задается в единицах длины, например, px, em и т.д. |
| % | Вычисляется относительно ширины содержащего блока. Для абсолютно позиционированных элементов процент вычисляется с учетом ширины области отступов padding содержащего блока. |
| auto | Ширина вычисляется в зависимости от значений других свойств. Значение по умолчанию. |
| inherit | Наследует значение свойства от родительского элемента. |

**Синтаксис**

width: 100px;

width: 10em;

width: 50%;

width: auto;

width: inherit;

CSS

### 6. Минимальная и максимальная ширина: свойства min-width и max-width

Свойства min-width и max-width позволяют ограничивать ширину содержимого до определенного диапазона. Значения не могут быть отрицательными. Для min-width значение по умолчанию 0, для max-width — none.

Свойства не наследуются.

|  |  |
| --- | --- |
| **min-width/max-width** | |
| Значения: |  |
| длина | Задает фиксированную минимальную или максимальную используемую ширину. |
| % | Указывает процент для определения используемого значения. Процент рассчитывается относительно ширины содержащего блока. |
| none | Означает отсутствие ограничений ширины блока. |
| inherit | Наследует значение свойства от родительского элемента. |

**Синтаксис**

min-width: 100px;

min-width: 10em;

min-width: 50%;

min-width: inherit;

CSS

max-width: 500px;

max-width: 20em;

max-width: 80%;

max-width: none;

max-width: inherit;

CSS

### 7. Высота содержимого: свойство height

Свойство height определяет высоту содержимого блока. Это свойство не применяется к незамещаемым строчным элементам. Значения длины не могут быть отрицательными.

Свойство не наследуется.

|  |  |
| --- | --- |
| **height** | |
| Значения: |  |
| длина | Высота области содержимого задается в единицах длины. |
| % | Задает высоту в процентах. Процент рассчитывается относительно высоты содержащего блока. Если высота содержащего блока не указана явно (то есть зависит от высоты содержимого) и этот элемент не является абсолютно позиционированным, значение вычисляется как auto. Для абсолютно позиционированных элементов процент вычисляется с учетом высоты области отступов padding содержащего блока. |
| auto | Высота зависит от значений других свойств. Значение по умолчанию. |
| inherit | Наследует значение свойства от родительского элемента. |

**Синтаксис**

height: 100px;

height: 10em;

height: 50%;

height: auto;

width: inherit;

CSS

### 8. Минимальная и максимальная высота: свойства min-height и max-height

Иногда полезно ограничить высоту элементов определенным диапазоном. Свойства min-height и max-height предлагают эту функциональность.

Свойства не наследуются.

|  |  |
| --- | --- |
| **min-height/max-height** | |
| Значения: |  |
| длина | Задает фиксированную минимальную или максимальную вычисленную высоту в единицах длины. Значения не могут быть отрицательными. |
| % | Указывает процент для определения используемого значения. Процент рассчитывается относительно высоты содержащего блока. Если высота содержащего блока не указана явно (т.е. зависит от высоты содержимого) и этот элемент не является абсолютно позиционированным, процентное значение обрабатывается как 0 для min-height или none для max-height. |
| none | Отсутствие ограничений высоты блока, только для max-height. |
| inherit | Наследует значение свойства от родительского элемента. |

**Синтаксис**

min-height: 100px;

min-height: 2em;

min-height: 50%;

min-height: inherit;

CSS

max-height: 500px;

max-height: 20em;

max-height: 80%;

max-height: none;

max-height: inherit;

CSS

### 9. Расчет высоты строки: свойства line-height и vertical-align

Как описано выше, пользовательские агенты (браузеры) передают блоки встроенного уровня в вертикальный стек линейных блоков. Высота линейного блока определяется следующим образом:

* Высота каждого встроенного прямоугольника в линейном блоке вычисляется. Для замещаемых, inline-block и inline-table элементов это высота их области поля (margin box).
* Блоки уровня строки выравниваются вертикально в соответствии со значением свойства vertical-align. Если они выровнены по верху или по низу, они должны быть выровнены так, чтобы минимизировать высоту линейного блока.

Высота линейного блока — это расстояние между самой верхней и самой нижней частью блока. Пустые встроенные элементы генерируют пустые встроенные блоки, но эти блоки по-прежнему имеют поля, отступы, границы, высоту строки и, таким образом, влияют на эти вычисления также, как и элементы с содержимым.

В элементе уровня блока, содержимое которого состоит из элементов встроенного уровня, свойство line-height определяет минимальную высоту линейных блоков внутри элемента. Минимальная высота состоит из минимальной высоты над базовой линией и минимальной глубины под ней.

Для элементов уровня строки свойство line-height указывает высоту, которая используется при расчете высоты линейного блока.

Отрицательные значения не допустимы.

Свойство наследуется.

|  |  |
| --- | --- |
| **line-height** | |
| Значения: |  |
| normal | Сообщает пользовательским агентам установить «разумное» значение на основе шрифта элемента. Значение по умолчанию. Когда элемент содержит текст, отображаемый более чем одним шрифтом, пользовательские агенты могут определить значение normal в соответствии с наибольшим размером шрифта. |
| длина | Значение задаётся в единицах длины, создавая фиксированное значение высоты строки. Если задать значение меньше единицы, смежные строки будут находить друг на друга. |
| число | Используемое значение свойства — это число, умноженное на размер шрифта элемента. |
| % | Вычисленное значение свойства — это процент, умноженный на вычисленный размер шрифта элемента. |
| inherit | Наследует значение свойства от родительского элемента. |

**Синтаксис**

line-height: normal;

line-height: 2em;

line-height: 1.5;

line-height: 50%;

line-height: inherit;

Свойство vertical-align влияет на вертикальное позиционирование в линейном блоке элементов уровня строки: display: inline и display: table-cell. Значения этого свойства имеют другие значения в контексте таблиц.

Свойство не наследуется.

|  |  |
| --- | --- |
| **vertical-align** | |
| Значения: |  |
| baseline | Выравнивает базовую линию элемента по базовой линии его родителя, совмещая среднюю линию элемента со средней линией родительского элемента. |
| sub | Делает элемент подстрочным (аналогично с элементом <sub>). Величина понижения элемента может меняться в зависимости от браузера пользователя. |
| super | Делает элемент надстрочным (аналогично с элементом <sup>). При этом значения sup и super не меняют размер шрифта, по умолчанию текст надстрочного и подстрочного элемента имеет такой же размер, как и текст родительского элемента. |
| top | Верхний край элемента совмещается с верхним краем самого высокого элемента в линии. |
| text-top | Верхний край элемента совмещается с верхним краем шрифта родительского элемента. |
| middle | Средняя линия элемента (обычно изображения) совмещается с линией, проходящей через середину родительского элемента. |
| bottom | Нижний край элемента совмещается с нижним краем самого низкого элемента в линии. |
| text-bottom | Нижний край элемента совмещается с нижним краем шрифта родительского элемента. |
| % | Не позволяет устанавливать middle, вычисляется как часть line-height элемента, а не его родителя, т.е. если установить значение vertical-align, равное 50% для элемента с line-height равным 20рх, то базовая линия элемента поднимется на 10px. |
| длина | Устанавливает значение в единицах длины, перемещая элемент на заданное расстояние. |
| inherit | Наследует значение свойства от родительского элемента. |

**Синтаксис**

vertical-align: baseline;

vertical-align: sub;

vertical-align: super;

vertical-align: text-top;

vertical-align: text-bottom;

vertical-align: middle;

vertical-align: top;

vertical-align: bottom;

vertical-align: 6em;

vertical-align: 10px;

vertical-align: 25%;

vertical-align: inherit;

CSS

РИС. 5. СВОЙСТВО VERTICAL-ALIGN

### 10. Изменение блочной модели: свойство box-sizing

Свойство box-sizing переключает блочную модель с фиксированных размеров длины и ширины на content-box и border-box. Это влияет на интерпретацию всех свойств, определяющих размеры, включая flex-basis.

Свойство не наследуется.

|  |  |
| --- | --- |
| **box-sizing** | |
| Значения: |  |
| content-box | Это поведение ширины и высоты, как указано в CSS2.1. Заданные ширина и высота (и соответствующие min/max-свойства) применяются к ширине и высоте области содержимого элемента. Поля padding и рамка border элемента располагаются за пределами указанной ширины и высоты. Значение по умолчанию. |
| border-box | Любые padding или border, заданные для элемента, размечаются и отрисовываются внутри указанных значений ширины и высоты. Ширина и высота содержимого вычисляются путем вычитания ширины границ и полей соответствующих сторон из указанных свойств ширины и высоты. Значение auto свойств width и height не зависит от свойства box-sizing и всегда устанавливает размер блока с содержимым. Сумма padding и border не должна превышать заданные значения width и height, в противном случае размер области содержимого будет равен нулю. |
| initial | Устанавливает значение свойства в значение по умолчанию. |
| inherit | Наследует значение свойства от родительского элемента. |

Некоторые HTML-элементы, например, <button>, по умолчанию имеют box-sizing: border-box.

**Синтаксис**

box-sizing: content-box;

box-sizing: border-box;

box-sizing: inherit;

box-sizing: initial;

2.4 позиционирование CSS-

CSS рассматривает макет html-документа как дерево элементов. Уникальный элемент, у которого нет родительского элемента, называется **корневым** элементом. Модуль CSS-позиционирование описывает, как любой из элементов может быть размещен независимо от порядка документа (т.е. извлечен из «потока»).

В CSS2 каждый элемент в дереве документа генерирует ноль или более блоков в соответствии с блочной моделью. Модуль CSS3 дополняет и расширяет схему позиционирования. Расположение этих блоков регулируется:

* размерами и типом элемента,
* схемой позиционирования (нормальный поток, обтекание и абсолютное позиционирование),
* отношениями между элементами в дереве документа,
* внешней информацией (например, размер области просмотра, внутренними размерами изображений и т.д.).

## Схемы позиционирования

* **СОДЕРЖАНИЕ:**
* [1. Содержащий блок](https://html5book.ru/css-position/#containing-block)
* [2. Выбор схемы позиционирования: свойство position](https://html5book.ru/css-position/#position)
* [3. Смещение блока: свойства top, right, bottom, left](https://html5book.ru/css-position/#position-prop)
* [4. Обтекание: свойство float](https://html5book.ru/css-position/#float)
* [5. Управление потоком рядом с плавающими элементами: свойство clear](https://html5book.ru/css-position/#clear)
* [6. Определение контекста наложения: свойство z-index](https://html5book.ru/css-position/#z-index)
* [6.1. Контекст наложения](https://html5book.ru/css-position/#stacking%20context)

В CSS блок элемента может быть расположен в соответствии с тремя схемами позиционирования:

#### Нормальный поток

Нормальный поток включает блочный контекст форматирования (элементы с display block, list-item или table), строчный (встроенный) контекст форматирования (элементы с display inline, inline-block или inline-table), и относительное и «липкое» позиционирование элементов уровня блока и строки.

#### Обтекание

В обтекающей модели блок удаляется из нормального потока и позиционируется влево или вправо. Содержимое обтекает правую сторону элемента с float: left и левую сторону элемента с float: right.

#### Абсолютное позиционирование

В модели абсолютного позиционирования блок полностью удаляется из нормального потока и ему присваивается позиция относительно содержащего блока. Абсолютное позиционирование реализуется с помощью значений position: absolute; и position: fixed;.

Элементом «вне потока» может быть плавающий, абсолютно позиционированный или корневой элемент.

### 1. Содержащий блок

Положение и размер блока(ов) элемента иногда вычисляются относительно некоторого прямоугольника, называемого **содержащим блоком** элемента (containing block). В общих словах, содержащий блок — это блок, который содержит другой элемент. В случае нормального потока корневой элемент html является содержащим блоком для элемента body, который, в свою очередь, является содержащим блоком для всех его дочерних элементов и так далее. В случае позиционирования содержащий блок полностью зависит от типа позиционирования. Содержащий блок элемента определяется следующим образом:

* Содержащий блок, в котором находится корневой элемент, представляет собой прямоугольник — так называемый **начальный содержащий блок**.
* Для некорневого элемента с position: static; или position: relative; содержащий блок формируется краем области содержимого ближайшего родительского блока уровня блока, ячейки таблицы или уровня строки.
* Содержащим блоком элемента с position: fixed; является окно просмотра.
* Для некорневого элемента с position: absolute; содержащим блоком устанавливается ближайший родительский элемент со значением position: absolute/relative/fixed следующим образом:  
  — если предок — элемент уровня блока, содержащим блоком будет область содержимого плюс поля элемента padding;  
  — если предок — элемент уровня строки, содержащим блоком будет область содержимого;  
  — если предков нет, то содержащий блок элемента определяется как начальный содержащий блок.
* Для «липкого» блока содержащим блоком является ближайший предок с прокруткой или окно просмотра, в противном случае.

### 2. Выбор схемы позиционирования: свойство position

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

Свойство не наследуется.

|  |  |
| --- | --- |
| **position** | |
| Значение: |  |
| static | Блок располагается в соответствии с нормальным потоком. Свойства top, right, bottom и left не применяются. Значение по умолчанию. |
| relative | Положение блока рассчитывается в соответствии с нормальным потоком. Затем блок смещается относительно его нормального положения и во всех случаях, включая элементы таблицы, не влияет на положение любых следующих блоков. Тем не менее, такое смещение может привести к перекрытию блоков, а также к появлению полосы прокрутки в случае переполнения.  Относительно позиционированный блок сохраняет свои размеры, включая разрывы строк и пространство, первоначально зарезервированное для него.  Относительно позиционированный блок создает новый содержащий блок для абсолютно позиционированных потомков.  Влияние position: relative; на элементы таблицы определяется следующим образом: Элементы с table-row-group, table-header-group, table-footer-group иd table-row смещаются относительно их обычной позиции в таблице. Если ячейки таблицы занимают несколько строк, смещаются только ячейки начальной строки. table-column-group, table-column не смещает соответствующий столбец и не оказывает визуального влияния. table-caption and table-cell смещаются относительно своего нормального положения в таблице. Если ячейка таблицы охватывает несколько столбцов или строк, то она смещается целиком. |
| absolute | Положение блока (и, возможно, размер) задается с помощью свойств top, right, bottom и left. Эти свойства определяют явное смещение относительно его содержащего блока. Абсолютно позиционированные блоки полностью удаляется из нормального потока, не влияя на расположение сестринских элементов.  Отступы margin абсолютно позиционированных блоков не схлопываются.  Абсолютно позиционированный блок создает новый содержащий блок для дочерних элементов нормального потока и потомков с position: absolute;.  Содержимое абсолютно позиционированного элемента не может обтекать другие блоки. Абсолютно позиционированный блок могут скрывать содержимое другого блока (или сами могут быть скрыты), в зависимости от значения z-index перекрывающихся блоков. |
| sticky | Положение блока рассчитывается в соответствии с нормальным потоком. Затем блок смещается относительно своего ближайшего предка с прокруткой или окна просмотра, если ни у одного из предков нет прокрутки.  «Липкий» блок может перекрывать другие блоки, а также создавать полосы прокрутки в случае переполнения.  «Липкий» блок сохраняет свои размеры, включая разрывы строк и пространство, первоначально зарезервированное для него.  «Липкий» блок создает новый содержащий блок для абсолютно и относительно позиционированных потомков. |
| fixed | Фиксированное позиционирование аналогично абсолютному позиционированию, с отличием в том, что для содержащим блоком устанавливается окно просмотра. Такой блок полностью удаляется из потока документа и не имеет позиции относительно какой-либо части документа. Фиксированные блоки не перемещаются при прокрутке документа. В этом отношении они похожи на фиксированные фоновые изображения.  При печати фиксированные блоки повторяются на каждой странице, содержащим блоком для них устанавливается область страницы. Блоки с фиксированным положением, которые больше области страницы, обрезаются. |
| initial | Устанавливает значение свойства в значение по умолчанию. |
| inherit | Наследует значение свойства от родительского элемента. |

**Синтаксис**

position: static;

position: relative;

position: absolute;

position: sticky;

position: fixed;

position: initial;

position: inherit;

CSS

РИС. 1. РАЗНИЦА МЕЖДУ СТАТИЧНЫМ, ОТНОСИТЕЛЬНЫМ И АБСОЛЮТНЫМ ПОЗИЦИОНИРОВАНИЕМ

### 3. Смещение блока: свойства top, right, bottom, left

Элемент считается позиционированным, если свойство position имеет значение, отличное от static. Позиционированные элементы генерируют позиционированные блоки и могут быть расположены в соответствии со следующими четырьмя физическими свойствами:

|  |  |
| --- | --- |
| **top** | |
| Значение: |  |
| auto | Влияние значения зависит от типа элемента. Значение по умолчанию. |
| длина | Смещение на фиксированном расстоянии от указанного края. Отрицательные значения допускаются. |
| % | Процентные значения вычисляются относительно высоты содержащего блока. Для «липкого» блока — относительно высоты корневого элемента. Отрицательные значения допускаются. |
| initial | Устанавливает значение свойства в значение по умолчанию. |
| inherit | Наследует значение свойства от родительского элемента. |

**Синтаксис**

top: 10px;

top: 2em;

top: 50%;

top: auto;

top: inherit;

top: initial;

CSS

Свойство top задает расстояние, на которое верхний край абсолютно позиционированного блока (с учетом его margin) смещается ниже верхнего края содержащего блока. Для относительно позиционированных блоков определяет смещение относительно верхнего края самого блока (то есть блоку задается позиция в нормальном потоке, а затем смещение от этой позиции в соответствии с этим свойством).

|  |  |
| --- | --- |
| **right** | |
| Значение: |  |
| auto | Влияние значения зависит от типа элемента. Значение по умолчанию. |
| длина | Смещение на фиксированном расстоянии от указанного края. Отрицательные значения допускаются. |
| % | Процентные значения вычисляются относительно ширины содержащего блока. Для «липкого» блока — относительно ширины корневого элемента. Отрицательные значения допускаются. |
| initial | Устанавливает значение свойства в значение по умолчанию. |
| inherit | Наследует значение свойства от родительского элемента. |

**Синтаксис**

right: -10px;

right: .5em;

right: -10%;

right: auto;

right: inherit;

right: initial;

CSS

Свойство right указывает расстояние, на которое правый край абсолютно позиционированного блока (с учетом его margin) смещен влево от правого края содержащего блока. Для относительно позиционированных блоков определяет смещение относительно правого края самого блока.

|  |  |
| --- | --- |
| **bottom** | |
| Значение: |  |
| auto | Влияние значения зависит от типа элемента. Значение по умолчанию. |
| длина | Смещение на фиксированном расстоянии от указанного края. Отрицательные значения допускаются. |
| % | Процентные значения вычисляются относительно высоты содержащего блока. Для «липкого» блока — относительно высоты корневого элемента. Отрицательные значения допускаются. |
| initial | Устанавливает значение свойства в значение по умолчанию. |
| inherit | Наследует значение свойства от родительского элемента. |

**Синтаксис**

bottom: 50px;

bottom: -3em;

bottom: -50%;

bottom: auto;

bottom: inherit;

bottom: initial;

CSS

Свойство bottom указывает расстояние, на которое нижний край блока смещен вверх относительно нижнего края содержащего блока. Для относительно позиционированных блоков определяет смещение относительно нижнего края самого блока.

|  |  |
| --- | --- |
| **left** | |
| Значение: |  |
| auto | Влияние значения зависит от типа элемента. Значение по умолчанию. |
| длина | Смещение на фиксированном расстоянии от указанного края. Отрицательные значения допускаются. |
| % | Процентные значения вычисляются относительно ширины содержащего блока. Для «липкого» блока — относительно ширины корневого элемента. Отрицательные значения допускаются. |
| initial | Устанавливает значение свойства в значение по умолчанию. |
| inherit | Наследует значение свойства от родительского элемента. |

**Синтаксис**

left: 50px;

left: 10em;

left: 20%;

left: auto;

left: inherit;

left: initial;

CSS

Свойство left указывает расстояние, на которое левый край смещен вправо от левого края содержащего блока. Для относительно позиционированных блоков определяет смещение относительно левого края самого блока.

Положительные значения смещают элемент внутрь содержащего блока, а отрицательные — за его пределы.

### 4. Обтекание: свойство float

Обтекание позволяет блокам смещаться влево или вправо на текущей строке. «Плавающий блок» смещается влево или вправо до тех пор, пока его внешний край не коснется края содержащего блока или внешнего края другого плавающего блока. Если имеется линейный блок, внешняя верхняя часть плавающего блока выравнивается с верхней частью текущего линейного блока.

При использовании свойства float для элементов рекомендуется задавать ширину. Тем самым браузер создаст место для другого содержимого. Если для плавающего элемента недостаточно места по горизонтали, он будет смещаться вниз до тех пор, пока не уместится. При этом остальные элементы уровня блока будут его игнорировать, а элементы уровня строки будут смещаться вправо или влево, освобождая для него пространство и обтекая его.

Правила, регулирующие поведение плавающих боков, описываются свойством float.

Свойство не наследуется.

|  |  |
| --- | --- |
| **float** | |
| Значение: |  |
| none | Отсутствие обтекания. Значение по умолчанию. |
| left | Элемент перемещается влево, содержимое обтекает плавающий блок по правому краю. |
| right | Элемент перемещается вправо, содержимое обтекает плавающий блок по левому краю. |
| inherit | Наследует значение свойства от родительского элемента. |

**Синтаксис**

float: left;

float: right;

float: none;

float: inherit;

CSS

Плавающий блок принимает размеры своего содержимого с учетом внутренних отступов и рамок. Верхние и нижние отступы margin плавающих элементов не схлопываются.

Плавающие элементы могут использовать отрицательные отступы margin, чтобы перемещаться за пределы области содержимого их родительского элемента.

Свойство автоматически изменяет вычисляемое (отображаемое в браузере) значение свойства display на display: block для следующих значений: inline, inline-block, table-row, table-row-group, table-column, table-column-group, table-cell, table-caption, table-header-group, table-footer-group. Значение inline-table меняет на table.

Свойство не оказывает влияние на элементы с display: flex и display: inline-flex. Не применяется к абсолютно позиционированным элементам.

РИС. 2. ОБТЕКАНИЕ ЭЛЕМЕНТОВ

### 5. Управление потоком рядом с плавающими элементами: свойство clear

Свойство clear указывает, какие стороны блока/блоков элемента не должны прилегать к плавающим блокам, находящемся выше в исходном документе. В CSS2 и CSS 2.1 свойство применяется только к неплавающим элементам уровня блока.

Свойство не наследуется.

|  |  |
| --- | --- |
| **clear** | |
| Значение: |  |
| none | Означает отсутствие ограничений на положение элемента относительно плавающих блоков. Значение по умолчанию. |
| left | Смещает элемент вниз относительно нижнего края любого плавающего слева элемента, находящемся выше в исходном документе. |
| right | Смещает элемент вниз относительно нижнего края любого плавающего справа элемента, находящемся выше в исходном документе. |
| both | Смещает элемент вниз относительно нижнего края любого плавающего слева и справа элемента, находящемся выше в исходном документе. |
| inherit | Наследует значение свойства от родительского элемента. |

**Синтаксис**

clear: none;

clear: left;

clear: right;

clear: both;

clear: inherit;

CSS

Для предотвращения отображение фона или границ под плавающими элементами используется правило {overflow: hidden;}.

### 6. Определение контекста наложения: свойство z-index

В CSS каждый блок имеет позицию в трех измерениях. В дополнение к горизонтальному и вертикальному положению, блоки выкладываются вдоль оси Z друг над другом. Положение вдоль оси Z особенно важно, когда блоки визуально накладываются друг на друга.

РИС. 3. ПОЛОЖЕНИЕ ЭЛЕМЕНТОВ ВДОЛЬ ОСИ Z

Порядок, в котором дерево документа отрисовывается на экране, описывается с помощью **контекста наложения**. Каждый блок принадлежит одному контексту наложения. Каждый блок в данном контексте наложения имеет целочисленный уровень, который является его положением на оси Z относительно других блоков в том же контексте наложения.

Блоки с более высокими уровнями всегда отображаются перед блоками с более низкими уровнями, а блоки с одинаковым уровнем располагаются снизу вверх в соответствии с порядком следования элементов в исходном документе. Блок элемента имеет ту же позицию, что и блок его родителя, если только ему не присвоен другой уровень свойством z-index.

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

Свойство не наследуется.

|  |  |
| --- | --- |
| **z-index** | |
| Значение: |  |
| auto | Вычисляется в 0. Если для блока задано position: fixed; или это корневой элемент, значение auto также устанавливает новый контекст наложения. Значение по умолчанию. |
| целое число | Определяет положение блока в текущем контексте наложения. Также устанавливает новый локальный контекст наложения. Можно использовать любое целое число, включая отрицательные числа. Отрицательные значения помещают элемент вглубь экрана. |
| inherit | Наследует значение свойства от родительского элемента. |
| initial | Устанавливает значение свойства в значение по умолчанию. |

**Синтаксис**

z-index: auto;

z-index: 0;

z-index: 5;

z-index: 999;

z-index: -1;

z-index: inherit;

z-index: initial;

CSS

#### 6.1. Контекст наложения

Если для элементов свойства z-index и position не заданы явно, контекст наложения равен порядку их расположения в исходном коде и браузер отображает элементы на странице в следующем порядке:

* Корневой элемент <html>, который содержит все элементы веб-странице.
* Блочные элементы, неплавающие и непозиционированные.
* Плавающие float непозиционированные элементы в порядке их расположения в исходном коде.
* Строковые непозиционированные элементы (текст, изображения).
* Позиционированные position элементы в порядке их следования в исходном коде. Последний из них будет расположен на переднем плане.

Свойство z-index создает новый контекст наложения. Оно позволяет изменить порядок наложения **позиционированных элементов**. Элементы будут отображаться на странице в следующем порядке (если для них не заданы свойства, влияющие на контекст наложения — opacity, filter, transform):

* Корневой элемент <html>, который содержит все элементы веб-странице.
* Позиционированные элементы с отрицательным значением z-index.
* Блочные элементы, неплавающие и непозиционированные.
* Плавающие float непозиционированные элементы в порядке их расположения в исходном коде.
* Строковые непозиционированные элементы (текст, изображения).
* Позиционированные элементы со значениями z-index: 0; и z-index: auto;.