# ЗНАЧЕНИЕ ЭЛЕМЕНТОВ , СИМВОЛОВ , СЕЛЕКТОРЫ, КЛАССОВ и ПСЕВДОКЛАССОВ CSS В HTML И РАЗНОЕ, GITHUB

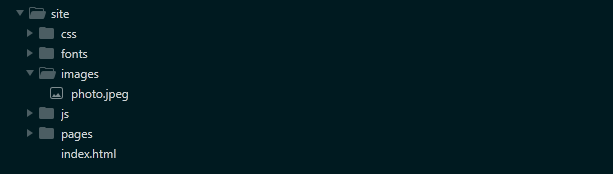
## ****Путь к файлу****

Существует два типа пути: абсолютный и относительный.

**Абсолютный** представляет собой полный адрес (URL) файла или страницы.

**Относительный**путь указывает место расположения картинки относительно файла, в котором он написан.

Например, если файл картинки **image.jpg** находится в той же папке, что и html-файл, то вы указываете "**image.jpg**". Если он находится во вложенной папке images, то есть уровнем ниже, то мы начинаем путь с её названия: "**images/image.jpg**". Если уровнем выше, то начинаем с символа "../", который поднимает вас на один уровень: "**../images/image.jpg**". Относительный путь обычно используется для файлов на сервере.



Предположим, у нас есть **папка сайта** со стандартным набором папок внутри: CSS, fonts, images, JS и файл index.html. В папке images есть картинка **photo.jpeg**. Если её вставить на страницу index.html с помощью тега, то путь к ней начнётся на уровне файла, в котором пишем код. Идём в папку images, находим там файл:

<img src="images/photo.jpeg">

А если мы указываем путь к этой же картинке, находясь в файле CSS (который, соответственно, расположен в папке CSS), то сначала пришлось бы подняться на уровень выше:

<img src="../images/photo.jpeg">

query - это библиотека на писаная на javascript-е.  
В jquery # означает id рассматриваемого элемента, а . - класс.

# это заглушка, в место которой в будущем появиться ссылка. Ведь разработчик может и не знать куда на момент разработки должна ссылаться ссылка.

Чаще всего атрибут **id** используется в качестве селектора для определения стиля элемента в каскадной таблице стилей (CSS), а также в скриптах JavaScript для манипулирования элементом с определенным идентификатором.

CSS («Cascading Style Sheets» или «Каскадные таблицы стилей») – это язык, предназначенный для преобразования внешнего вида страницы на сайте в удобную для восприятия пользователем форму. Как правило, язык CSS используют для описания интернет-страниц, уже написанных на языках HTML либо XHTML. Кроме этого, CSS нередко применяют в совокупности с такими документами XML, как SVG или XUL.Как работает CSS?​

Используя данный язык для оформления документа, он отыскивает в HTML коде нужный элемент, к примеру, заголовок, и задает необходимое свойство (цвет), чтобы присвоить ему значение (синий).

Сейчас существует 7 видов селекторов:для тегов;для классов;для ID;универсальные;атрибутов;для взаимодействия с псевдоклассами;для управления псевдоэлементами.

Виды селекторов в CSS

Селектор по элементам;

Селектор по классам;

Селектор по id;

Контекcтный селектор.

**Селектор**определяет, какой элемент (или элементы) станет объектом определения стилей. **Свойство**селектораотвечает за выбор характеристики для изменений. И наконец, его **значение**помогает определить, как именно что-либо необходимо изменить. Вся эта запись называется **правило CSS**.

**Запись правила CSS** в общем виде выглядит так:

селектор {

свойство-1: значение-1;

свойство-2: значение-2;

}

h1 {

**color**: red;

**background**: blue;

}

Универсальный селектор обозначается звездочкой (\*). Он выбирает все в документе (или внутри родительского элемента, если он связан с другим элементом и комбинатором-потомком).

В CSS селектор класса обозначают именем, которое следует за точкой (.). Идентификатор обозначают именем, которое следует за знаком решетки (#).

Таким образом CSS может напоминать следующий код:

#top {

background-color: #ccc;

padding: 1em

}

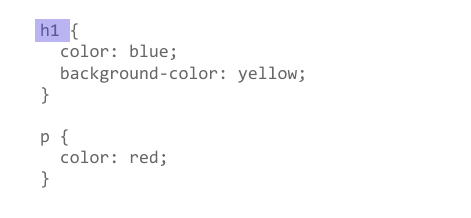
.intro {

color: red;

font-weight: bold;

}

В [CSS](https://developer.mozilla.org/en-US/docs/Glossary/CSS) селекторы используются для нацеливания на [HTML-элементы](https://developer.mozilla.org/en-US/docs/Glossary/HTML) на наших веб-страницах, которые мы хотим стилизовать. Существует широкий спектр селекторов CSS, позволяющих обеспечить детальную точность при выборе элементов для стиля. В этой статье и ее подстатьях мы подробно рассмотрим различные типы, чтобы увидеть, как они работают.



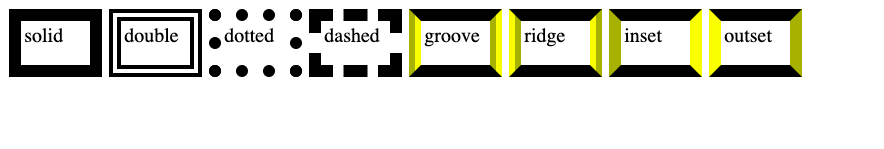
**Запятая в селекторе CSS разделяет несколько селекторов в пределах одного и того же стиля.**

**th {color: red; }   
 td {color: red; }   
 p.red {color: red; }   
 div # firstred {color: red; }**

С этим синтаксисом вы говорите, что хотите, чтобы теги **th**, теги **td**, теги абзаца с красным классом и тег div с идентификатором firstred all имели стиль красный цвет. Селекторы могут записываться через запятую  **th, td, p.red, div # firstred {color: red; } если у них один и тот же стиль.**

**Stylesheet - Таблица стилей Ряд стилей, сохраненных в файле, которые определяют форму (макет) документа.**

**Стиль границы**(border-style) может принимать восемь значений. У двух самых востребованных — solid и double — особенностей почти нет. Вид точек и тире в браузерах отличается, поэтому значения **dotted** и **dashed** нужно использовать с осторожностью. Для **groove, ridge, inset** и **outset** лучше задавать два цвета, так как эти значения связаны с переходом цветов.



По классу

.brd {

border: 4px double black; /\* Параметры границы \*/

background: #fc3; /\* Цвет фона \*/

padding: 10px; /\* Поля вокруг текста \*/

}

Значение атрибута тега **<link>** — rel остаётся неизменным независимо от кода, как приведено в данном примере. Значение href задаёт путь к CSS-файлу, он может быть задан как относительно, так и абсолютно. Заметьте, что таким образом можно подключать таблицу стилей, которая находится на другом сайте.

HTML тег **B** создает жирный текст. Закрывающий тег обязателен.

Тег <noscript>

**Нельзя забывать, что у многих пользователей все еще медленный интернет, в том числе 3G, и многие просто не захотят пользоваться сайтом, который долго загружается. Используйте тег <noscript>, объявленный в header. Подробнее**[**здесь**](https://puzzleweb.ru/html/tag_noscript.php)**. Дополнительно можно изучить вопросы – поддержка ретины, отсутствие багов при увеличенном шрифте и др.**

|  |  |
| --- | --- |
|  | <script type="text/javascript">    document.write("Hello World!")  </script>    <noscript>Ваш браузер не поддерживает JavaScript!</noscript> |

В соответствии со значениями атрибута align – left, right, center, justify и char – броузер выравнивает содержимое каждой ячейки в строке полевому или правому краю ячейки, по ее центру, по обоим краям или по определенному символу соответственно. Атрибуту align можно

присвоить значения left, right, center, приказывая броузеру выровнять содержимое ячейки по левому или правому краю ячейки или по ее центру.

font-size – размер шрифта, в частности, определяющий высоту букв. Значение-число интерпретируется как множитель относительно размера шрифта.

*HTML* состоит из **элементов**, которые обозначаются **тегами** — открывающая и закрывающая скобка**<>**. У любого элемента есть как минимум один тег, но чаще их два — открывающий <> и закрывающий </>. Между тегами находится текстовое содержимое. Необязательно запоминать сразу все теги, можно пользоваться подсказками из сети. Вот так выглядит элемент параграфа: <p>Текст параграфа, который увидит пользователь</p>,<h1>,<em> и тд.

У элементов могут быть **атрибуты** — дополнительные настройки, которые пишутся внутри открывающего тега. Есть элементы, без которых ваша страница может неправильно отображаться в браузерах у пользователей. С этих элементов должно начинаться создание любого HTML-документа.

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Элементы** | **Описание** |

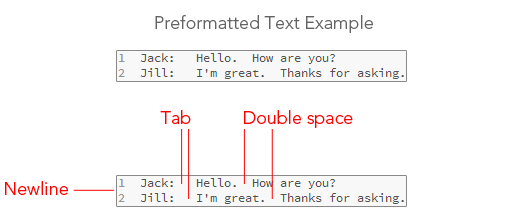
Alt <img> Указывает альтернативный текст, если исходный элемент не может быть отображен браузером

HTML-элементы могут быть вложены друг в друга, и тогда внутренние элементы называют **потомками** или дочерними, а те, в которые они вложены — **родителями**. Это делается для оформления с помощью CSS, для семантики или для использования одного элемента в другом. Отступы упрощают чтение кода и не влияют на результат, но их стоит ставить, чтобы упростить работу с кодом. Отступы состоят из пробелов и табов: пробелов может быть 2 или 4 — это зависит от принятых в компании правил оформления.

## Как работает <pre>

В HTML документе элемент <pre> отображает предварительно отформатированный текст. Это означает, что отступы, сделанные с помощью tab, двойные пробелы, переносы строки и другие типографские символы будут сохранены внутри элемента <pre>.

По умолчанию браузеры отображают содержимое элемента <pre>, с помощью моноширинных шрифтов, таких как Courier или Monaco. Это обычное дело для вывода кода.

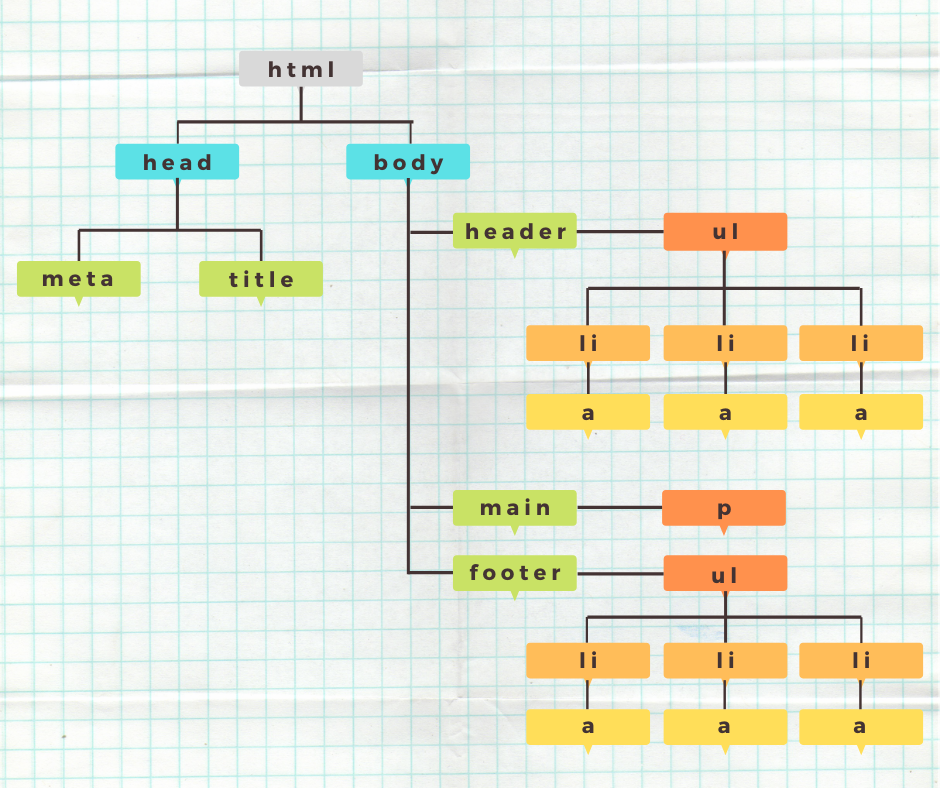


Если бы вы разместили тот же текст в другом элементе, скажем div то, все ваши двойные пробелы, переносы строки и отступы просто игнорировались бы. Таким образом, даже в вашем коде текст будет выглядеть следующим образом:

|  |  |
| --- | --- |
| 1 | <div> |
| 2 | Jack:   Hello.  How are you? | |

|  |  |  |
| --- | --- | --- |
| 3 | Jill:   I'm great.  Thanks for asking. | |
|  | 4 </div>  В результате мы увидим следующее:  https://ruseller.com/lessons/les2106/images/1412040701pre-1.png  Тот же текстовый блок внутри <pre> будет отображён со шрифтом одного размера и со всем вашим дополнительными пробелами и прочими деталями:   |  |  | | --- | --- | | 1 | <pre> | | 2 | Jack:   Hello. How are you? | |  |  |  |  | | --- | --- | --- | | 3 | Jill:   I'm great.  Thanks for asking. | | | 4 | </pre> |   https://ruseller.com/lessons/les2106/images/1412040701pre-2.png Правильная разметка в исходном коде Элемент <pre> должен использоваться для текста, который имеет типографское форматирование. Поскольку это влияет на значение содержимого, например в стихах, транскрипциях и, конечно же, в программном коде.  Если вы хотите отобразить блок исходного кода HTML документа, то используйте элемент <code>, вложенный в элемент <pre>. Это позволит поисковым ботам, социальных приложениям, RSS ридерам сразу же понять, что перед ними программный код.  Пример отображения JavaScript кода:  [view source](https://ruseller.com/lessons.php?id=2106&rub=1#viewSource)  [print](https://ruseller.com/lessons.php?id=2106&rub=1#printSource)[?](https://ruseller.com/lessons.php?id=2106&rub=1#about)   |  |  | | --- | --- | | 1 | <pre><code>// Logs "Hello World!" | | 2 | // in the browser's developer console | |  |  |  | | --- | --- | | 3 | console.log(<code>&quot;</code>Hello World!<code>&quot;</code>);</code></pre> |  Использование вложенных элементов HTML Вы можете использовать другие HTML элементы внутри <pre> и, тем не менее, они будут отображаться корректно.  Для демонстрации концепции, я разметил некоторые em и strong элементов, а так же изменил цвет текста, используя атрибут style.  [view source](https://ruseller.com/lessons.php?id=2106&rub=1#viewSource)  [print](https://ruseller.com/lessons.php?id=2106&rub=1#printSource)[?](https://ruseller.com/lessons.php?id=2106&rub=1#about)   |  |  | | --- | --- | | 1 | <pre><code>var total = 0; | | 2 | <em style="color: green;">// Add 1 to total and display in a paragraph</em> | |  |  |  | | --- | --- | | 3 | <strong style="color: blue;">document.write('&lt;p&gt;Sum: ' + (total + 1) + '&lt;/p&gt;');</strong></code></pre> |   https://ruseller.com/lessons/les2106/images/1412040701pre-3.png  Тег **<nav>** задает навигацию по сайту. Если на странице несколько блоков ссылок, то в **<nav>** обычно помещают приоритетные ссылки. Также допустимо использовать несколько тегов **<nav>** в документе. Запрещается вкладывать **<nav>** внутрь **<address>**. Синтаксис **<nav>**ссылки**</nav>**  **Header** -  - nav –  -… |
|  | *HTML* состоит из **элементов**, которые обозначаются **тегами** — открывающая и закрывающая скобка**<>**. У любого элемента есть как минимум один тег, но чаще их два — открывающий <> и закрывающий </>. Между тегами находится текстовое содержимое. Необязательно запоминать сразу все теги, можно пользоваться подсказками из сети. Вот так выглядит элемент параграфа: <p>Текст параграфа, который увидит пользователь</p>,<h1>,<em> и тд.  У элементов могут быть **атрибуты** — дополнительные настройки, которые пишутся внутри открывающего тега. Есть элементы, без которых ваша страница может неправильно отображаться в браузерах у пользователей. С этих элементов должно начинаться создание любого HTML-документа.   |  |  |  | | --- | --- | --- | | **Атрибут** | **Элементы** | **Описание** |   Alt <img> Указывает альтернативный текст, если исходный элемент не может быть отображен браузером ОписаниеАтрибут wrap говорит браузеру, как осуществлять перенос текста в поле <textarea> и в каком виде отправлять данные на сервер. Если этот атрибут отсутствует, текст в поле набирается одной строкой, когда число введенных символов превышает ширину области, появляется горизонтальная полоса прокрутки. Нажатие кнопки Enter переносит текст на новую строку, и курсор устанавливается у левого края поля.Синтаксис **<textarea** wrap="soft | hard"**>**  **</textarea>**  **wrapper -- Компьютерная техника: родное приложение, а не адаптированное с других платформ**  HTML-элементы могут быть вложены друг в друга, и тогда внутренние элементы называют **потомками** или дочерними, а те, в которые они вложены — **родителями**. Это делается для оформления с помощью CSS, для семантики или для использования одного элемента в другом. Отступы упрощают чтение кода и не влияют на результат, но их стоит ставить, чтобы упростить работу с кодом. Отступы состоят из пробелов и табов: пробелов может быть 2 или 4 — это зависит от принятых в компании правил оформления. HTML тег <style> Элемент <style> (от англ. "style" &#8210; «стиль») задаёт CSS стили для HTML-документа, то есть то, как будут отображаться отдельные элементы и вся страница в целом. На одной веб-странице можно использовать сколько угодно элементов <style> .  **Примечание:** Если атрибут scoped не установлен, то элемент <style> должен располагаться внутри элемента head.  **Совет:** Если вы хотите подключить внешний файл с таблицей стилей, используйте для этого тег link, он создает связь между текущим документом и внешним ресурсом.  **Совет:** Если на странице используется код JavaScript, его обычно добавляют после таблиц стилей. Программы JavaScript часто полагаются на таблицы CSS, поэтому, добавляя таблицы стилей первыми, вы гарантируете, что код JavaScript будет располагать всей необходимой для своего выполнения информацией.  Тег viewport. По определению viewport - это видимая пользователю область сайта без прокрутки. Что такое viewport? Viewport - это видимая пользователю область веб-страницы. Значением атрибута является целое неотрицательно число от 233 до 10000 пикселей или константа device-height , которая задаёт высота страницы в соответствии с размером экрана.  <input>¶ Тег <input> (от англ. input — ввод) является одним из разносторонних элементов формы и позволяет создавать разные части интерфейса и обеспечивать взаимодействие с пользователем. Главным образом <input> предназначен для создания текстовых полей, различных кнопок, переключателей и флажков. Произвольная текстовая строка, которая служит именем параметра.  Теги <dl> </dl> определяют список определений:   |  | | --- | | <dl> <dt>Австралия</dt> <dd>Шестой по величине материк</dd> <dt>Мадагаскар</dt> <dd>Остров в западной части Индийского океана</dd> </dl> |   Результат:  Австралия  Шестой по величине материк  Мадагаскар  Остров в западной части Индийского океана  Спецификация **XHTML** также содержит теги <dl> </dl>.  Список определений состоит из терминов и определений к ним.  Тэг **<dt>** определяет термин, а **<dd>** задает определение к нему.  [**<del>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/del) — вычёркивает текст, показывает, что информация устарела, но важна для сравнения, например, цена товара в магазине.https://lms.skillfactory.ru/assets/courseware/v1/2a2165b8f702a6756d0a53989718ca50/asset-v1:Skillfactory+FR+2020+type@asset+block/SF_FR_A1_HTML_Del.png  <span>7 999 **&#8381;**</span>  <del>9 999 **&#8381;**</del>  [**<sup>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/sup) и [**<sub>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/sub) — верхний и нижний регистры, полезны для сносок и формул. Пример: 8th, H2O.  [**<q>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/q) и [**<blockquote>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/blockquote) — элементы для цитат, в первом случае внутри текста, во втором — для отдельного блока текста.  [**<br>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/br) — перенос строки, после него начнется новая строка, не стоит использовать для оформления блоков на странице. Перенос строки стоит использовать на уровне текста внутри параграфа, например, для стихов или строчек с адресом компании.  [**<hr>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/hr) — создаёт черту, которая отделяет содержимое на уровне параграфов. Можно использовать, чтобы обозначить изменение темы.  [**<ol>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/ol) — **нумерованный список**, здесь важен порядок, последовательность действий/пунктов, которые нельзя менять местами, нумеруется числами.  [**<li>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/li) — **пункт списка**, используется в списках любого типа.  [**<ul>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/ul) — **маркированный список**, здесь последовательность не важна.  [**<pre>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/pre) — показывает отформатированное содержимое с любым количеством **пробелов**, например, когда автору сайта нужно на странице показать код с отступами. Если вы захотите отобразить на странице HTML-код, то нужно заменить все скобки и некоторые знаки спецсимволами.  <pre>  Асфальт — стекло.  Иду и звеню.  Леса и травинки —  сбриты.  На север  с юга  идут авеню,  на запад с востока —  стриты.  </pre> |





К самым распространённым **блочным** элементам можно отнести следующие:

[<html>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/html) — документ HTML;

[<body>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/body) — тело документа;

[<div>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/div) —  блок;

[<p>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/p) —  параграф;

[<h1–h6>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements) —  заголовки от самого большого до самого малого;

<[ul](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/ul" \t "_blank), [ol](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/ol" \t "_blank), [dl](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/dl" \t "_blank)> —  маркированный, нумерованный список и список определений;

[<form>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/form) —  форма для обмена информацией между клиентом и сервером.

## 1.4 Специальные символы

 Добавлено в закладки

Текст на веб-странице можно оборачивать в разные теги в зависимости от задачи. Рассмотрим несколько редких тегов:

[**<del>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/del) — вычёркивает текст, показывает, что информация устарела, но важна для сравнения, например, цена товара в магазине.

<span>7 999 **&#8381;**</span>

<del>9 999 **&#8381;**</del>

[**<sup>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/sup) и [**<sub>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/sub) — верхний и нижний регистры, полезны для сносок и формул. Пример: 8th, H2O.

<p>Теорему Пифагора можно выразить с помощью уравнения: a<sup>2</sup> + b<sup>2</sup> = c<sup>2</sup>.</p>

<p>Любимая молекула разработчиков C<sub>8</sub>H<sub>10</sub>N<sub>4</sub>O<sub>2</sub>, также известная как "кофеин".</p>

[**<q>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/q) и [**<blockquote>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/blockquote) — элементы для цитат, в первом случае внутри текста, во втором — для отдельного блока текста.

Вот пример содержимого, для которого можно использовать элемент **<q>**, чтобы вставить цитату в текст пункта списка:

Русский писатель, философ и общественный деятель XIX века Владимир Одоевский в незаконченном утопическом романе «4338-й год», написанном в 1837 году, похоже, первым предсказал появление современных блогов и интернета: в тексте романа есть строки: "между знакомыми домами устроены магнетические телеграфы, посредством которых живущие на далёком расстоянии общаются друг с другом".

<ul>

<li>Русский писатель, философ и общественный деятель XIX века Владимир Одоевский в незаконченном утопическом романе «4338-й год», написанном в 1837 году, похоже, первым предсказал появление современных блогов и интернета: в тексте романа есть строки: <q>между знакомыми домами устроены магнетические телеграфы, посредством которых живущие на далёком расстоянии общаются друг с другом</q>.

</li>

</ul>

Пример **цитаты** как отдельного блока на странице. Здесь весь текст цитаты — это содержимое элемента <blockquote>:

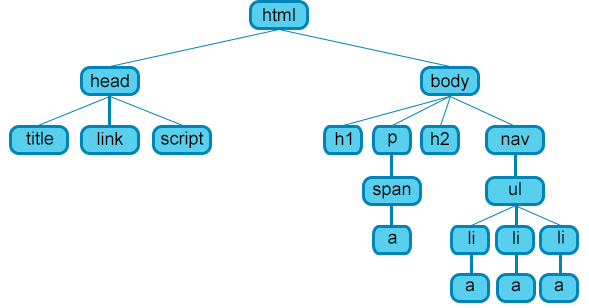
<p>В 1991 году Всемирная паутина стала общедоступна в интернете, а в 1993 году появился знаменитый веб-браузер NCSA Mosaic. Всемирная паутина набирала популярность.</p>

<blockquote>Можно считать, что существуют две ясно различимые эры в истории Web: [до браузера Mosaic] Марка Андриссена и после. Именно сочетание веб-протокола от Тима Бернерс-Ли, который обеспечивал коммуникацию, и браузера (Mosaic) от Марка Андриссена, который предоставил функционально совершенный пользовательский интерфейс, создало условия для наблюдаемого взрыва (интереса к Веб). За первые 24 месяца, истёкшие после появления браузера Моsaic, Web прошёл стадию от полной неизвестности (за пределами считанного числа людей внутри узкой группы учёных и специалистов лишь одного мало кому известного профиля деятельности) до полной и абсолютно везде в мире его распространённости.</blockquote>

По умолчанию браузеры оформляют текст, обернутый **<q>** , кавычками, а **<blockquote>** — отодвигают от левого края. Эти стили оформления можно будет менять.

Некоторые символы нельзя просто так написать в текстовом содержимом HTML-элемента, потому что браузер подумает, что это часть кода, например, угловые скобки, кавычки. Чтобы показать их пользователю, нужно использовать **специальные символы**, которые браузер расшифрует в нужный вам знак. Например:

* + **&lt;** — левая угловая скобка, начало тега (<),
  + **&gt;**— угловая скобка, конец тега (>),
  + **&quot;** — двойная кавычка (“). Зачастую в качестве двойных кавычек также используют **&laquo;** («) — открывающая двойная кавычка и **&raquo;** (») — закрывающая двойная кавычка
  + **&amp;** — знак амперсанда (&)
  + **&ndash**; — среднее тире (–)
  + **&mdash;** — длинное тире (—),
  + **&nbsp;**— неразрывный пробел. «Неразрывный» означает, что подобный пробел будет игнорироваться переносом строки документа, и словосочетание будет всегда на одной строке.
* Нумерованные и маркированные списки могут находится друг в друге, вложенность при этом не ограничена, например:
* <ul>
* <li>Пункт первый</li>
* <li>
* Пункт второй
* <ol>
* <li>Подпункт первый</li>
* <li>Подпункт второй</li>
* </ol>
* </li>
* <li>Пункт третий</li>
* </ul>
* [**<pre>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/pre) — показывает отформатированное содержимое с любым количеством **пробелов**, например, когда автору сайта нужно на странице показать код с отступами. Если вы захотите отобразить на странице HTML-код, то нужно заменить все скобки и некоторые знаки спецсимволами.
* <pre>
* Асфальт — стекло.
* Иду и звеню.
* Леса и травинки —
* сбриты.
* На север
* с юга
* идут авеню,
* на запад с востока —
* стриты.
* </pre>
* Также для отображения кода можно использовать отдельный тег **code**, но он не будет показывать все пробелы и их придётся отмечать вручную:



 Простейшая структура веб-страницы

Чтобы разобраться во взаимодействии элементов веб-страницы, необходимо рассмотреть так называемые **«родственные отношения»**между элементами. Отношения между множественными вложенными элементами подразделяются на родительские, дочерние и сестринские.

**Предок**- элемент, который заключает в себе другие элементы. На рисунке 1 предком для всех элементов является . В то же время элемент является предком для всех содержащихся в нем тегов:

# ,

, ,

# и т.д.

**Потомок**- элемент, расположенный внутри одного или более типов элементов. Например, является потомком , а элемент

Является потомком одновременно для и .

**Родительский элемент**- элемент, связанный с другими элементами более низкого уровня, и находящийся на дереве выше их. На рисунке 1 и . Тег

Является родительским только для .

**Дочерний элемент**- элемент, непосредственно подчиненный другому элементу более [высокого уровня](https://socscan.ru/shvejjnye-mashiny/zvukovye-effekty-studii-pond-5-pond5-otlichnyi-videostok-s.html). На рисунке 1 только элементы

# ,

|  |  |
| --- | --- |
| **Атрибут** | **Описание, принимаемое значение** |
| crossorigin | Указывает, должен ли использоваться CORS (технология браузеров, которая позволяет предоставить веб-странице доступ к ресурсам другого домена) при извлечении изображения с сайта. anonymous — в кросс-доменный запрос браузер автоматически добавляет заголовок Origin, содержащий имя домена, с которого осуществлён запрос. Если сервер не ответит CORS-заголовком Access-Control-Allow-Origin: \* (или имя домена вместо звёздочки), то загрузка изображения будет заблокирована. use-credentials — если сервер не предоставляет учётные данные с помощью Access-Control-Allow-Credentials: true , то загрузка изображения будет заблокирована. |
| href | Основной атрибут тега, в качестве значения выступает путь к файлу со стилями. |
| hreflang | Определяет язык текста в документе, на который идет ссылка. |
| media | Определяет тип устройства, к которым должен быть применен ресурс ссылки. |
| nonce | Генерируемая случайным образом на сервере строковая переменная, которая устанавливает правила использования встроенных стилей с целью защиты контента. Значение атрибута — строка текста. |
| rel | Атрибут определяет отношения между текущим документом и документом, на который идет ссылка. alternate — ссылка на тот же документ, но в другом формате (например, страницы для печати, перевод, зеркало, лента в формате RSS или Atom), .   archives — указывает на то, что документ по ссылке представляет исторический интерес. Ссылка может указывать на коллекцию записей, документов и других материалов. author ссылка на страницу об авторе документа или на страницу с контактными данными автора. bookmark ссылка на ближайшего предка статьи, являющегося связующим звеном, или на раздел статьи, наиболее тесно связанных с элементом, если нет предка. external используется для указания того, что страница на которую ведет ссылка не являются частью данного сайта. first указывает ссылку, ведущую на первый документ из последовательности документов. help ссылка на документ со справкой. icon определяет путь к иконке, которая будет использована для текущего документа. last указывает ссылку, ведущую на последний документ в последовательности документов. license ссылка на сведения об авторских правах для документа. next указывает, что этот документ является частью серии, и что ссылка ведет на следующий документ в этой серии.  nofollow указывает на то, что ссылка не одобрена автором страницы или что ссылка носит коммерческий характер. noreferrer указывает на то, что заголовок запроса клиента, содержащий url источника запроса, не должен передаваться при переходе по ссылке. pingback указывает адрес пингбэк-сервера, что дает возможность для блога автоматически оповещать сайты, ссылающиеся на него. prefetch указывает, что следует заранее кэшировать файл, на который ведет ссылка. prev указывает, что этот документ является частью серии, и что ссылка ведет на предыдущий документ в этой серии.  search указывает, что ссылаемый документ содержит интерфейс поиска и связанных с ним ресурсов. sidebar указывает, что ссылаемый документ, если это возможно, будет показан в дополнительном контексте браузера, и некоторые браузеры при щелчке по гиперссылке открывают окно для добавления ссылки в панель закладок. stylesheet ссылка на внешний файл, который будет использоваться в качестве таблицы стилей для данного документа. tag указывает на то, что метка, на которую ведет гиперссылка, относится к данному документу. up указывает, что страница является частью иерархической структуры, и что гиперссылка ведет на более высокий уровень ресурса в структуре. |
| sizes | Указывает размер иконок для визуального отображения. Атрибут sizes используется только совместно с rel="icon" , и может принимать следующий значения: ширинах высота - определяет список размеров, разделенных пробелами, каждый размер должен быть в формате - ширинах высота (размеры иконки задаются в пикселях), например: ; any - иконка может масштабироваться до любого размера. |
| title | Определяет заголовок ссылки или название набора альтернативных таблиц стилей. Значение атрибута — текст. |
| type | Определяет MIME-тип документа, на который идет ссылка. В данном случае он принимает значение "text/css" . |

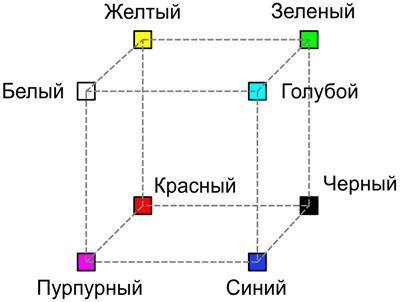
Контейнером называют элемент html-страницы, который предназначен для выделения определенного ее фрагмента. Он может иметь одно из трех значений: LEFT (по левому краю), RIGHT (по правому краю) и CENTER (по центру). По умолчанию – CENTER. Значение этого атрибута – количество объединяемых ячеек.

nline-block Это значение генерирует блочный элемент, который обтекается другими элементами веб-страницы подобно строчному элементу. понимает значение block как list-item ;значение inline-block применяется только для строчных элементов, с блочными элементами работает некорректно. display : value ; Где value может принимать значения: block - блочный элемент (автоматически создает перевод строки) inline - встроенный элемент (не создает перевода строки); inline-block - производная от inline.

ЦВЕТА.

Альфа-канал, также известный как ***маска-канал***, — это просто способ объединить переходную прозрачность с изображением. Формат [GIF](https://ru.wikipedia.org/wiki/GIF) поддерживает простую бинарную прозрачность (когда любой пиксель может быть либо полностью прозрачным, либо абсолютно непрозрачным).

Чаще всего модель строится так, чтобы куб был единичным. Точки, соответствующие базовым цветам, расположены в вершинах куба, лежащих на осях: красный – (1;0;0), зелёный – (0;1;0), синий – (0;0;1). При этом вторичные цвета (полученные смешением двух базовых) расположены в других вершинах куба: голубой - (0;1;1), пурпурный - (1;0;1) и жёлтый – (1;1;0). Чёрный и белые цвета расположены в начале координат (0;0;0) и наиболее удалённой от начала координат точке (1;1;1). Рис. показывает только вершины куба.



Счет в шестнадцатеричной системе происходит аналогично счету в десятичной. Давайте попробуем считать и записывать числа конструируя их из имеющихся шестнадцати цифр:

*Ноль*- **0**;  
*Один*- **1**;  
*Два*- **2**;  
...  
и так далее…  
...  
*Восемь*- **8**;  
*Девять*- **9**;  
*Десять*- **a**;  
*Одиннадцать*- **b**;  
*Двенадцать*- **c**;  
*Тринадцать*- **d**;  
*Четырнадцать*- **e**;  
*Пятнадцать*- **f**;

А что делать дальше? Все цифры кончились. Как же изобразить число Шестнадцать? Поступим аналогично тому как мы поступали в десятичной системе. Там мы вводили понятие десятка, здесь же введем понятие "шестнадцать" и скажем, что шестнадцать - это одина "шестнадцать" и ноль единиц. А это уже можно и записать - "10 16 ".

Итак, *Шестнадцать*- **10**16 (одна "шестнадцать", ноль единиц)  
*Семнадцать*- **11**16 (одна "шестнадцать", одна единица)  
...  
и так далее…  
...  
*Двадцать пять*- **19**16 (одна "шестнадцать", девять единиц)  
*Двадцать шесть*- **1a**16 (одна "шестнадцать", десять единиц)  
*Двадцать семь*- **1b**16 (одна "шестнадцать", одинадцать единиц)  
...  
и так далее…  
...  
*Тридцать*- **1e**16 (одна "шестнадцать", четырнадцать единиц)  
*Тридцать один*- **1f**16 (одна "шестнадцать", пятнадцать единиц)  
*Тридцать два*- **20**16 (две "шестнадцать", ноль единиц)  
*Тридцать три*- **21**16 (две "шестнадцать", одна единица)  
...  
и так далее…  
...  
*Двести пятьдесят пять*- **ff**16 (пятнадцать по "шестнадцать", пятнадцать единиц)

*Двести пятьдесят шесть*- **100**16 (одна "Двести пятьдесят шесть", ноль по "шестнадцать", ноль единиц)  
*Двести пятьдесят семь*- **101**16 (одна "Двести пятьдесят шесть", ноль по "шестнадцать", одна единица)  
*Двести пятьдесят восемь*- **102**16 (одна "Двести пятьдесят шесть", ноль по "шестнадцать", две единицы)  
...  
и так далее...  
...

**Шестнадцатеричная система счисления имеет алфавит, состоящий из 16 цифр:**

## 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, b, c, d, e, f.

При записи числа в шестнадцатеричной системе для записи цифр обозначающих числа 10, 11, 12. 13, 14. 15 используются соответственно буквы А, В, С, D, E, F.

**Перевод чисел из шестнадцатеричной системы в десятичную**

Перевести любое шестнадцатеричное число в десятичное можно по уже известной формуле

**Примеры.**

АЕ07 16 =10∙16 3 +14∙16 2 +0∙16 1 +7∙16 0 =44551 10 .

100 16 =1∙16 2 +0∙16 1 +0∙16 0 =256 10 .

58 16 =5∙16 1 +8∙16 0 =.88 10 .

2А 16 =2∙16 1 +10∙16 0 =42 10 .

Перевод числа из десятичной системы в шестнадцатеричную осуществляется также, как в двоичную.

**Перевод чисел из шестнадцатеричной системы в двоичную и обратно**

Перевести любое шестнадцатеричное число в двоичное можно следующим образом. Каждая цифра шестнадцатеричной записи числа записывается четырехзначным двоичным числом - *тетрадой*. После этого нули, стоящие слева, можно отбросить.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
|  | 2) 2A= 0010 1010 2 = 101010 2 . | 3) 58 16 = 0101 1000 2 = 1011000 2 . |

И наоборот, перевести любое двоичное число в шестнадцатеричное можно аналогичным образом. Каждые четыре двоичные цифры, считая справа налево, записываются одной шестнадцатеричной цифрой. Эти цифры располагаются также справа налево.

**Примеры.**

2. 101010 2 = 10 1010 2 = 2A.

3. 1011000 2 = 101 1000 2 = 58 16 .

HSL, HLS или HSI — цветовая модель, в которой цветовыми координатами являются тон, насыщенность и светлота. Следует отметить, что HSV и HSL — две разные цветовые модели.[Википедия](https://ru.wikipedia.org/wiki/HSL)

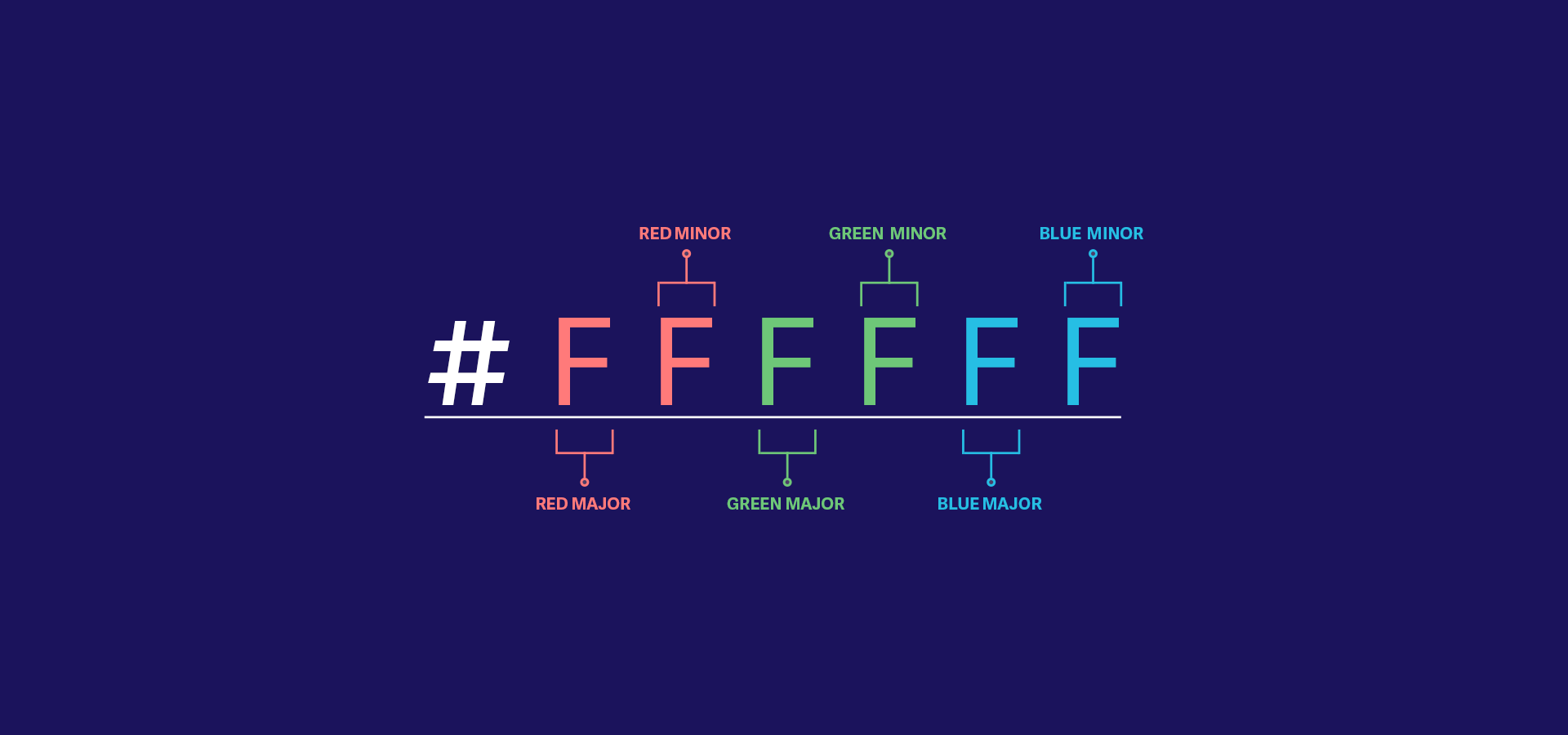
Hex: Hex (hexadecimal) — обозначение шестнадцатеричной системы счисления. Hex-редактор — приложение для редактирования данных, представляющее данные как последовательность байтов в hex-коде. Хекс (значения) Гекс — математическая игра на ромбической доске, имеющей гексагональную сетку.

Как видно из диаграмм, интенсивность яркости и оттенка увеличивается от 0 до F и уменьшается от F до 0. Цветовой код #000000 показывает истощение оттенка и преобладании темноты, а цветовой код #FFFFFF показывает увеличение оттенка и яркости. В шестнадцатеричном цветовом коде первые две цифры обозначают красный канал, третий и четвертый — зеленый, а последние две цифры обозначают синий канал.

## Шестнадцатеричные цветовые коды

Настольные и портативные цифровые устройства поддерживают цветовую модель RGB. RGB означает красный, зеленый и синий, и только эти три цвета используются для воспроизведения всех других цветов для цифровых дисплеев. Цветовой код шестнадцатеричного цвета — 6-значный цветовой код с префиксом хеша (#), который содержит цифровую информацию о цвете.

Шестнадцатеричные цветовые коды — прописываются через шесть цифр: Красный основной, Красный Малый, Зеленый основной, Зеленый второстепенный, Синий основной и Синий второстепенный. Основная цифра определяет, будет ли получающийся цвет оттенком красного, зеленого или синего; а младшая цифра определяет оттенок или тон этого цвета. Цифры с более высоким значением более яркие и насыщенные, а цифры с более низким значением темные и ненасыщенные.

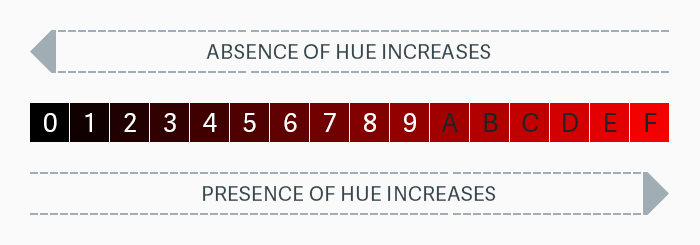


Красный - #FF0000; Зеленый - #00FF00; Синий - #0000FF. Давайте рассмотрим еще несколько шестнадцатеричных цветовых кодов, которые интерпретируются, когда два канала в модели RGB находятся на самом ярком уровне.  
Желтый — # FFFF00  
Фиолетовый — # FF00FF  
Бирюзовый — # 00FFFF

Цветовой код Hex содержит числовые значения от 0 до 9 и алфавитные значения от A до F, где 0 более темный цвет, а F более светлый.

Помимо этого F представляет проявление оттенка, а 0 — отсутствие оттенка. В приведенной ниже таблице показан оттенок красного канала.

Как видно из диаграмм, интенсивность яркости и оттенка увеличивается от 0 до F и уменьшается от F до 0. Цветовой код #000000 показывает истощение оттенка и преобладании темноты, а цветовой код #FFFFFF показывает увеличение оттенка и яркости. В шестнадцатеричном цветовом коде первые две цифры обозначают красный канал, третий и четвертый — зеленый, а последние две цифры обозначают синий канал.



**Относительные единицы измерения** это единицы, которые устанавливают размер, зависящий от другой величины *(относительно чего-либо)*.

В таблице приведен перечень относительных единиц измерения, использующихся в CSS.

|  |  |
| --- | --- |
| Единица | Описание |
| em | Высота шрифта текущего элемента. 1em - это текущий размер шрифта, относительно которого можно брать любые пропорции: 0.5em или 2em. |
| ex | Высота символа x. Зависит от текущего шрифта. |
| px | Размер задается в пикселях. Пиксель мельчайшая часть изображения. Его размер зависит от разрешения матрицы монитора. |
| % | Проценты. Размер задается в %, как правило, относительно величины/размера родительского элемента |

**Абсолютные единицы измерения в CSS и HTML**используются реже, чем относительные. Абсолютные единицы независимы ни от чего.

|  |  |
| --- | --- |
| Единица | Описание |
| cm | Сантиметр, см. |
| in | Дюйм. 1 дюйм равен 2,54 см. |
| mm | Миллиметр, мм. |
| pc | Пика. 1 пика равна 12 пунктам |
| pt | Пункт. 1 пункт равен 1/72 дюйма |
|  |  |

Вы можете перенести с XHTML на HTML5, используя тот же алгоритм.

|  |  |
| --- | --- |
| **Старый Тег HTML4** | **Новый Тег HTML5** |
| <div id="header"> | <заголовок> |
| <div id="menu"> | <нав> |
| <div id="content"> | <секция> |
| <div class="article"> | <статья> |
| <div id="footer"> | <футер> |

**:nht-child**

**Нечетные и четные — это ключевые слова, которые можно использовать для сопоставления дочерних элементов с четным или нечетным индексом (индекс первого дочернего элемента равен 1).**

Здесь мы указываем два разных цвета фона для нечетных и четных элементов p:

p:nth-child(odd) {  
  background: red;  
}  
  
pp:nth-child(even) {  
  background: lightgreen;  
}

[Попробуй сам "](https://www.w3schools.com/cssref/tryit.asp?filename=trycss3_nth-child_odd_even)

### Пример

Используя формулу ( *an* + *b* ). Описание: *a* представляет размер цикла, *n* — счетчик (начиная с 0), а *b* — значение смещения.

Здесь мы указываем цвет фона для всех p элементов, индекс которых кратен 3 (выберем третий, шестой, девятый и т. д.):

p:nth-child(3n+0) {  
  background: red;  
}

[Попробуй сам "](https://www.w3schools.com/cssref/tryit.asp?filename=trycss3_nth-child_formula)

### Пример

Здесь мы указываем цвет фона для всех p элементов, индекс которых кратен 3. Затем мы вычитаем 1 (выберем второй, пятый, восьмой и т. д.):

p:nth-child(3n-1) {  
  background: red;  
}

Давайте вернемся к «3n + 3» из оригинального примера. Как это работает? Почему он выбирает каждый третий элемент? Хитрость заключается в понимании «n» и алгебраического выражения, которое представляет. Представьте себе «n» как начинающееся с нуля, а затем набор всех положительных целых чисел. Затем завершите выражение. Таким образом, 3n — это «3xn», а все выражение вместе — «(3xn)+3». Теперь, подставляя в нулевое и положительное целые числа, получаем:

(3 x 0) + 3 = 3 = 3-й элемент  
(3 x 1) + 3 = 6 = 6-й элемент  
(3 x 2) + 3 = 9 = 9-й элемент  
и т.д.

Как насчет :nth-child(2n+1)?

(2 x 0) + 1 = 1 = 1-й элемент  
(2 x 1) + 1 = 3 = 3-й элемент  
(2 x 2) + 1 = 5 = 5-й элемент  
и т.д.

Эй, подожди! Это то же самое, что и «странный», поэтому, вероятно, не нужно использовать его очень часто. Но подождите сейчас. Разве мы не показали наш первоначальный пример как чрезмерно сложный? Что, если вместо «3n+3» мы использовали «3n+0», или еще проще «3n».

(3 x 0) = 0 = нет соответствия  
(3 x 1) = 3 = 3-й элемент  
(3 x 2) = 6 = 6-й элемент  
(3 x 3) = 9 = 9-й элемент  
и т.д.

Так что, как видите, матчи точно такие же, не нужно «+3». Мы можем использовать отрицательные n значений, а также использовать вычитание в выражениях. Например, 4н-1:

(4 x 0) – 1 = -1 = нет соответствия  
(4 x 1) – 1 = 3 = 3-й элемент  
(4 x 2) – 1 = 7 = 7-й элемент  
и т.д.

Использование значений «-n» кажется немного странным, потому что, если конечный результат отрицательный, совпадения нет, поэтому вам нужно будет добавить к выражению, чтобы снова получить его положительным. Как выясняется, это довольно умная техника. Вы можете использовать его для выбора "первых n элементов" с "-n+3":

-0 + 3 = 3 = 3-й элемент  
-1 + 3 = 2 = 2-й элемент  
-2 + 3 = 1 = 1-й элемент  
-3 + 3 = 0 = нет соответствия  
и т.д.

**nth-child и nth-of-type являются псевдоклассами и могут применяться только к элементам. – в 2013г.**

**:nth-of-type() {**

**}**

/\* Выбирает каждый четвёртый элемент <p>

среди любой группы соседних элементов. \*/

p:nth-of-type(4n) {

color: lime;

}

## Значения[¶](https://hcdev.ru/css/nth-of-type/?query=%3Anth-child#_2)

odd

Все нечётные номера элементов.

even

Все чётные номера элементов.

<число>

Порядковый номер дочернего элемента относительно своего родителя. Нумерация начинается с 1, это будет первый элемент в списке.

<выражение>

Задаётся в виде an±b, где a и b — целые числа, а n — счётчик, который автоматически принимает значение 0, 1, 2...

Если a равно нулю, то оно не пишется и запись сокращается до b. Если b равно нулю, то оно также не указывается и выражение записывается в форме an. a и b могут быть отрицательными числами, в этом случае знак плюс меняется на минус, например: 5n-1.

За счёт использования отрицательных значений a и b некоторые результаты могут также получиться отрицательными или равными нулю. Однако на элементы оказывают влияние только положительные значения из-за того, что нумерация элементов начинается с 1.

В табл. 1 приведены некоторые возможные выражения и ключевые слова, а также указано, какие номера элементов будут задействованы.

Табл. 1. Результат для различных значений псевдокласса

| **Значение** | **Номера элементов** | **Описание** |
| --- | --- | --- |
| 1 | 1 | Первый элемент, является синонимом псевдокласса :first-of-type. |
| 5 | 5 | Пятый элемент. |
| 2n | 2, 4, 6, 8, 10,… | Все чётные элементы, аналог значения even. |
| 2n+1 | 1, 3, 5, 7, 9,… | Все нечётные элементы, аналог значения odd. |
| 3n | 3, 6, 9, 12, 15,… | Каждый третий элемент. |
| 3n+2 | 2, 5, 8, 11, 14,… | Каждый третий элемент, начиная со второго. |
| n+4 | 4, 5, 6, 7, 8,… | Все элементы, кроме первых трёх. |
| -n+3 | 3, 2, 1 | Первые три элемента. |
| 5n-2 | 3, 8, 13, 18, 23,… | — |
| even | 2, 4, 6, 8, 10,… | Все чётные элементы. |
| odd | 1, 3, 5, 7, 9,… | Все нечётные элементы. |

**Допустимо комбинировать два псевдокласса :nth-of-type для выбора диапазона элементов. Здесь будут выбраны все элементы со второго по пятый.**

**:nth-of-type(n + 2):nth-of-type(-n + 5) {**

**/\* … \*/**

**}**

Но между двумя селекторами конечно же существует разница.

Наш селектор **:nth-child** выбирает элемент если:

1. Это параграф.
2. Второй элемент-наследник в родительском элементе.

Наш селектор **:nth-of-type** выбирает элемент, если:

1. Это второй элемент-наследник типа параграф в родительском элементе.

Селектор *:nth-of-type* накладывает меньше условий.

Выравнивание содержимого управляет выравниванием элементов по **главной оси.**

Здесь вы погрузитесь немного глубже в различие между основной и поперечной осью. Во-первых, давайте вернемся к flex-direction: row.

#container { display: flex; flex-direction: row; justify-content: flex-start;}

В вашем распоряжении пять команд для использования *justify-content*:

1. Флекс-старт
2. Гибкий конец
3. Центр
4. Пространство между
5. Космос-вокруг

<link href="https://fonts.googleapis.com/icon?family=Material+Icons"

rel="stylesheet">

<span class="material-icons">

mail

</span>

<span class="material-icons">

phone

</span>

Правило 2

Разместите [шрифт значка](https://github.com/google/material-design-icons/tree/master/font) в месте, например

https://example.com/material-icons.woff, и добавьте следующее правило CSS:

@font-face {

font-family: 'Material Icons';

font-style: normal;

font-weight: 400;

src: url(https://example.com/MaterialIcons-Regular.eot); /\* For IE6-8 \*/

src: local('Material Icons'),

local('MaterialIcons-Regular'),

url(https://example.com/MaterialIcons-Regular.woff2) format('woff2'),

url(https://example.com/MaterialIcons-Regular.woff) format('woff'),

url(https://example.com/MaterialIcons-Regular.ttf) format('truetype');

}

УСТАНОВКА GITHUB НА КОМП ЧЕРЕЗ ТЕРМИНАЛ

После того, как все действия по установке завершены, давайте проверим, что*Git* появился в системе компьютера. Откройте терминал и введите git --version. На экране терминала должна появиться текущая версия программы на вашей машине. Эта проверка подходит для всех операционных систем.

blahblahs-MacBook-Air:~ blahblah$ git --version

git version 2.26.2

**Git** — довольно гибкая система, и в ней есть множество настроек. Но мы начнём с необходимых. Для начала установим имя пользователя и адрес электронной почты. Эти данные будут отображены в каждом изменении, которое вы вносите, и другие пользователи смогут понять, кто автор тех или иных изменений.

В терминале запустите по очереди команды:

git config --global user.name "Name"

git config --global user.email email@example.ru

\*Обратите внимание, что в этих командах есть опция --global. Это значит, что вам достаточно будет один раз ввести эти данные и они будут сохранены для всех проектов. Если вы захотите поменять информацию, то нужно будет ввести ту же команду, только без  --global.

Теперь у вас есть локальный репозиторий и репозиторий на сервере (на github.com). Пришло время связать локальный репозиторий на вашем компьютере с удалённым репозиторием, который вы только что создали на GitHub. Чтобы это осуществить, снова перейдём в терминал.

// в папке Skillfactory

git remote add origin https://github.com/<your\_name>/<remote\_repository\_name>.git

Обратите внимание, что вместо <your\_name> и <remote\_repository\_name> следует указать ваш аккаунт на Github и название созданного на Github репозитория.

Остался последний шаг: пришло время добавить локальные изменения (коммит) на удалённый сервер. Для этого в терминале выполните команду:

// в папке Skillfactory

git push origin master

Терминал запросит ваш Username и пароль.

Теперь вы можете видеть изменения в репозитории по адресу:

https://github.com/<your\_name>/<remote\_repository\_name>

**git init  
Эта команда создаст .git репозиторий в Вашем проекте. Репозиторий или “repo” это коллекция всех изменений, которые были совершены на протяжении всего времени после инициализации репозитория. Это первое что нужно сделать для нового проекта.**

**Команда « touch » является стандартной командой, которая используется в операционной системе Linux для создания и изменения временных меток файла. echo > — непосредственная команда, создающая файл.**

## Создать папку — mkdir или md

Создаём пустую папку code внутри папки html :

Создаём несколько пустых вложенных папок — для этого записываем их через косую черту.

## Удалить папку — rmdir или rd

При этом удалить можно только пустую папку. Если попытаться удалить папку, в которой что-то есть, увидим ошибку.

## Показать дерево папок — tree

Если вы хотите посмотреть содержимое всего диска, введите tree в корне нужного диска. Получится красивая анимация, а если файлов много, то ещё и немного медитативная.

## Удаление файла — del или erase

Команда для удаления одного или нескольких файлов.

## Переименование файла — ren или rename

Последовательно вводим ren, старое и новое имя файла.

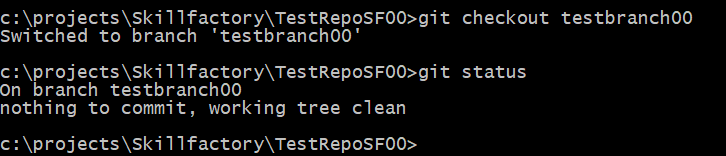
**Чтобы приступить к работе над изменениями для Чёрной пятницы, создадим специальную ветку:**

git branch black-Friday

**Создадим новую ветку в Git с помощью известной нам команды git branch, после которой следует имя создаваемой ветки. Назовем её следующим образом:**

**git branch testbranch00**

1. Для проверки состояния выполните команды git status и git log. Последняя показывает историю изменений: перечисляет коммиты, сделанные в репозитории, в обратном хронологическом порядке. Последние коммиты находятся вверху.
2. Команду git branch --list. Она выведет список веток, которые существуют в вашем локальном репозитории. Текущая ветка *master* будет выделена зелёным цветом.
3. Команду git branch --list -v . Вы увидите информацию о последних коммитах каждой из веток. Так как коммитов в ветку *testbranch00* ещё не было, то она указывает на коммит в *master*, от которого была создана.
4. Команда git checkout переключит нас на созданную ветку. Выполняя команду git checkout, укажите после неё имя ветки, на которую вы хотите переключиться. Например, git checkout testbranch00. Вы получите сообщение о переключении на ветку *testbranch00*.
5. Выполните команду git status. Вы увидите сообщение о том, что вы находитесь на ветке *testbranch00*.



**Чтобы удалить ветку, достаточно выполнить команду:**

git branch -D black-Friday

1. **Если введем git log --help - он предоставит нам документацию по какой-то определенной команде (в данном случае это - log)**

## Команды одной строкой

Очистить консоль — **cls**

Информация о системе — **systeminfo**

Информация о сетевых настройках — **ipconfig**

Список запущенных процессов — **tasklist**

Как создать новый текстовый файл через cmd? Для этого вводится команда NUL> с именем файла. Пример на изображении:

