[Что же собой представляет HTML?](#h.geweo2dfwlx0)

[Теги](#h.xilcd9bgwc69)

[Атрибуты](#h.vkrhkvlq2o83)

[Class id](#h.17ovlwb02q3c)

[Вложенность](#h.a93x2k70cc60)

[Блочные строчные](#h.nz0asgspmk85)

[CSS](#h.prss9ltlggrq)

[Применение css](#h.mgm1b2m73q7f)

[Виды селекторов](#h.963xnjun8rh5)

[тег](#h.93gibx5paicp)

[Class](#h.flzn4ey28vl9)

[id](#h.4yiwnze35lo2)

[Атрибута](#h.xu6ja48bobsc)

[Комбинированый](#h.geasxdn2d4ep)

[Вложенность](#h.jc6cpol6c0yd)

[Последовательность](#h.c1r42zeoi93n)

[Псевдоклассы](#h.pz0lhjtfj7bf)

[Псевдоэлементы](#h.lbm3bakbqnq6)

[Правила записи](#h.cejiu1szixwo)

[Приоритеты в css](#h.52y532o9vhh4)

[Каскад](#h.oriwtip6ey9m)

[Применение](#h.n1chk5xjvc4l)

[единицы измерения](#h.bvdzldeg3bsu)

[em](#h.mdah3j7rholq)

[Виды тегов](#h.ceo3r5w5s2si)

[Текстовые](#h.liwirctdcqkt)

[p](#h.m9zd76uwsfwo)

[H1-h6](#h.nxnunemwhflt)

[b, i](#h.g2vna491gh4k)

[a](#h.7cglcuhajdc3)

[Группирующие теги](#h.btjwqswhc2qe)

[<ol> <ul>](#h.pq8bqlitu34m)

[Список определений](#h.7p5eil6kt4ug)

[blockquote](#h.z9vvtp528lrm)

[Табличные теги](#h.tvu58qvxg8lh)

[Элементы формы](#h.ctcy51gianrz)

[Теги для вставки изображений](#h.t7ikfc4k2inc)

[Видео](#h.lc83wa7ea94p)

[Аудио](#h.22e15jrw7sr)

[Логические теги](#h.av3x4b2dl05y)

[main](#h.3vahpjirhfaf)

[section](#h.ft8msprkdt36)

[article](#h.xy4dgv89i6om)

[nav](#h.54lr2gnz62l)

[aside](#h.38z21jw419un)

[header](#h.ecg2esnz9ne3)

[footer](#h.yteznbjqwnfn)

[figure](#h.7d2n8uw5f58o)

[time](#h.6c9i677sspsj)

[IE 7, 8](#h.doc5mvt4wn8h)

[Outline](#h.6e9sy52bowt4)

[SEO](#h.iarkznuc39yt)

[xHTML](#h.c8p2zplzqqrz)

[Свойства css](#h.zhkyul35quul)

[Font Family](#h.8xyuycl6o90)

[Font Style](#h.mweddi5e59nl)

[Font Variant](#h.qik1bymly6ap)

[Font](#h.c02gncpf1gic)

[Свойства текста](#h.3fm43i2xsudl)

[Color](#h.4huhi6f3sgve)

[Letter Spacing](#h.jx4rmhhz2r4u)

[Word Spacing](#h.8uuxzjy5pzod)

[Text Transform](#h.pud10mo5qz8b)

[Text Decoration](#h.otxyw62z2ko7)

[White Space](#h.yl8ux1flwsnr)

[Text Align](#h.3075orlf0xep)

[Direction](#h.6mmi8afh83a0)

[Text indent](#h.hhjqodioy4ar)

[Vertical Align](#h.697p3vjxdasz)

[Свойства фона](#h.4kdl0pm4cdop)

[Background Color](#h.e14emaxesps)

[Background Image](#h.g4p3y8mvg2x7)

[Background Repeat](#h.35wwk741kih7)

[Background Attachment](#h.u25bsxhjz79i)

[Background Position](#h.qrnsul67oibb)

[Background](#h.r5jsohrzpmm)

[Свойства отображения](#h.kpa2n7mpc04s)

[Display](#h.m9cwl2vl58ns)

[Visibility](#h.r2zb6594l5n1)

[Свойства блока](#h.46zpx6hbykho)

[Width](#h.mbqzej4i9pro)

[Margin](#h.6cl6kq4e1wva)

[Padding](#h.oqcsimtcp1hz)

[Border](#h.5jrqsfbjyele)

[Реализация отступов](#h.mwpe5hekyte)

[CSS Float](#h.xnaqfevake1e)

[CSS-позиционирование](#h.poayr7eiyk73)

[CSS3](#h.wdycv3c3445v)

Давайте разберемся из чего состоит современная web-страница:

1. HTML-файл, в котором заложена структура страницы и ее контент.
2. CSS-файл, в котором содержатся правила для отображения каждого элемента на странице.
3. Набор картинок. Это и иллюстрации к контенту и фрагменты оформления страницы.
4. JavaScript. Он помогает с вычислениями, обеспечивает взаимодействие пользователя со страницей, создает анимированные эффекты и т.д.

В этом курсе мы подробно остановимся на первых двух пунктах, начнем с HTML.

# Что же собой представляет HTML?

С одной стороны, это язык разметки гипертекстовых документов, а с другой, - это файл с расширением .html, в котором записан код страницы.

Далее мы рассмотрим обе эти ипостаси.

За время существования языка было создано несколько стандартов и версий. И хотя на момент создания курса стандартом является версия языка xHTML1.0 (вот уже 15лет), но более перспективным и востребованным считается молодой стандарт HTML5, выход которого назначен на 2014год. Именно на основе HTML5 мы будем проводить наше обучение.

Как и любой другой документ, HTML-документ создается и оформляется в соответствии с определенными правилами и логикой.

HTML-документ состоит из четырех обязательных частей, расположенных в строгой последовательности:

1. <!DOCTYPE ...>
2. Основной контейнер <html> в котором содержится еще 2 части:
   1. Служебная секция <head>
   2. Тело документа <body>

Далее рассмотрим каждый элемент в отдельности.

Каждый HTML-документ должен начинаться с объявления своего типа.

Тип документа, его разновидность и версия объявляются с помошью инструкции <!DOCTYPE>.

Например, если мы создаем HTML5-документ, доктайп будет выглядеть так:

|  |  |
| --- | --- |
| 1 | <!DOCTYPE html> |

А если xHTML документ, то доктайп может выглядеть так:

|  |  |
| --- | --- |
| 1 | <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "<http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>"> |

Не волнуйтесь, вам не нужно запоминать содержимое этих строк, для каждого типа документа существует готовый шаблон. Один такой мы создадим с вами в процессе обучения.

Doctype необходим для браузеров, чтобы они понимали по каким правилам отрисовывать документ и для программ-валидаторов, чтобы знать с каким стандартом сверять этот документ.

<html> - это общий контейнер для всех остальных HTML-элементов, кроме <!DOCTYPE >.

Он содержит служебную секцию <head> основной раздел документа - <body>.

<html>

<head>

Meta info

</head>

<body>

Page content goes here.

</body>

</html>

Секция <head> содержит описание документа и служебную информацию. Ее содержимое не выводится в окне браузера, но необходимо для правильного отображения самого документа.

<html>

<head>

Meta info

</head>

...

</html>

Секция <head> располагается в начале контейнера <html> и, как правило, содержит:

1. служебную информацию, оформленная тегами <meta>, например указание кодировки документа:

<meta charset="utf-8">

Имя страницы оформленное тегом <title>, его содержимое отображается в шапке окна браузера:

<title>My webpage</title>

подключения внешних файлов CSS, оформленные тегами <link>:

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

подключения файлов JavaScript, оформленные тегами <script>:

<script src="js/main.js"></script>

# Теги

HTML (HyperText Markup Language) - это язык разметки Web-страниц.

Для понимания того как происходит разметка HTML-документа проведем аналогию с форматированием текста в любом текстовом редакторе. В обоих случаях, мы берем неотформатированный текст и указываем какая его часть будет заголовком, какая параграфом а какая списком или даже ссылкой. Только в HTML форматирование происходит не в визуальном редакторе, а в коде с помощью HTML-тегов. Каждый тег обозначает одну сущность – будь то заголовок, ссылка или параграф текста. Теги в HTML выглядят следующим образом:

<p>This is a paragraph</p>

Каждый тег заключен в символы «<» и «>». Первый тег (<tagname>) называется открывающим и обозначает начало начало чего-либо, в нашем случае параграфа. Второй тег (</tagname>) называется закрывающим и обозначает конец чего-либо. То что находится между открывающим и закрывающим тегом называется "телом тега". Тело тега выводится на экран браузера, его вид и поведение зависит от того в какой тег оно завернуто.

Следует отметить, что существуют теги, в которых не предусмотрено тело и соответственно нет закрывающего тега:

<br>, <img>, <input>, <link>

## Атрибуты

Вторым важным элементом языка разметки HTML являются - атрибуты, которые предоставляют дополнительную информацию о тегах.

Синтаксис атрибутов выглядит так:

<tag\_name attribute\_name="attribute\_value">

...

</tag\_name>

Атрибуты всегда записываются внутри открывающего тега.

Если бы существовал специальный тег для разметки информации об автомобилях мы возможно могли бы использовать следующие атрибуты:

<car manufacturer="toyota" model="corolla" body\_style="sedan" color="white" fuel\_type="diesel">

Как видете, атрибуты помогают нам описывать HTML-элементы.

Существуют специальные атрибуты, характерные только для определенных тегов, и универсальные, которые можно назначить любому тегу.

Примеры специальных атрибутов приведены ниже:

<a href="http://google.com">Link to Google</a>

<img src="/images/image1.jpg" alt="image description">

## Class id

Наиболее распространенными универсальными атрибутами являются **id** и **class**. С их помощью тегам присваивают имена собственные, что в свою очередь, позволяет задавать уникальные стили конкретным элементам, а не всем сразу. Без этого было бы не возможно реализовать все многообразие форм и размеров, присущее современным сайтам, ведь количество тегов ограничено.

При этом **id** присваивает уникальное имя, элемент с таким именем может встречаться только один раз на странице, например:

<div id="header">

...

</div>

**class** присваивает групповое имя, элементов с таким именем может быть несколько на странице. Как правило элементы с одинаковым именем имеют одинаковое оформление.

Например, присвоение блоку имени "post" будет выглядеть следующим образом:

<div class="post">

...

</div>

Одному HTML-элементу можно присвоить несколько имен классов. В этом случае имена перечисляются через пробел:

<div class="photo alignright">

...

</div>

## Вложенность

С целью реализации более сложных структур теги можно вкладывать друг в друга. Например, чтобы выделить одно слово в заголовке другим цветом нам понадобиться следующая конструкция:

|  |  |
| --- | --- |
| 1 | <h1>Hello, <mark>Mr. Smith</mark></h1> |

Обратите внимание на порядок открытия и закрытия тегов. В HTML существует строгое правило: **"теги закрываются в порядке обратном их открытия"**, или другими словами "первым пришел, последним ушел".

В наше примере сначала открывается тег <h1> потом открывается, вложенный в него, тег <mark>, потом закрывается тег </mark> и только после этого закрывается тег </h1>.

Это правило применимо и к более сложным конструкциям, например пост в блоге может выглядеть так:

|  |  |
| --- | --- |
| 1  2  3  4  5 | <article class="post">  <h1>Hello world!</h1>  <p>Lorem ipsum sid dolor...</p>  <a href="#">Read more</a>  </article> |

## Блочные строчные

Все теги можно условно разделить на блочные и строчные. Главным отличием этих двух типов является принцип их отображения в браузере:

1. Блочные элементы, идущие в коде подряд, будут отображаться друг под другом, каждый с новой строки, а строчные - в одну линию, слева направо, и если им не хватит места в этой строке они будут переносится на следующую.
2. Блочные элементы занимают всю ширину родителя, а строчные – ровно столько места сколько и их содержимое

Кроме того, для строчных элементов не применяются свойсва характерные для блоков: размеры, вертикальные отступы, горизотнальное выравнимание текста и др.

В HTML существует еще одно правило относительно вложенности тегов: **нельзя помещать блочные теги в строчные.** Например:

<span><div>text</div></span>

Такая конструкция не валидна, так как <span> это строчный элемент, а <div> - блочный. Любые другие варианты вложений: блочный в блочный, строчный в блочный или строчный в строчный допускаются.

Забегая вперед, отметим, что CSS позволяет изменить модель поведения любого элемента со строчного на блочный и наоборот.

ul li{

display:inline;

}

И хотя способ отображения элементов браузером измениться, это не изменит их сущность и не повлияет на правило вложенности.

С помощью CSS-свойств можно также получить и гибридный тип элемента - **строчно-блочный**.

ul li{

display:inline-block;

}

Такой элемент будет вести себя как строчный, но к нему будут применимы все свойства блочных элементов (высота, ширина, отступы и т.д.).

# CSS

CSS (Cascading Style Sheets) - это каскадные таблицы стилей, с помощью которых осуществляют оформление готового HTML-кода.

Основная цель CSS - отделить дизайн документа от его содержимого. Таким образом, CSS отвечает за оформление и внешний вид документа, а HTML - за его содержание и логическую структуру.

Фактически для каждого элемента в HTML можно написать некий набор (таблицу) правил, которые будут отвечать за его отображение и стилизацию.

CSS никак не влияет на поисковую оптимизацию сайта (SEO).

История эволюции CSS тесно переплетена с развитием HTML. Стоит отметить, что компания W3C занимается стандартизацией как HTML, так и CSS. C 1996 года ею было выпущено несколько редакций CSS. Последним утвержденным стандартом является редакция CSS2.1.

С каждой редакцией количество возможностей и функций CSS постоянно увеличивалось. В настоящее время все большую популярность приобретает - еще не утвержденная редакция CSS3, которая обещает стать качественно новым прорывом в верстке.

Давайте рассмотрим синтаксис CSS на конкретном примере:

p { background:red;}

где p – это селектор, ключ, по которому осуществляется выборка HTML-элементов, к которым будут применены стили. В нашем случае это все параграфы.

background – свойство, которое мы хотим назначить для этого тега;

red – значение свойства background.

Универсальная формула записи стиля выглядит так:

селектор {свойство:значение;}

Если один селектор имеет больше одного свойства, записываем стили следующим образом:

селектор {

свойство1:значение;

свойство2:значение;

}

При этом все пары "свойство - значение" необходимо заключить в фигурные скобки "{ }" и после каждой из них обязательно поставить точку с запятой " ; ".

Также можно применить один набор свойств к нескольким селекторам, в этом случае селекторы перечисляются через запятую:

селектор1,

селектор2 {

свойство1:значение;

свойство2:значение;

}

## Применение css

Существует три основных способа применить стили к существующему HTML-документу:

1. Применение inline-стилей, то есть назначение стилей каждому конкретному HTML-элементу с помощью атрибута **style=" "**:

<p style="color:#f00; margin:0;">a unique paragraph in red</p>

1. В этом случае заданные свойства применятся только к этому конкретному элементу. Это совсем не рационально. На практике этот метод используется только, если нужно динамически задавать свойства посредством JavaScript.
2. Встраивание стилей непосредственно в секцию <head> HTML-документа с помощью тега **<style>**:

<style type="text/css">

p {

color: red;

}

</style>

1. В этом случае стили применятся ко все указанным элементам, но только на этой конкретной странице. Для других страниц придется писать свойства заново и многие из них дублировать, так как страницы одного сайта обычно имеет единое оформление, а часть элементов вообще остаются неизменными на всех страницах сайта.
2. Подключение внешних стилей в виде отдельного текстового .css-файла с помощью тега **<link>**:

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

В этом случае мы можем применить один и тот же набор стилей ко всем страницам сайта. Это самый оптимальный способ.

Поскольку основной целью CSS является отделение дизайна документа от его содержимого, далее мы будем использовать только вариант с внешним подключением CSS.

## Виды селекторов

Мы изучили базовые правила написания CSS, но все еще остается открытым вопрос: "Как задать свойства конкретному элементу HTML?".

Существует всего четыре основных способа обратиться к HTML-элементу из CSS: по названию тега, по имени класса, по идентификатору или по аттрибуту. Так же активно используются всевозможные комбинации этих способов.

Давайте рассмотрим каждый способ в отдельности.

### тег

Наиболее простой способ обращения к HTML-элементам из CSS - по названию тега. В этом случае, в качестве селектора записывается названию тега, которым мы хотим задать свойства:

h1 {

font-size: 24px;

color: #666;

border-bottom:2px solid #666;

margin: 0 0 10px;

}

Данный селектор следует читать, как: "Все заголовки первого уровня". Другими словами, селектор тега предполагает обращение ко всем тегам данного типа на странице.

### Class

Кроме названия тегов, в качестве обращения к HTML-элементам можно использовать имя класса. В этом случае, в качестве селектора записывается имя класса, присвоенное в HTML элементам, которым мы хотим задать свойства. В общем виде обращение по классу выглядит так:

.classname {

property: value;

}

Где "точка" перед словом "classname" говорит нам о том, что "classname" - это имя класса.

Поскольку класс это групповое имя, то свойства применяться ко всем HTML-элементов с указанным именем класса. И читается такой селектор так: "все элементы с классом post".

Если HTML-элементу присвоено несколько имен классов, то он будет наследовать свойства от каждого класса.

### id

Селектор id применяется, когда необходимо обратиться к уникальному HTML-элементу. В этом случае, в качестве селектора записывается имя идентификатора элемента, которому мы хотим задать свойства:

Где "решетка" перед словом "idname" говорит нам о том, что "idname" - это уникальное имя элемента.

В данном случае свойство применится только к одному HTML-элементу с идентификатором "header".

### Атрибута

Кроме названия тегов, классов и идентификаторов в качестве обращения к HTML-элементам можно использовать название атрибута и/или его значение. В общем виде обращение по атрибуту выглядит так:

tagname[attributeName="attributeValue"] {

property: value;

}

Где "[ ]" говорят нам о том что это обращение по атрибуту, а tagname - имя тега которому присвоен атрибут.

Рассмотрим на примере. Для того, чтобы обратиться к элементу

<input type="text" placeholder="User name" >

можно сделать следующую запись в CSS:

input[type="text"] {

color: #f00;

}

Такой селектор читается как: "для всех <input> у которых есть атрибут type со значением text".

Кроме того, можно осуществлять выборку лишь по наличию атрибута, не беря в учет его значение:

img[alt] {

border: 1px solid #f00;

}

В этом случае свойство применится ко всем <img> с атрибутом alt.

### Комбинированый

Кроме того, в CSS возможно применение комбинаций различных видов селекторов. Рассмотрим пару примеров:

a.button {

display: block;

}

В этом случае стили будут применены ко всем тегам <a> с классом "button".

div#header {

background: #aaa;

}

этом случае стили будут применены только к тегам **<div>** с идентификатором "header".

Обычно указание тега перед именем класса или идентификатора считается избыточным. Поэтому прибегать к такого рода селекторам нужно лишь в крайнем случае.

### Вложенность

Однако для того чтобы присвоить свойства конкретной группе элементов не обязательно всем им присваивать имена классов. Достаточно присвоить имя только родительскому элементу и воспользоваться контекстной формой записи селекторов и в этом нам помогут селекторы вложенности.

Простая вложенность в CSS-селекторе обозначается "пробелом":

#navigation a {

color:#666;

}

Такие селекторы "читаются" справа налево. В нашем случае свойства будут применены "для всех <a> внутри тега с идентификатором navigation".

Непосредственная вложенность в CSS-селекторе обозначается символом ">":

#navigation > li > a {

color:#666;

}

В этом случае случае свойсва будут применены "для всех <a> вложенных непосредственно в <li>, которые, в свою очередь, вложены непосредственно в тег с идентификатором navigation".

В чем разница? В первом случае свойства применятся ко всем ссылкам в навигации, включая выпадающее меню, а во-втором - только к ссылкам первого уровня.

### Последовательность

Иногда свойства элементов зависят не от вложенности, а от последовательности элементов. В таком случае имеет значение какие элементы расположены перед теми элементами, которым мы хотим задать свойства. Для такой конкретизации используйте селектор "+":

h1 + p {

font-weight: bold;

}

В данном случае свойтво применится ко всем <p>, которые в коде следуют сразу после <h1> (см. пример справа).

Псевдоклассы (англ. pseudo-class) - ключевые слова добавляемые к селектору для описания специальных состояний элемента.

### Псевдоклассы

Псевдоклассы позволяют применять стили к элементам не только исходя из их местоположения в дереве документа, но и в зависимости от внешних факторов, таких как история посещений (например, :visited), статус элемента (например, :checked или :focus), позиции курсора мыши (например, :hover), и др.

Вот перечень наиболее популярных псевдоклассов:

* :hover позволяет определить стиль для ссылок, в момент наведения курсора мыши (onmouseover).
* :link - это еще не посещенные ссылки.
* :visited - уже посещенные уже посещенные ссылки.
* :active позволяет определить стили для ссылок, в момент нажатия кнопки мыши (onmousedown).
* :focus позволяет определить стили для элементов при получении ими фокуса, например, для текстового поля формы получение фокуса означает, что курсор установлен в поле.
* :first-child позволяет определить стили для первого дочернего элемента селектора.
* :lang позволяет определять стили в зависимости от языка контента. Который в свою очередь устанавливается атрибутом lang="fr".
* :checked определяет отмеченное состояние чекбокса или радиобатона.
* :nth-child

Синтаксически псевдоклассы записываются через двоеточие после селектора того элемента, к которому они относятся. Вот пара примеров:

a:hover {

text-decoration: underline;

}

#nav li a:hover span {

color: #000;

}

### Псевдоэлементы

Псевдоэлементы :before и :after служат для вставки контента до или после контента того элемента к которому они применяются.

Псевдоэлементы записываются через двоеточие после того элемента к которому применяются и имеют одно обязательное свойство "content":

.box:before {

content:'';

}

Значение свойства content может:

1. оставаться пустым - равносильно добавлению элемента <span> в начало\в конец выбранного элемента;
2. содержать текст или спец-символ - равносильно добавлению элемента <span> с соответствующим текстом или спец-символом в начало\в конец выбранного элемента, т.о. этот html-код:

<div class="box">

<span>BEFORE</span>

Lorem ipsim duis aum.

</div>

равносилен такой записи в CSS:

.box:before {

content:'BEFORE';

}

1. содержать адрес изображения - указанное изображение будет вставлено в начало\в конец выбранного элемента.

Элементы созданные с помощью :before\:after наследует свойства элемента к которому они применены, но для них можно дополнительно задать любые CSS-свойства :

a.download-pdf:before {

content:'';

float:left;

width:6px;

height:6px;

background:url(../images/pdf.png) no-repeat;

}

Области применения:

* замена пустых элементов и дополнительной вложенности при реализации оформления блоков;
* реализации булетов;
* "хвосты" ссылок, дропдаунов, блоков с сообщениями;
* маски изображений;
* вертикальное выравнивание элементов внутри блока с фиксированной высотой;
* фоны слева и справа от заголовков;
* сложные и тени;
* и др.

Внимание! Псевдоэлементы не работают в браузерах IE7 и старше.

### Правила записи

Для правильного написания селекторов необходимо мысленно разбить страницу на самостоятельные логические блоки. Описание каждого элемента должно начинаться с имени логического блока (контекстная форма записи):

.box .holder{..}

.box h3{..}

.box ul a{..}

#footer a{..}

В тоже время свойства для элементов повторяющихся в разных логических блоках могут быть заданы глобально:

.button-more{..}

При написании селекторов всегда должен выдерживаться баланс между универсальностью и конкретностью:

* селекторы не должны быть слишком длинными;
* селекторы должны позволять использовать описанный элемент в других блоках, или переместить описанный блок в другую колонку.

Следует отметить, что названия **class** и **id** должны быть логичными, то есть они не могут быть абстрактными или написанными с ошибками.

## Приоритеты в css

Довольно часто складывается ситуация когда для одного и того же HTML-элемента подходит несколько правил CSS или свойств, заданных несколькими способами. И иногда эти свойства/правила противоречат друг другу. Какое свойство или правило должно работать в этом случае? Давайте разберемся.

Существуют четкие правила, согласно которым применяются стили, если для какого-то элемента подходят несколько правил одновременно и их свойства противоречат друг другу. Это правила называются "каскадом", в котором для каждого свойства/правила рассчитывается приоритет (вес), что делает результат предсказуемым.

Общее правило звучит так: "большим приоритетом обладает то свойство, которое написано в коде позже". Но это правло работет только если селекторы пересекающихся правил идентичны или имеют одинаковый приоритет:

#content p{

color:blue;

}

...

#content p{

color:red;

}

или свойства записаны внутри одного селектора:

#content p{

color:blue;

...

font-size:12px;

margin:0;

...

color:red;

}

В обоих случаях параграфы будут красными.

Во всех остальных случаях нужно рассчитывать приоритет ("вес") селекторов/свойств.

### Каскад

Ниже приведен список методов присвоения свойств и селекторов в порядке убывания приоритета:

1. Наибольшим приоритетом обладают свойства заданные с параметром **!important**. Например:

.warning {

color:#f00 !important;

}

Чуть меньшим приоритетом обладают стили заданные инлайново в HTML через атрибут style=" ". Например:

<strong style="color:#f00;">lorem ipsum dolor</strong>

Далее идут свойства заданные с помощью селектора идентификатора. Например:

#content{

color:#f00;

}

Далее идут свойства заданные с помощью селектора класса. Например:

.post {

color:#f00;

}

Наименьшим приоритетом обладают свойства заданные с помощью селектора тега. Например:

p{

color:#f00;

}

### Применение

Согласно общему правилу свойства, написанные в коде позже, должны иметь больший приоритет и все параграфы в постах должны быть красными. Но, как вы видите справа, это нет так. Наоборот, все параграфы синие, т.е. сработало верхнее правило. Почему?

Дело в том что приоритет селектора **#content p { }** выше чем приоритет селектора **.post p { }**. И снова, почему? Для упрощения сравнения селекторов мы можем мысленно сократить их на одинаковые типы селекторов: в обоих селекторах есть имя тега, сокращаем. В результате в первом селекторе остается один идентификатор **#content**, а во втором один класс **.post**. А как мы помним из предыдущего шага приоритет идентификатора выше. Именно поэтому срабатываю свойства записанные в первом правиле.

Хорошо, разобрались. Но нам то нужно чтобы параграфы в постах были красными. Как этого добиться? Для этого нужно увеличить "вес" второго селектора, путем добавления в его начало имени родительского контейнера (одного или нескольких, в зависимости от ситуации):

#content .post p {

color: red;

}

В результате будут работать оба правила но каждое для своей части параграфов.

Использовать параметр !important для увеличения веса конкретного свойства допускается лишь в крайних случаях, когда обычный способ не помогает.

### единицы измерения

Теперь рассмотрим, какие единицы измерения используются в CSS. Есть два типа единиц измерения: абсолютные и относительные.

К относительным относятся:

1. px
2. em
3. %

К абсолютным относятся:

1. in
2. cm
3. mm
4. pt

### **em**

**em** – измеряемое значение, которое зависит от размера шрифта текущего элемента, присвоенного через свойство **font-size**.

Если же **font-size** не задан, наследуется значение родительского элемента. Поэтому 1em изначально соответствует размеру шрифта, заложенному в браузере по умолчанию (16px).

По такому же принципу работает и единица «%». 100% = 1em. На практике можно использовать как «%», так и «em

# Виды тегов

## Текстовые

### p

Характеристики элемента:

* тип: блочный;
* свойства по умолчанию: margin-top, margin-bottom.

С помощью параграфов оформляются практически все текстовые фрагменты на странице

### H1-h6

Заголовки должны располагаться в коде в правильной последовательности от <h1> до <h6>(см. пример). И если на их сновании построить содержание страницы оно должно быть корректным:

Характеристики элемента:

* тип: блочный;
* свойства по умолчанию: margin-top, margin-bottom, font-size, font-weight.

### b, i

Теги **<b>** и **<i>** используются для изменения начертания части текстового фрагмента: **<b>** придает тексту жирность, а **<i>** делает его наклонным

Теги **<strong>** и **<em>** по умолчанию дают аналогичный визуальный эффект: **<strong>** придает тексту жирность, а **<em>** делает его наклонным. Но применяются они для обозначения важной информации:**<strong>** в большей степени, а **<em>** - в меньшей.

Тег **<span>** это универсальный строчный элемент, не имеющий смысловой нагрузки и свойств по умолчанию. С его помощью оформляют небольшие текстовые фрагменты для которых не подходит <p> или реализуют сложное форматирование текста внутри строчных элементов.

Все перечисленные элементы являются строчными. Текст заключенный в <strong> и <b> по умолчанию выводится жирным, а в <i> и <em> - наклонным. <span> и <mark> не имеют свойств по умолчанию.

### a

Тег <a> относится к строчным, но позволяет помещать внутрь себя и блочные элементы. Таким образом мы можем сделать кликабельным целый блок:

* тип: строчный;
* свойства по умолчанию: text-decoration:underline, color:#00f;

## Группирующие теги

### <ol> <ul>

<ol> (нумерованный список) и <ul>(ненумерованный список с буллетами). Элементы списков в обоих размечаются размечаются тегом <li>.

На практике не нумерованный список имеет очень широкое применение. <ul> используют для разметки:

* всевозможных навигаций и меню;
* списков продуктов, спонсоров, новостей и событий;
* ссылки на социальные сети;
* пейджингов;
* элементов управления слайдшоу и табов;
* и многого другого.

Характеристики элементов <ul>,<ol>:

* тип: блочный;
* свойства по умолчанию: margin-top, margin-bottom, padding-left, list-style.

Характеристики элементов <li>:

* тип: блочный;

### Список определений

<dl>

<dt>Tel:</dt>

<dd>8 800 123 345 6</dd>

</dl>

Характеристики элементов <dl>:

* тип: блочный;
* свойства по умолчанию: margin-top, margin-bottom.

Характеристики элемента <dt>:

* тип: блочный.

Характеристики элемента <dd>:

* тип: блочный;
* свойства по умолчанию: margin-left.

### blockquote

<blockquote>

<q>Text quote</q>

<cite>author</cite>

</blockquote>

Характеристики элемента <blockquote>:

* тип: блочный;
* свойства по умолчанию: margin.

Характеристики элемента <q>:

* тип: строчный;
* свойства по умолчанию: quotes.

Характеристики элемента <cite>:

* тип: строчный;
* свойства по умолчанию: font-style:italic.

## Табличные теги

В HTML для разметки табличных данных используются специальные теги:

* <table> - им обозначают начало и конец таблицы;
* <tr> - обозначает начало и конец строки таблицы и располагается внутри таблицы;
* <td> - обозначает начало и конец ячейки таблицы и может располагаться только внутри тега <tr>;
* <th> - применяется для обозначения заглавной ячейки строки или столбца, и тоже может располагаться только внутри тега <tr>.

обратите внимание, после объединения количество тегов(th/td) в разных строках может отличаться, но количество логических ячеек (с учетом индексов объединения) должно оставаться одинаковым.

По отображению в браузере таблицы больше похожи на блочные элементы: они всегда начинаются с новой строки и к ним применимы свойства, характерные для блочных элементов (размеры, отступы и т.д). Но в отличии от блочных элементов таблицы не растягиваются на всю ширину родительского элемента, а стараются сжаться по ширине до размеров содержимого.

Сломать таблицу практически невозможно, даже если поместить в ячейки таблицы контент заведомо больших размеров, таблица сама подстроится под содержимое, игнорируя заданные размеры.

Характеристики тега table:

* тип: табличный;
* свойства по умолчанию: border-collapse:separate, margin;

Характеристики тега td:

* тип: табличный;
* свойства по умолчанию: padding, vertical-align:middle;

Характеристики тега th:

* тип: табличный;
* свойства по умолчанию: padding, font-weight:bold, text-align:center;

## Элементы формы

Тег **<form>** указывает на начало и конец формы. У этого тега есть обязательный атрибут **action="#"**,

Эти теги связаны с помощью атрибутов <label **for="search-field"**> и <input **id="search-field"** type="search"> с одинаковым значением. Эта связка добавляет удобства в работе с формами, позволяя активировать элементы формы по клику на их подписи.

**<input type="submit">** - кнопка для отправки заполненных данных на сервер. В атрибуте «value» этого тега указывается текст, который будет отображаться на кнопке.

Как вы уже заметили, тег **<input>** бывает разных типов. Вот те, которые вам будут встречаться на практике:

* **<input type="text">** - текстовое поле ввода;
* **<input type="password">** - поле ввода пароля, маскирует вводимые текст;
* **<input type="search">** - разновидность текстового поля ввода, предназначен для ввода поисковых запросов;
* **<input type="tel">** - разновидность текстового поля, предназначен для ввода номеров телефонов;
* **<input type="email">** - разновидность текстового поля, предназначен для ввода e-mail адресов;
* **<input type="checkbox">** - флажок, позволяет выбрать несколько вариантов из предложенных;
* **<input type="radio">** - переключатель, позволяет выбрать только один вариант из предложенных. Имеет обязательный атрибут name, уникальный для каждой группы переключателей;
* **<input type="file">** - поле + кнопка для выбора и отправки файлов;
* **<input type="reset">** - кнопка сброса данных формы;
* **<input type="submit">** - кнопка отправки формы;
* **<input type="image">** - кнопка отправки формы, выполненная в виде картинки.

<textarea rows="5" cols="80"></textarea>

Размер поля задается двумя обязательными атрибута – **cols** и **rows.** Они задают количество символов, которые будут помещаться в **<textarea>** по горизонтали и вертикали соответственно.

Тег **<select>** создает выпадающий список с ограниченным числом вариантов из которых пользователь может сделать выбор. Выбрать можно как один вариант так и несколько (если для тега <select> задать атрибут multiple="multiple").

<button type="submit"><span>Submit<span></button>

Все элементы форм относятся к строчному типу и имеют множество свойств по умолчанию.

## Теги для вставки изображений

<img src="images/img.jpg" alt="" width="100" height="50">

src и alt - обязательные атрибуты. Атрибуты width и height задают размер изображения в px. Хоть они и не являются обязательными, указывать размеры - это правило хорошего тона. Они резервируют необходимое место, пока не загрузится изображение.

Тег <img> относится к строчным элементам.

### Видео

Начиная с HTML5 появилась возможность встраивать видео в страницу использую только теги HTML. Для этого нужно приложить видеофайл к проекту и указать к нему путь внутри тега <video>:  
<video controls="controls" width="640" height="360">

<source src="inc/echo-hereweare.mp4" type="video/mp4" />

<source src="inc/echo-hereweare.webm" type="video/webm" />

<source src="inc/echo-hereweare.ogv" type="video/ogg" />

</video>

Где <source> это тег который указывает путь к видеофайлу и формат видеозаписи. Разные браузеры поддерживают разные видеоформаты поэтому для обеспечения максимальной кроссбраузерности каждый видеоролик нужно предоставить в трех видеоформатах: mp4, webm, ogv.

Для браузеров которые не поддерживают теги HTML5 видео по прежнему подключается через flash player. Но делается это автоматически с помощью плагина mediaelements.js.

На тот случай если в браузере будет отключен JavaScript fallback можно прописать и вручную:

<video controls="controls" width="640" height="360">

<source src="inc/echo-hereweare.mp4" type="video/mp4" />

<source src="inc/echo-hereweare.webm" type="video/webm" />

<source src="inc/echo-hereweare.ogv" type="video/ogg" />

<!-- Flash fallback for non-HTML5 browsers without JavaScript -->

<object width="640" height="360" type="application/x-shockwave-flash" data="swf/flashmediaelement.swf">

<param name="movie" value="swf/flashmediaelement.swf" />

<param name="flashvars" value="controls=true&file=../inc/echo-hereweare.mp4" />

<img src="inc/echo-hereweare.jpg" width="640" height="360" title="No video playback capabilities" />

</object>

</video>

### Аудио

В HTML5 также можно встраивать аудио в страницу использую только теги HTML. Для этого нужно приложить аудиофайл к проекту и указать к нему путь внутри тега <audio>:

<audio controls="controls" height="30" width="400">

<source type="audio/ogg" src="inc/track01.ogg"/>

<source type="audio/mp3" src="inc/track01.mp3"/>

<p>No audio playback capabilities. <a href="inc/track01.mp3">Download the song</a></p>

</audio>

Где <source> это тег который указывает путь к аудиофайлу и формат аудиозаписи. Разные браузеры поддерживают разные форматы поэтому для обеспечения максимальной кроссбраузерности аудиозапись нужно предоставить в двух форматах: mp3 и ogg.

Для браузеров которые не поддерживают теги HTML5 видео обычно предусматривается ссылка для скачивания аудиофайла. Но можно воспользоваться и плагином mediaelements.js он обеспечит нормальную работу плеера в старых браузерах.

## Логические теги

## main

Тегом <main> обозначают основной контент на странице. Он должен включать только уникальный контент, характерный только для этой страницы, и не должен содержать контент повторяющийся на всех странцах сайта: header, footer, навигацию и т.д.

Для тега <main> рекомендуется использовать атрибут role="main".

Тегом <main> нельзя использовать внутри элементов <article>, <aside>, <footer>, <header> или <nav>.

## section

Тег <section> применяется для разбивки контента страницы или статьи на логические части. Каждая секции документа, как правило, имеет свой заголовок. Примером могут быть блоки "Introduction", "About Us", "Contact information" и т.д.

Также тег <section> применяется для для группировки однотипных элементов под общим заголовком. Примером могут быть блоки "Resent posts", "News", "Our last projects"

Блоки в сайдбарах также обычно размечают как секции или их разновидности.

## article

<article> это разновидность секции. Она применяется для разметки целостных самодостаточных независимых частей страницы, которые не потеряют смысл за пределами этого документа и могут быть повторно использованны на другом ресурсе (repost). Яркими примерами таких секция являются новостные статьи или посты в блоге

## nav

<nav> это разновидность секции, содержащая навигационные ссылки на другие страницы или на части этойже страницы. Как и любая секция <nav> может иметь свой заголовок.

На практике тег <nav> применяется для реализации всевозможных навигаций и организации групп ссылок

Тег <nav> не используют:

* для замены тега <ul>, если нужно список вкладывают внутрь;
* для реализации переключателя слайдов или табов;
* для оформления ссылок на социальные сети;
* для ссылок на другие языковые версии сайта.

## aside

<aside> это разновидность секции. Она применяется для разметки информация дополняющей основной контент вокруг <aside>. Другими словами, одно и тоже содержимое тега <aside> будет восприниматься по разному в зависимости от того в каком логическом родительском элементе он находиться. Это может быть дополнительная информация к статье если родителем будет <article>, или дополнительная информация ко всей странице, если логическим родителем будет <body>.

На практике, чаще всего, это будут сайдбары или верхняя часть "жирного" футера (но не внутри тега <footer>):

## header

Тег <header> применяется для разметки вводного контента для своего ближайшего родителя. <header> обычно содержит заголовок, но это не обязательно. Кроме заголовка и вводной информации может содержать логотипы, навигационные средства, форму поиска, содержание документа\статьи и др. Тег <header> не является обязательным, и необходим скорее для отделения служебной и вступительной информации от собственно контента. Поэтому если из всей вступительной информации у вас есть только заголовок, то и тег <header> вам вероятно не нужен.

Тег <header> может применяется для реализации хедера всего документа, или хедера статьи или секции.

## footer

Тег <footer> применяется в качестве колонтитула для своего ближайшего родителя. <footer> обычно содержит служебную (meta) информацию, информацию об авторе, ссылки на соотносящиеся документы, копирайт. Тег <footer> также не является обязательным, и необходим скорее для отделения служебной информации от собственно контента документа или статьи. Поэтому если из всей служебной информации у вас есть только ссылка "read more", то и тег <footer> вам не нужен.

Тег <footer> может применяется для реализации футера всего документа, или футера статьи или секции. При этом футер не обязательно должен находиться в конце статьи или секции, он может находиться и прямо под заголовком. <footer> и <header> практически взаимозаменяемы, только <footer> не может содержать заголовок статьи или секции.

## figure

Тег <figure> применяется для реализации иллюстраций к контенту с подписями.

<figure>

<img src="images/laptop.png" alt="image description">

<figcaption>The <a href="#">MacBook Air</a></figcaption>

</figure>

## time

<time> выделяет дату или время в тексте.

<p>Our first date was <time datetime="2006-09-23">a Saturday</time>.<p>

Внутри тега <time> время и дату можно записывать в любом формате, но обязательно нужно дублировать их в атрибуте datetime в [понятном для компьютеров формате](http://www.w3.org/TR/html5/text-level-semantics.html#datetime-value).

## IE 7, 8

Старые браузеры IE 7, 8 не поддерживают логические теги. Для работы с ними нужно подключать JavaScript [HTML5 Shiv](https://github.com/aFarkas/html5shiv), который создаст набор логических тегов в этих браузерах:

<!--[if lt IE 9]>

<script src="js/html5shiv.js"></script>

<![endif]-->

Кроме того, для всех логических тегов нужно принудительно задать блочный тип:

header, footer, article, section, nav, figure, aside, main { display: block; }

Основной вашей задачей при использовании логических тегов будет разметить весь дополнительный (не основной) контент и служебную информацию. И если, после того как вы "отрезали" <header>, <footer> и все <aside>, относящиеся к документу в целом, у вас остался однородный цельный контент, можете считать что задача уже выполнена и дальнейшая разметка логическими тегами ненужна. Нет, вы можете конечно завернуть оставшийся контент в <article>, но это необязательно.

Но, если оставшийся контент логически неоднороден: состоит из нескольких статей или разделов и подразделов, тогда продолжайте логическую разметку тегами <section> и <article>. Опять же, лучше начать с оформления подразделов, т.к. информация находящаяся за пределами логических тегов по умолчанию считается основным контентом документа.

В любом случае, лучше ориентироваться на outline. О нем речь пойдет в следующем шаге.

## Outline

Outline это автоматически создаваемое содержание web-страницы. Он описывает структуру документа и может использоваться поисковыми системами для улучшения результатов поиска.

В предыдущих версиях HTML в создании outline принимали участие только заголовки, поэтому так важен был их порядок.

В HTML5, кроме заголовков, в этом процессе участвуют <section>, <article>, <aside> и <nav>. Каждый из этих тегов создает подраздел в структуре документа.

Какие практические знания дает это пример?

1. Каждая секция/статья HTML5-документа может начинаться с заголовка <h1> и даже в этом случае можно выстроить иерархию разделов и подразделов страницы.
2. Внутри каждого раздела/подраздела порядок заголовков должен сохраняться правильным от <h1> к <h6>.
3. Для создания раздела или подраздела документа/статьи лостаточно завернуть блок контента в <section> или его разновидность.

Верно построенный outline это индикатор правильной логической разметки страницы, поэтому корректный outline является одним из основных требований семантической верстки.

## SEO

Наша задача будет сводиться лишь к тому, чтобы придерживаться базовых принципов семантической верстки при порезке страницы:

1. **Применять HTML-теги только по прямому назначению.** "Зачем?" - спросите вы - "Ведь есть CSS." Да, CSS это мощный инструмент, который позволяет нам любому тегу придать необходимый вид: сделать из параграфа заголовок, а списку придать вид таблицы и т.д. Но, дело в том, что поисковые системы игнорирую CSS и анализируют только HTML-код. Поэтому очень важно использовать для разметки правильные теги. Это сделает документ более логичным, читабельным и понятным для поисковых систем.
2. **Проверять outline.** Расставляя заголовки или разбивая документ на разделы логичискими тегами, следите за тем, чтобы сохранялась логичная структура документа.
3. **Размещать основное содержимое как можно ближе к началу HTML-документа.** На практике это означет лишь что колонка "content" должна иди в доде до "сайдбаров".

## xHTML

Кроме того, что в xHTML теги <link> и <meta> являются самозакрывающимися есть еще ряд отличий в оформлении секции <head>

В xHTML нет логических тегов:

В xHTML есть ряд нюансов в реализации форм:

Во-первых, элементы форм не могут находиься непосредственно в теге <form>. Их нужно доболнительно оборачивать в <fieldset>, <div> или <p>.

Во-вторых, не существует инпутов с типами email, tel и search, вместо них используется инпут с типом text

И в-третьих, для отображения текста подсказки используется атрибут value="" вместо не существующего placeholder="".

Очистка значения при фокусе на инпут осуществляется с помощью JavaScript.

Как вы уже знаете в XHTML нет логических тегов, поэтому outline строится исключительно по заголовкам. При этом на странице может быть только один <h1> и он должен озаглавливать всю страницу или статью в колонке "content".

Правильность построения outline в xHTML документах проверяют с помощью плагина Web Developer Toolbar для браузера Firefox, используя команду: Information --> View Document Outline.

# Свойства css

## Font Family

Свойство font-family назначает шрифт или семейство шрифта

Все шрифты делятся на три основных семейства:.

* serif - шрифты с засечками.
* sans-serif - шрифты без засечек.
* monospace - шрифты в котором все буквы имеют одинаковую ширину.

### Font Style

Свойство font-style определяет начертание шрифта: normal (обычное) или italic (*наклонное*).

### Font Variant

Свойство font-variant определяет отображение строчных букв: normal (Обычное) или small-caps (Сделать прописными уменьшенного размера).

### Font

Свойство font это сокращенная форма записи всех свойств шрифта. Свойства должны записываться в следующем порядке:

selector{

font: font-style font-variant font-weight font-size/line-height font-family;

}

При этом font-size и font-family являются обязательными параметрами, а остальные задаются по необходимости

Сокращенную запись стоит использовать всегда, когда изменяется три и более свойств шрифта.

Для отображения шрифта на странице необходимо его наличие на компьютере пользователя. Это касается и разных начертаний одного и того же шрифта: для отображения **bold** или *italic* текста требуются отдельные шрифты, отличные от normal.

В комплектацию любой операционной системы входит определенный набор шрифтов, которые называют безопасными или веб-шрифтами (web safe fonts). И даже если вы планируете использовать безопасный шрифт, лучше перестраховаться и указать хотя бы один альтернативный шрифт а так же семейство шрифтов.

|  |  |
| --- | --- |
| 1  2  3 | .post{  font:12px Arial, Helvetica, sans-serif;  } |
|  |  |

Когда браузер встречает первый шрифт в списке **font**, он проверяет его наличие на компьютере пользователя. Если такого шрифта нет, браузер переходит к следующему имени из списка и также проверяет его на наличие. Заканчивают список обычно ключевым словом, которое описывает семейство шрифта - **serif**, **sans-serif**, **cursive**, **fantasy** или **monospace**. Если ни одного из явно указанных шрифтов в системе не оказалось, браузер применит любой из заданного семейства.

Безопасные веб-шрифты и их семейства можно посмотреть [здесь](http://markupacademy.com/upload/files/4/9/287/web-safe-fonts.pdf).

## Свойства текста

### Color

### Letter Spacing

Свойство letter-spacing задает интервал между символами. В качестве единицы измерения можно использовать px и em.

### Word Spacing

### Text Transform

Свойство text-transform определяет регистр символов: capitalize (Первый Буква Каждого Слова Прописная), uppercase (ВСЕ ПРОПИСНЫЕ) или lowercase (все строчные). Значение none отменяет трансформацию.

### Text Decoration

Свойство text-decoration добавляет оформление текста: underline (подчеркивания), line-through (~~перечеркивания~~) или overline (линии над текстом). Значение none отменяет декорирование.

### White Space

Свойство white-space определяет поведение слов, разделенных пробелом: normal - отображает текст с автоматическим переносом строк, nowrap запрещает перенос строк, то есть весь текст будет отображаться одной строкой (возможен только принудительный разрыв с помощью тега <br>), pre показывает текст с сохранением всех пробелов и переносов, которые есть в HTML-коде. Значение normal отменяет изменения.

### Text Align

Свойство text-align задает горизонтальное выравнивание текста: center (по центру), justify (по ширине), left (по левому краю) и right (по правому краю).

### Direction

Свойство direction устанавливает направление чтения/отображения текста: rtl - справа налево, ltr - слева направо (значение по умолчанию).

### Text indent

Свойство text-indent устанавливает отступ первой строки текстового блока (красную строку). Применяется только к блочным элементам. Допускается отрицательное значение.

Например, с помощью больших отрицательных значений можно спрятать текст при подмене текста картинкой.

### Vertical Align

Свойство vertical-align выравнивает элемент по вертикали относительно своего родителя (только в ячейках таблицы) или окружающих элементов (только для строчных тегов). Может принимать значения: baseline (выравнивание по базовой линии), middle (выравнивание вертикально по центру), top (выравнивание по верхнему краю), bottom (выравнивание по нижнему краю), sub (отображение элемента в виде нижнего индекса), super (отображение элемента в виде верхнего индекса).

## Свойства фона

### Background Color

### Background Image

### Background Repeat

Возможные значения: repeat (повторять по вертикали и горизонтали, это значение по умолчанию)

### Background Attachment

Свойство background-attachment устанавливает возможность прокрутки фонового изображения вместе с содержимым элемента.

### Background Position

### Background

Свойство background это сокращенная форма записи свойств фона. Значения должны идти в порядке:

body{

background: background-color background-image background-repeat background-position;

}

## Свойства отображения

### Display

Свойство display задаёт способ отображения элемента. Любой элемент можно отображать как: block (блочный), inline (строчный), inline-block (строчно-блочный), list-item (элемент списка), table (таблицу), table-cell (ячейку таблицу), table-footer-group (низ таблицы), table-header-group (шапку таблицы), table-row (строку таблицы), table-row-group (область данных таблицы), none (не отображать совсем).

### Visibility

Свойство visibility задает видимость элемента. Допустимые значения: visible (видимый), hidden (скрытый)

Visibility:hidden, в отличии от display:none резервирует место в потоке под скрываемый элемент.

## Свойства блока

### Width

### Margin

могут иметь отрицательные значения. Значение auto указывает, что размер отступов будет автоматически рассчитан браузером.

### Padding

Значения могут задаваться в любых единицах измерения (px, em, %) и не могут быть отрицательными. Значение auto также не работает.

### Border

Можно задавать border-width (толщину границы), border-color (цвет границы), border-style (стиль границы, в основном используется dotted, dashed, solid). Все эти значения можно задавать во отдельности или, что правильнее, воспользоваться сокращенной формой записи:

.box{

border: border-width border-style border-color;

}

для свойства border-width применимы теже правила записи значений, что и для свойств padding и margin

### Реализация отступов

Вертикальные интервалы между элементами или логическими блоками страницы реализуем внешним отступом снизу для этих элементов. Это гарантирует, что при добавлении в HTML нового неописанного элемента между ним и предыдущим элементом будет отступ.

Горизонтальные отступы, а также отступы сверху стараемся реализовывать присвоением свойства padding для родительского элемента. Это позволит задать их единожды для всех элементов внутри.

## CSS Float

Все блочные элементы отображаются на странице сверху вниз, а строчные - слева направо в том же порядке в котором они расположены в HTML-коде. При этом все элементы "всплывают" вверх пока не упрутся в предыдущий элемент или границу документа. Такая логика позволяет легко прогнозировать результат вывода элементов и управлять ими.

Порядок вывода элементов на странице и называется "потоком".

Свойство **float** позволяет изменить направление "всплытия" на влево(значение left) или вправо (значение right). Это может понадобиться, например, когда слева нужно поместить изображение, которое справа будет обтекать текст или чтобы организовать контент в несколько колонок.

Элемент с заданным свойством float:left или float:right (далее по тексту float-элемент) "всплывает" в указанном направлении и прижимается к границе родителя или документа. При этом, float-элемент вынимается из потока и блок который следует за ним, игнорирует его и всплывает вверх, на его место. При этом содержимое этого блока обтекает float-элемент со свободной стороны.

При использовании float-элементов нужно учитывать ряд нюансов:

* Отступы между float-элементом и обтекающем его контентом нужно задавать на float-элемент.
* По вертикали float-элементы позиционируется так же, как если бы они был в нормальном потоке (до задания для них свойства float, то есть свойство float не влияет на вертикальное расположение элемента.
* Float-элементам, как правило, задают ширину. Это позволят лучше контролировать их поведение.
* Если разместить подряд несколько float-элементов они выстроятся в одну строку. Если в первой строке всем не хватит места, они будут переноситься на следующие почти также как и слова в тексте.
* Свойство float делает строчные элементы блочными и соответственно восприимчивыми к свойствам характерным только для блоков (width, height, etc.)

Чтобы запретить обтекание float-элементов, нужно применить свойство **clear** для следующего за ним элемента.

Свойство **clear** имеет три возможных значения:

* left - запрещает обтекать float-элементы с правой стороны.
* right - запрещает обтекать float-элементы с левой стороны.
* both - запрещает обтекать float-элементы.

Понятие HasLayout введено Microsoft по отношению к браузерам IE. HasLayout определяет способность браузера правильно вычислять размеры блока относительно его содержимого. Но поскольку поведение которое описывает HasLayout частично характерно и для других браузеров, понятие прижилось и будет использоваться нами в отношении всех браузеров.

Обычно браузеры без проблем справляются с этой задачей, если только содержимое блока не является float-элементом(ами).

## CSS-позиционирование

CSS-позиционирование позволяет размещать элементы на странице, задав им точные координаты. При этом «точкой отсчета» может быть окно браузера, весь документ (тег html), любой родительский элемент или же нормальное положение элемента в потоке. Например, если «точкой отсчета» для позиционируемого логотипа является родительский блок «header», то говорят: «Логотип позиционируется относительно header».

За выбор метода позиционирования отвечает свойство position, которое может принимать четыре значения: static, absolute, relative и fixed.

Координаты позиционируемым элементам задаются свойствами top, right, bottom и left. Left и right задают смещение по оси X, top и bottom по оси Y относительно границ «точки отсчета». Координаты можно задавать как в абсолютных, так и относительных единицах измерения, при этом значения могут быть как положительными, так и отрицательными. Нельзя задавать два противоположных значения одновременно, например: left и right. В 99% случаев нужно задавать обе координаты – и по Х, и по Y, но возможно и указание только одной из них.

Свойства top, right, bottom и left не работают без указания свойства position.

Не рекомендуется использовать свойство **margin** для указания смещения абсолютно позиционируемых элементов, вместо традиционных **top**, **right**, **bottom** и **left**.

Абсолютное позиционирование применяется в случаях, когда нужно разместить один элемент поверх другого, или при необходимости, чтобы присутствие позиционируемого элемента никак не влияло на положение остальных.

Кроме того, **position:relative** , заданный на родительский элемент, устанавливает «точку отсчета» при абсолютном позиционировании (см. шаг position:absolute). Также он необходим для корректной работы свойства **z-index**.

Свойство z-index задает элементу координату по оси Z. Эта ось направлена из глубины монитора к пользователю. Таким образом, чем выше значение z-index, тем ближе элемент находиться к пользователю.

Свойство z-index актуально только для позиционируемых элементов. Оно не сработает если для элемента не задано свойство postion со значениями relative, absolute или fixed.

## CSS3

CSS3 это новая версия языка CSS, которая придет на смену CSS2.1. В ней появится много новых свойств, которые значительно расширят возможность CSS по оформлению web-страниц. CSS3 позволит разработчикам с помощью CSS-кода реализовывать некоторые вещи, которые ранее требовали использования изображений или JavaScript. Среди них: простая анимация, градиентные заливки, тени и закругленные углы, разбивка текста на колонки, новые типы селекторов и многое другое.

CSS3 обладает полной обратной совместимостью, поэтому Вы можете использовать все старые свойства наряду с новыми.

Стандарт CSS3 разбит на отдельные модули, многие из которых все еще находятся в разработке, поэтому реализация поддержки свойств сильно зависит от браузеров. Старые браузеры такие как IE7-IE8 не поддерживают большинство свойств CSS3. Ранние версии FF и Safari и Opera могут поддерживать свойства CSS3 только с вендорными префиксами -moz-, -webkit- или -o-. Учитывая это, целесообразно использовать генератор свойств CSS3, который сам подберет корректный набор свойств и префиксов.

На данном этапе мы будем использовать только те возможности CSS3 для которых, существует кросбраузерное решение. В основном это:

* **border-radius** - скругляет углы элементов;
* **text-shadow** - задает тень для текста;
* **box-shadow** - задает тень для элемента;
* **opacity** - задает уровень прозрачности для элемента;
* **rgba()** - зада цвета с определенным уровнем прозрачности;
* **linear-gradient** - создает градиентную заливку;
* **transform** - позволяет трансформировать элемент: маштабировать, поворачивать, искажать;
* **background-size** - регулирует размер фонового изображения;
* **multiple columns** - позволяет разбить текст на колонки.

CSS3 свойство **opacity** делает блок и его содержимое полупрозрачным.

Основная особенность этого свойства состоит в том, что значение прозрачности действует на все дочерние элементы внутри, а не только на фон. Это значит, что и фон и текст станут полупрозрачными и увеличить уровень прозрачности, добавляя opacity: 1, не получится

Реализовать прозрачность в браузерах IE6-8 можно с помощью фильтра. Фильтры прописываются в отдельном ie.css. Для корректной работы фильтра необходимо задать размер для блока:

.box {

filter: alpha(opacity=50); /\* IE 6-8 \*/

height:1%;

}

Обычно по дизайну полупрозрачным должен быть только фон элемента, а текст непрозрачным для сохранения его читабельности. Свойство opacity здесь не подходит, потому что сделает текст внутри элемента также частично прозрачным.

CSS-функция **rgba()** позволяет задавать цвета с определенным уровнем прозрачности для фона, теней, границ и т.д. Рассмотрим пример:

.box {

background: rgba(r, g, b, alfa);

}

Все элементы HTML изначально являются прямоугольными и в CSS2.1 не было никаких инструментов для изменения геометрии HTML-элементов. Поэтому для реализации блока со скругленными углами приходилось использовать соответствующие фоновые изображения для прямоугольного блока.

В CSS3 появилось свойство **border-radius**, которое позволяет скруглять углы HTML-элементам без использования графики. При этом радиус скругления можно задать для всех углов одновременно:

Или для каждого угла в отдельности, начиная с верхнего левого и дальше по часовой стрелке.

Скругления углов могут быть несимметричными: скруглятся сильнее по оси X или Y:

В таком случае радиусы скругления по оси X и Y замеряются отдельно, и задаются не одним, а двумя числами через слеш Rx/Ry для каждого угла:

.box{

border-radius: 43px/29px 43px/29px 43px/29px 43px/29px;

}

Предпочтительнее использовать сокращенную запись, но для одного угла можно задать скругление и таким образом:

.box{

border-top-left-radius: 16px;

}

В CSS2.1 не было никаких инструментов для создания теней блоков. Поэтому для реализации теней приходилось использовать соответствующие фоны.

В CSS3 появилось свойство **box-shadow**, которое позволяет задавать тени для блоков:

.box {

-webkit-box-shadow: offX offY blur spread color; /\* Safari 5, iOS, Android, Blackberry \*/

box-shadow: offX offY blur spread color;

}

Где offX и offY смещение по горизонтали и вертикали соответственно, могут принимать как положительные, так и отрицательные значения; blur - размер тени; spread - насыщенность тени; color - цвет тени, может задаваться как в системе hex:

.box {

-webkit-box-shadow: 5px 5px 5px #000;

box-shadow: 5px 5px 5px #000;

}

так и rgba

.box {

-webkit-box-shadow: 5px 5px 5px rgba(0, 0, 0, 0.5);

box-shadow: 5px 5px 5px rgba(0, 0, 0, 0.5);

}

Детальную информацию о тени блока так же можно найти в редакторе PhotoShop:

http://markupacademy.com/upload/tinymce/userfiles/css3/b-sh-1.png

Где 1 - цвет тени, 2 - прозрачность тени, 3 - направление смещения(работает по принципу солнечных часов), 4 - смещение тени и 5 - размер тени.

Также возможно реализовать внутреннюю тень добавив параметр "inset" в конец строки:

.box {

-webkit-box-shadow: 5px 5px 5px rgba(0, 0, 0, 0.5) inset;

box-shadow: 5px 5px 5px rgba(0, 0, 0, 0.5) inset;

}

Блоку, как и тексту, можно назначить несколько теней одновременно, перечислив их через запятую.

CSS3 дает нам возможность задать тень для текст. Ранее тень на тексте можно было сохранить только если вырезать его картинкой.

Свойство **text-shadow** задает внешнюю тень для текста:

.text {

text-shadow: offX offY blur color;

}

Еще одна, ранее недоступная, возможность CSS3 это создание градиентных заливок. Для этих целей предназначены CSS-функциии **linear-gradient()** и**radial-gradient()**.

Реализация градиентов сильно зависит от браузера и требует введения большого количества параметров поэтому целесообразно будет воспользоваться [онлайн генератором градиентов](http://www.colorzilla.com/gradient-editor//). Генератор позволяет вручную настроить цвета и точки перехода. Всю информацию о градиенте можно найти в редакторе PS:http://markupacademy.com/upload/tinymce/userfiles/CSS3/grad5.jpg

Но быстрее и проще импортировать градиент из изображения. Для этого нужно сохранить градиент ввиде однопиксельной полоски в формате png24. И воспользоваться инструментом "Import from image" генератора градиентов:http://markupacademy.com/upload/tinymce/userfiles/css3/css3-6.jpg

Далее в разделе "Preview" нужно выбрать направление градиентной заливки. Отмечаем галочкой пункт "IE9 support". Если необходима поддержка IE6-8 отмечаем галочкой пункт "IE".

<http://markupacademy.com/upload/tinymce/userfiles/css3/css3-7.jpg>

Копируем получившийся код в CSS-файл.

.box {

background: #1e5799; /\* Old browsers \*/

background: url(data:image/svg+xml;base64,); /\* IE 9 \*/

background: -moz-linear-gradient(top, #1e5799 0%, #2989d8 50%, #207cca 51%, #7db9e8 100%); /\* FF3.6+ \*/

background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,#1e5799), color-stop(50%,#2989d8), color-stop(51%,#207cca), color-stop(100%,#7db9e8)); /\* Chrome,Safari4+ \*/

background: -webkit-linear-gradient(top, #1e5799 0%,#2989d8 50%,#207cca 51%,#7db9e8 100%); /\* Chrome10+,Safari5.1+ \*/

background: -o-linear-gradient(top, #1e5799 0%,#2989d8 50%,#207cca 51%,#7db9e8 100%); /\* Opera 11.10+ \*/

background: -ms-linear-gradient(top, #1e5799 0%,#2989d8 50%,#207cca 51%,#7db9e8 100%); /\* IE10+ \*/

background: linear-gradient(top, #1e5799 0%,#2989d8 50%,#207cca 51%,#7db9e8 100%); /\* W3C \*/

}

В этом коде нужно изменить строку отвечающую за заливку цветом в старых браузерах:

.box {

background-color: #1e5799 !important; /\* Old browsers \*/

...

}

Если на этапе создания градиента был отмечен пункт IE, генератор добавит к коду фильтр, которым реализуется градиент в браузерах IE6-8. Фильтр невалиден и должен выноситься в ie.css. Для корректной работы фильтра элементу нужно задать размер.

/\* ie.css \*/

.box {

...

filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#1e5799', endColorstr='#7db9e8',GradientType=0 ); /\* IE6-8 \*/

height:1%;

}

CSS3 также позволяет трансформировать объект. Для этой цели служит свойство **transform**. Допустимы следующие значения:

* scale(0.1-2) - масштабирование;
* rotate(deg) - поворот на задданный угол, deg может принимать как положительные так и отрицательные значения;
* translate(x,y) - сдвиг по горизонтали/вертикали в px, x и y могут принимать как положительные так и отрицательные значения;
* skew(x,y) - скос по горизонтали и/или вертикали, x и y могут принимать как положительные так и отрицательные значения;

Необходимые значения перечисляются в одном свойстве через пробел:

.box{

-moz-transform: scale(0.7) rotate(25deg) translateX(0px) translateY(0px) skewX(10deg) skewY(0deg);

-webkit-transform: scale(0.7) rotate(25deg) translateX(0px) translateY(0px) skewX(10deg) skewY(0deg);

-o-transform: scale(0.7) rotate(25deg) translateX(0px) translateY(0px) skewX(10deg) skewY(0deg);

-ms-transform: scale(0.7) rotate(25deg) translateX(0px) translateY(0px) skewX(10deg) skewY(0deg);

transform: scale(0.7) rotate(25deg) translateX(0px) translateY(0px) skewX(10deg) skewY(0deg);

}

Реализовать трансформацию в браузерах IE7-8 можно только с помощью сложных фильтров для построения которых лучше воспользоваться специальными генераторами: [css-transform-generator](http://demos.aimweb.name/css-transform-generator/) или [matrix-calculator](http://www.boogdesign.com/examples/transforms/matrix-calculator.html). Из-за разницы работы transform и filter в IE может понадобится корректировка положения элемента с помощью magrin или других CSS свойств.

CSS3 свойство **box-sizing** может изменить стандартное поведение блочной модели CSS на более удобное, когда когда padding и border уже включены в указанные в CSS размеры блока.

.box{

-moz-box-sizing: border-box;

box-sizing: border-box;

}

Значение "content-box" переключает блочную модель CSS на стандартное поведение.

box-sizing:border-box облегчает вычисление размеров блоков в резиновых а адаптивных макетах, позволяя не брать в учет внутренние отступы и границы, а также позволяет использовать комбинированные единицы измерения (см. пример).

Еще одно полезное новшество CSS3 это возможность изменять размер фоновых изображений. Свойство **background-size** позволяет задать размер в пикселях или в процентах от родительского элемента.

.logo{

background-size: 100% auto;

}

.icon{

background-size: 36px 36px;

}

.box{

background-size: cover;

}

Это позволяет, например, использовать одни и теже изображения для десктопной и мобильной версии сайта или реализовывать поддержку Retina дисплеев, но об этом позже.

Так же доступны варианты автоматического вычисления размеров фона:

* background-size:cover - масштабирует изображение с сохранением пропорций так, чтобы оно полностью покрыло блок. При этом часть изображения может оказаться за пределами видимости.
* background-size:contain - масштабирует изображение с сохранением пропорций таким образом, чтобы картинка целиком поместилась внутрь блока. При этом у части блока может не быть фона.

Но эти способы менее кроссбраузерны.

Старые браузеры не поддерживают большинство свойств CSS3 (см. таблицу).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | IE7 | IE8 | IE9 | IE10 | FF4+ | Safary5+ | Chrome | Opera 11+ |
| border-radius | нет | нет | да | да | да | да | да | да |
| text-shadow | нет | нет | нет | да | да | да | да | да |
| box-shadow | нет | нет | да | да | да | да | да | да |
| opacity/rgba | да\* | да\* | да | да | да | да | да | да |
| font-face | да | да | да | да | да | да | да | да |
| gradient | да\* | да\* | да | да | да | да | да | да |
| resize | нет | нет | нет | да | да | да | да | да |
| word-wrap | да | да | да | да | да | да | да | да |
| overflow-x  overflow-y | да | нет | да | да | да | да | да | да |
| transform | да\* | да\* | да | да | да | да | да | да |
| background-size, background-origin, background-clip | нет | нет | да | да | да | да | да | да |
| transition | нет | нет | нет | да | да | да | да | да |
| border-image | нет | нет | нет | нет | да | да | да | да |
| multiple columns | нет | нет | нет | да | да | да | да | да |
| multiple backgrounds | нет | нет | да | да | да | да | да | да |
| box-sizing | нет | да | да | да | да | да | да | да |

да\* - возможно реализовать с использованием фильтров.

Но благодаря хорошей обратной совместимости мы можем не переживать об этом и выполнять поддержку старых браузеров по принципу "изящной деградации" - сохранение функциональности в ущерб оформлению. В этом случае, в браузерах которые не поддерживают новые свойства, блоки просто не будут иметь скруглений, теней, их фоны не будут полупрозрачными и т.д., но при этом они будут того же цвета, размера и выполнять туже функцию что и во всех современных браузерах.

По требованию клиента мы можем сохранить некоторые эффекты и для старых браузеров. Мы можем использовать для них отдельный CSS-файл и альтернативные фоновые изображения или воспользоваться фреймверком [PIE.htc](http://css3pie.com/) (самостоятельно).

Мы расмотрели наиболее кросбраузерные возможности CSS3. Конечно это далеко не полный список возможностей новой редакции CSS, но вполне достаточный для реализации проектов средней сложности.

И если вы все еще думаете, что CSS3 дал нам возможность использовать в проектах тени, скругления и градиенты, задайтесь вопросом: "Могли ли мы раньше использовать в проектах тени, скругления, градиенты и т.д.?" - Конечно ДА! Могли ли мы делать это только средствами CSS? - НЕТ!

Кроме очевидного, CSS3 обладает следующими преимуществами:

1. Освобождает от необходимости вырезать лишние картинки.
2. Улучшает читабельность кода, т.к. больше ненужны громоздкие HTML-конструкции с многоярусной вложенностью для реализации теней, скруглений и пр.
3. Ускоряет работу над проектом и скорость загрузки страницы.
4. CSS3 предоставляет много новых инструментов для решения старых задач.

А углубившись в изучение этой технологии вы быстро поймете что CSS3 это еще и анимация, и 3D моделирование и даже функциональное программирование, но это уже самостоятельн

# 