**Основы CSS3. Свойства**

|  |  |
| --- | --- |
| * [Цвет в CSS](https://metanit.com/web/html5/6.1.php) * [Стилизация шрифтов](https://metanit.com/web/html5/6.2.php) * [Внешние шрифты](https://metanit.com/web/html5/6.3.php) * [Высота шрифта](https://metanit.com/web/html5/6.4.php) * [Форматирование текста](https://metanit.com/web/html5/6.5.php) * [Стилизация абзацев](https://metanit.com/web/html5/6.6.php) * [Стилизация списков](https://metanit.com/web/html5/6.7.php) * [Стилизация таблиц](https://metanit.com/web/html5/6.8.php) * [Блочная модель](https://metanit.com/web/html5/6.9.php) * [Внешние отступы](https://metanit.com/web/html5/6.10.php) * [Внутренние отступы](https://metanit.com/web/html5/6.11.php) | * [Границы](https://metanit.com/web/html5/6.12.php) * [Размеры элементов. Box-sizing](https://metanit.com/web/html5/6.13.php) * [Фон элемента](https://metanit.com/web/html5/6.14.php) * [Создание тени у элемента](https://metanit.com/web/html5/6.15.php) * [Контуры элементов](https://metanit.com/web/html5/6.16.php) * [Обтекание элементов](https://metanit.com/web/html5/6.17.php) * [Прокрутка элементов](https://metanit.com/web/html5/6.18.php) * [Линейный градиент](https://metanit.com/web/html5/6.19.php) * [Радиальный градиент](https://metanit.com/web/html5/6.20.php) |

**Цвет в CSS**

В CSS широкое распространение находит использование цветов. Чтобы установить цвет текста, фона или границы, нам надо указать цвет.

Например, определим красный цвет для фона элемента div:



В CSS есть несколько различных свойств, которые в качестве значения требует определенный цвет.

Например, за установку цвета текста отвечает свойство **color**, за установку фона элемента – свойство **background-color**, а за установку цвета границы – **border-color**.

Существует несколько различных способов определения цвета текста.

* **Шестнадцатеричного значение**. Оно состоит из отдельных частей, которые кодируют в шестнадцатеричной системе значения для красного, зеленого и синего цветов.

Например, **#1C4463**.

Здесь первые два символа 1C представляю значение красной компоненты цвета, далее 44 – значение зеленой компоненты цвета и 63 – значение уровня синего цвета. Финальный цвет, который мы видим на веб-странице, образуется с помощью смешивания этих значений.

Если каждое из трех двухзначных чисел содержит по два одинаковых символа, то их можно сократить до одного. Например, #5522AA можно сократить до #52A, или, к примеру, #eeeeee можно сократить до #eee. При этом не столь важно, в каком регистре будут символы.

* **Значение RGB**. Значение RGB также представляет последовательно набор значений для красного, зеленого и синего цветов (Red, Green, Blue). Значение каждого цвета кодируется тремя числами, которые могут представлять либо процентные соотношения (0–100%), либо число от 0 до 255. Например:



Здесь каждый цвет имеет значение 100%. И в итоге при смешивании этих значений будет создаваться белый цвет. А при значениях в 0% будет генерироваться черный цвет:



Между 0 и 100% будут находиться все остальные оттенки.

Но, как правило, чаще применяются значения из диапазона от 0 до 255. Например,



* **Значение RGBA**. Это тоже самое значение RGB плюс компонент прозрачности (Alpha). Компонент прозрачности имеет значение от 0 (полностью прозрачный) до 1 (не прозрачный). Например:



* **Значение HSL**.

HSL представляет аббривиатуру:

Hue – тон,

Saturation – насыщенность и

Lightness – осветленность.

HSL задает три значения.

*Первое* значение Hue угол в круге оттенков – значение в градусах от 0 до 360. Например, красный – 0 (или 360 при полном обороте круга). Каждый цвет занимает примерно 51°.

*Второе* значение – Saturation – представляет насыщенность, то указывает, насколько чистым является цвет. Насыщенность определяется в процентах от 0 (полное отсутствие насыщенности) до 100% (яркий, насыщенный цвет).

*Третье*значение – Lightness – определяет осветленность и указывается в процентах от 0 (полностью черный) до 100 (полностью белый).

Для получения чистого цвет применяется значение 50 %.

Например:



Данный цвет является эквивалентом значений #1C4463 и rgb(28, 68, 99)

* **Значение HSLA**. Аналогично RGBA здесь к HSL добавляется компонента прозрачности в виде значения от 0 (полностью прозрачный) до 1 (не прозрачный). Например:



* **Строковые значения**. Существует ряд константных строковых значений, например, red (для красного цвета) или green (для зеленого цвета). К примеру,



является эквивалентом



Полный перечень цветов можно найти на странице

<https://developer.mozilla.org/en-US/docs/Web/CSS/color_value>

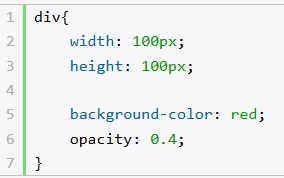
Существует множество бесплатных онлайн-генераторов цвета, где можно настроить и посмотреть цвет в нужном формате. Например, [генератор цвета на mdn](https://developer.mozilla.org/ru/docs/Web/CSS/CSS_Colors/Color_picker_tool).

### **Прозрачность**

Ряд настроек цвета позволяют установить значение для альфа-компоненты, которая отвечает за прозрачность.

Но также в CSS есть специальное свойство, которое позволяет установить прозрачность элементов – свойство **opacity**.

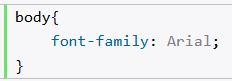
В качестве значения оно принимает число от 0 (полностью прозрачный) до 1 (не прозрачный):



## Стилизация шрифтов

### Семейство шрифтов

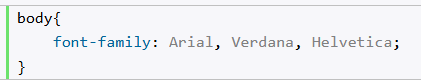
Свойство **font-family** устанавливает семейство шрифтов, которое будет использоваться. Например:



В данном случае устанавливается шрифт Arial.

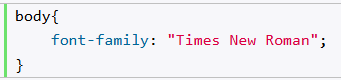
Шрифт свойства font-family будет работать, только если у пользователя на локальном компьютере имеется такой же шрифт. По этой причине нередко выбираются стандартные шрифты, которые широко распространены, как Arial, Verdana и т.д.

Также нередко применяется практика нескольких шрифтов:

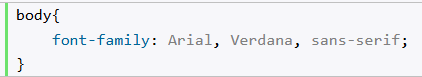


В данном случае основным шрифтом является первый – Arial. Если он на компьютере пользователя не поддерживается, то выбирается второй и т.д.

Если название шрифта состоит из нескольких слов, например, Times New Roman, то все название заключается в кавычки:



Кроме конкретных стилей также могут использоваться общие универсальные шрифты, задаваемые с помощью значений **sans-serif** и **serif**:



Так, если ни Arial, ни Verdana не поддерживаются на компьютере пользователя, то используется sans-serif – универсальный шрифт без засечек.

### **Типы шрифтов**

#### Шрифты с засечками

Шрифты с засечками названы так, потому что на на концах основных штрихов имеют небольшие засечки. Считается, что они подходят для больших кусков текста, так как визуально связывают одну букву с другой, делая текст более читабельным.

Распространенные шрифты с засечками: Times, Times New Roman, Georgia, Garamond. Универсальный обобщенный шрифт с засечками представляет значение **serif**.

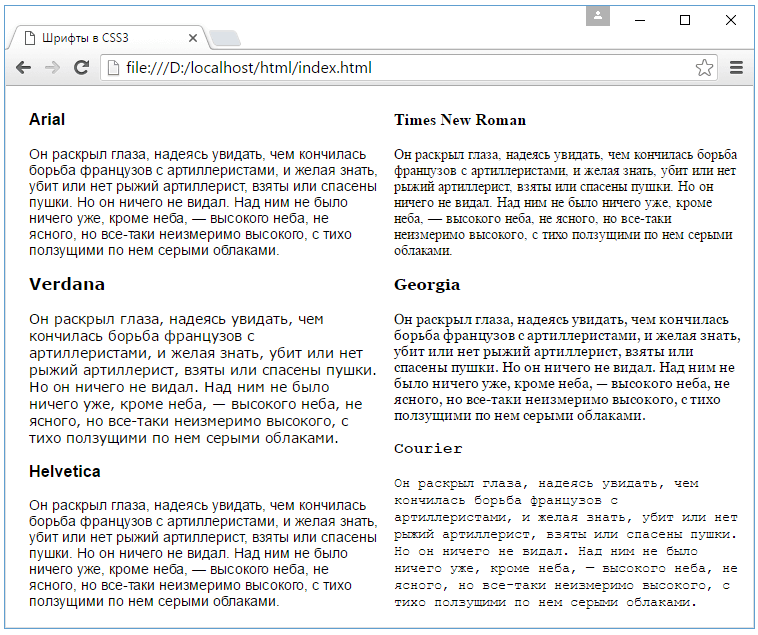
#### Шрифты без засечек

В отличие от шрифтов с засечками шрифты из этой группы не имеют засечек. Наиболее распространенные шрифты этой группы: Arial, Helvetica, Verdana.

#### Моноширинные шрифты

Моноширинный шрифт преимущественно применяется для отображения программного кода и не предназначен для вывода стандартного текста статей. Свое название эти шрифты получили от того, что каждая буква в таком шрифте имеет одинаковую ширину. Примеры подобных шрифтов: Courier, Courier New, Consolas, Lucida Console.

Примеры шрифтов:



### **Толщина шрифта**

Свойство **font-weight** задает толщину шрифта. Оно может принимать 9 числовых значений: 100, 200, 300, 400,...900. 100 – очень тонкий шрифт, 900 – очень плотный шрифт.

В реальности чаще для этого свойства используют два значения: normal (нежирный обычный текст) и bold (полужирный шрифт):



### **Курсив**

Свойство **font-style** позволяет выделить текст курсивом. Для этого используется значение italic:

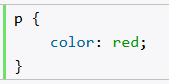


Если надо отменить курсив, то применяется значение normal:



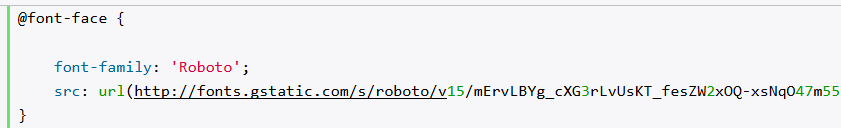
### **Цвет шрифта**

Свойство **color** устанавливает цвет шрифта:



## Внешние шрифты

Не всегда стандартные встроенные шрифты, как Arial или Verdana, могут быть удобны. Нередко встречается ситуация, когда веб-дизайнер хочет воспользоваться возможностями какого-то другого шрифта, которого нет среди встроенных, но который доступен из внешнего файла. Такой шрифт можно подключить с помощью директивы **font-face**:

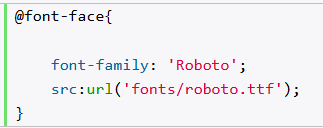


Свойство font-family задает название шрифта, а свойство src – путь к шрифту.В данном случае веб-страница будет подгружать шрифт, который расположен в интернете по адресу

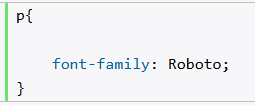
*http://fonts.gstatic.com/s/roboto/v15/mErvLBYg\_cXG3rLvUsKT\_fesZW2xOQ-xsNqO47m55DA.woff2*

В качестве альтернативы можно загрузить файл шрифта на локальный компьютер и уже оттуда подгружать его на веб-страницу.

Как правило, для хранения своих шрифтов рядом с веб-страницей создается папка *fonts*:



После подключения шрифта, его можно использовать в стилях:



В данном случе используется шрифт Roboto, созданный компанией Google и определенный в файле в формате woff2. Однако не все браузеры поддерживают данный формат шрифтов.

Существует несколько различных форматов шрифтов:

TrueType( расширение ttf),

Open Type (расширение otf),

Embedded Open Type (расширение eot),

Web Open Font Format (woff/woff2),

Scalable Vector Graphic (svg).

Разные браузеры могут поддерживать разные шрифты. И чтобы решить проблему поддержки шрифтов создатели шрифта часто создают сразу несколько форматов.

Можно сразу эти форматы определить. Например:



Как и в предыдущем случае свойства ***font-family*** и ***src*** задают название и путь к шрифту.

Но теперь для совместимости добавляется еще одно свойство ***src***.

Второе свойство src устанавливает сразу несколько шрифтов.

Первым шрифтом также идет шрифт в формате ***EOT***, но теперь к расширению файла ***.eot*** добавляется значение ***?#iefix***.

Это делается для совместимости с версиями Internet Explorer 6–8. Если после ***.eot*** не будет добавляться это значение, то шрифт может неправильно отображаться в Internet Explorer 6-8.

После параметра url идет определение формата шрифта:

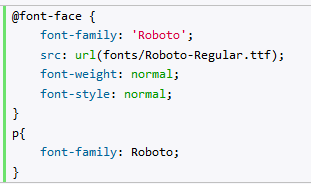


Когда браузер начнет загружать веб-страницу, на которой таким образом определен шрифт, то браузер не будет подгружать все шрифты во всех форматах, а загрузит только первый шрифт, который для него окажется понятным.

#### Разные версии шрифта

Загрузим шрифт Roboto по ссылке [Roboto](https://www.google.com/fonts/download?kit=hY6_qqcqX_JNp8IlbByiXfesZW2xOQ-xsNqO47m55DA) на локальный компьютер. Распакуем загруженный архив в папку, которую назовем fonts, и положим эту папку в один каталог рядом с веб-страницей. В загруженной папке шрифта Roboto можно увидеть, что она содержит не один файл, а сразу несколько:

|  |  |
| --- | --- |
|  | Зачем нам столько файлов? Дело в том, что каждый шрифт должен определять отдельный стиль для обычного текста, для текста, выделенного курсивом, для текста, выделенного жирным, для текста, сочетающего выделение жирным и курсивом и т.д.  Чтобы браузер мог автоматически распознавать разные варианты шрифта, к директиве @font-face добавляются свойства **font-weight** и**font-style**, которые соответственно устанавливают выделение жирным и выделение курсивом: |



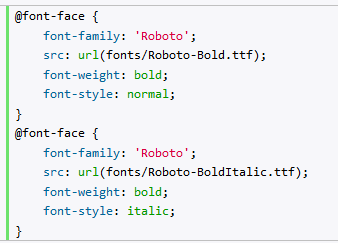
Поскольку версия шрифта ***Roboto-Regular.ttf*** применяется для текста, не выделенного курсивом и жирным, то вместе с ним устанавливаются значения:



То есть тем самым устанавливаем, что выделения курсивом не будет (***font-style: normal;***) и выделения жирным не будет (***font-weight: normal;***)

Кроме версии ***Roboto-Regular.ttf***, как видно выше на картинке, в папке есть еще другие версии шрифта ***Roboto***. Например, курсивная версия шрифта ***Roboto-Italic.ttf*** и ряд других.

Если хотим, чтобы при выделении курсивом браузер использовал именно курсивную версию, то надо добавить еще одну директиву font-face:



Значение font-weight: bold указывает, что данная версия шрифта применяется при выделении жирным.

Теперь используем все эти шрифты:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Шрифты в CSS3</title>

<style>

@font-face {

font-family: 'Roboto';

src: url(fonts/Roboto-Regular.ttf);

font-weight: normal;

font-style: normal;

}

@font-face {

font-family: 'Roboto';

src: url(fonts/Roboto-Italic.ttf);

font-weight: normal;

font-style: italic;

}

@font-face {

font-family: 'Roboto';

src: url(fonts/Roboto-Bold.ttf);

font-weight: bold;

font-style: normal;

}

@font-face {

font-family: 'Roboto';

src: url(fonts/Roboto-BoldItalic.ttf);

font-weight: bold;

font-style: italic;

}

p{

font-family: Roboto;

}

</style>

</head>

<body>

<p>Стиль Roboto может выделять <i>курсивом</i> и <b>жирным</b>, либо <b><i>и тем, и другим</i></b></p>

</body> </html>

Теперь к тексту в тегах <i></i>, который использует курсив, будет применяться версия "***Roboto-Italic.ttf***", а к тексту в тегах ***<b></b>*** – шрифт "***Roboto-Bold.ttf***".

### Источники шрифтов

В интернете можно найти множество нестандартных шрифтов. Наиболее популярным репозиторием шрифтов является [*https://www.google.com/fonts/*](https://www.google.com/fonts/) – набор шрифтов от компании *Google*.

Также другим известным репозиторием шрифтов является [*Font Squirrel*](https://www.fontsquirrel.com/fonts/list/hot_web).

Так же такой популярный шрифт как [*FontAwesome*](http://fontawesome.ru/). Он предоставляет множество различных интересных иконок, которые можно использовать на веб-странице.

## Высота шрифта

Для установки размера шрифта используется свойство **font-size**:

div{

font-size: 18px;

}

В данном случае высота шрифта составит 18 пикселей. Пиксели представляют наиболее часто используемые единицы измерения. Чтобы задать значение в пикселях, после самого значения идет сокращение "px".

Если к тексту явным образом не применяется высота шрифта, то используются значения браузера по умолчанию. Например, для простого текста в параграфах это 16 пикселей. Это базовый стиль текста.

Базовый стиль для разных элементов текста отличается: если для параграфов это 16 пикселей, то для заголовков h1 это 32 пикселя, для заголовков h2 – 24 пикселя и т..д.

Для измерения шрифта также можно использовать самые разные единицы измерения.

### **Ключевые слова**

В CSS имеется семь ключевых слов, которые позволяют назначить размер шрифта относительно базового:

* **medium**: базовый размер шрифта браузера (16 пикселей)
* **small**: 13 пикселей
* **x-small**: 10 пикселей
* **xx-small**: 9 пикселей
* **large**: 18 пикселей
* **x-large**: 24 пикселя
* **xx-large**: 32 пикселя

Например:

font-size: x-large;

### **Проценты**

Проценты позволяют задать значение относительно базового или унаследованного шрифта. Например:

font-size: 150%;

В данном случае высота шрифта будет составлять 150% от базового, то есть 16px \* 1,5 = 24px

Наследование шрифта может изменить финальное значение. Например, следующую ситуацию:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Шрифты в CSS3</title>

<style>

div {font-size: 10px;}

p {font-size: 150%;}

</style>

</head>

<body>

<div>

<p>Однажды в студеную зимнюю пору</p>

</div>

</body>

</html>

Здесь элемент p наследует от контейнера – блока div шрифт высотой в 10 пикселей. То есть 10 пикселей теперь будет базовым для параграфа.

Далее для элемента p определяется новая высота шрифта в 150%. Это значит, что финальная высота будет равна 10px \* 1,5 = 15px.

### **Единица еm**

Единица измерения еm во многом эквивалентна процентам. Так, 1em равен 100%, .5em равно 50% и т.д.

## Форматирование текста

### text-transform

Свойство ***text-transform*** изменяет регистр текста. Оно может принимать следующие значения:

* ***capitalize***: делает первую букву слова заглавной
* ***uppercase***: все слово переводится в верхний регистр
* ***lowercase***: все слово переводится в нижний регистр
* ***none***: регистр символов слова никак не изменяется

Например:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Шрифты в CSS3</title>

<style>

p.lowercase {text-transform: lowercase;}

p.uppercase {text-transform: uppercase;}

p.capitalize { text-transform: capitalize;}

</style>

</head>

<body>

<div>

<p class="capitalize">Текст использует значение capitalize</p>

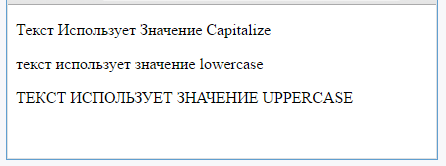
<p class="lowercase">Текст использует значение lowercase</p>

<p class="uppercase">Текст использует значение uppercase</p>

</div>

</body>

</html>



### **Свойство text-decoration**

Свойство **text-decoration** позволяет добавить к тексту некоторые дополнительные эффекты. Это свойство может принимать следующие значения:

* underline: подчеркивает текст
* overline: надчеркивает текст, проводит верхнюю линию
* line-through: зачеркивает текст
* none: к тексту не применяется декорирование

Например:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Шрифты в CSS3</title>

<style>

p.under {

text-decoration: underline;

}

p.over {

text-decoration: overline;

}

p.line {

text-decoration: line-through;

}

p.mixed {

text-decoration: underline line-through;

}

a.none {

text-decoration: none;

}

</style>

</head>

<body>

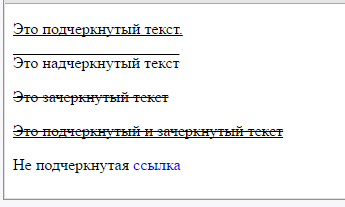
<div>

<p class="under">Это подчеркнутый текст.</p>

<p class="over">Это надчеркнутый текст</p>

<p class="line">Это зачеркнутый текст</p>

<p class="mixed">Это подчеркнутый и зачеркнутый текст</p>

 <p>Не подчеркнутая <a href="index.php" class="none">ссылка<a></p>

</div>

</body>

</html>

При необходимости можно комбинировать значения. Так, в предпоследнем случае применялся стиль:

p.mixed { text-decoration: underline line-through; }

### **Межсимвольный интервал**

Два свойства CSS позволяют управлять интервалом между символами и словами текста. Для межсимвольного интервала применяется атрибут **letter-spacing**, а для интервала между словами – **word-spacing**:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Шрифты в CSS3</title>

<style>

p.smallLetterSpace {

letter-spacing: -1px;

}

p.bigLetterSpace {

letter-spacing: 1px;

}

p.smallWordSpace{

word-spacing: -1px;

}

p.bidWordSpace{

word-spacing: 1px;

}

</style>

</head>

<body>

<div>

<h3>Обычный текст</h3>

<p>Над ним не было ничего уже, кроме неба...</p>

<h3>letter-spacing: -1px;</h3>

<p class="smallLetterSpace">Над ним не было ничего уже, кроме неба...</p>

<h3>letter-spacing: 1px;</h3>

<p class="bigLetterSpace">Над ним не было ничего уже, кроме неба...</p>

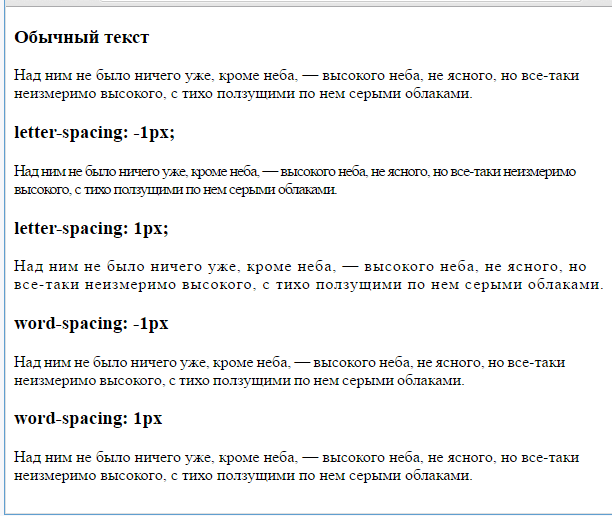
<h3>word-spacing: -1px</h3>

<p class="smallWordSpace">Над ним не было ничего уже, кроме неба...</p>

<h3>word-spacing: 1px</h3>

<p class="bigWordSpace">Над ним не было ничего уже, кроме неба...</p>

</div> </body> </html>



### **text-shadow**

С помощью свойства **text-shadow** можно создать тени для текста. Для этого свойства необходимо задать четыре значения: горизонтальное смещение тени относительно текста, вертикальное смещение тени относительно текста, степень размытости тени и цвет отбрасываемой тени. Например:

h1{

text-shadow: 5px 4px 3px #999;

}



В данном случае горизонтальное смещение тени относительно букв составляет 5 пикселей, а вертикальное смещение вниз – 4 пикселя. Степень размытости – 3 пикселя, и для тени используется цвет #999.

Если нам надо было бы создать горизонтальное смещение влево, а не вправо, как по умолчанию, то в этом случае надо было бы использовать отрицательное значение. Аналогично для создания вертикального смещения вверх также надо использовать отрицательное значение. Например:

h1{

text-shadow: -5px -4px 3px #999;

}



**Стилизация абзацев**

Отдельная группа свойств CSS позволяет стилизовать большие группы текста, например, установить высоту строки или выравнивание текста.

### **line-height**

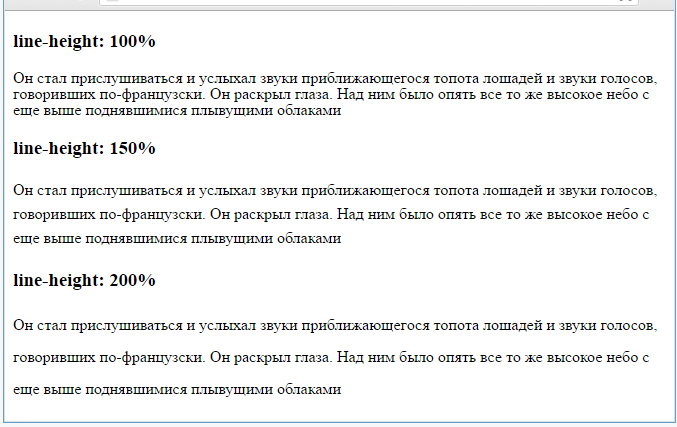
Свойство **line-height** определяет межстрочный интервал. Для его установки можно использовать пиксели, проценты или единицы em. Как правило, применяются либо проценты, либо em. Например:

p{

line-height: 150%;

}

Если это свойство не установлено, то по умолчанию используется значение line-height: 120%;.



### **text-align**

Свойство **text-align** выравнивает текст относительно одной из сторон веб-страницы. Оно принимает следующие значения:

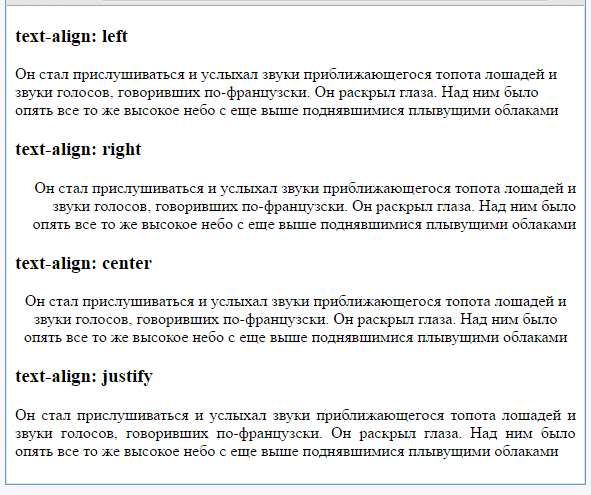
* left: текст выравнивается по левой стороне
* right: текст выравнивается по правой стороне
* justify: выравнивание по ширине, слова равномерно распределяются по строке
* center: выравнивание по центру

Например:

p{

text-align: left;

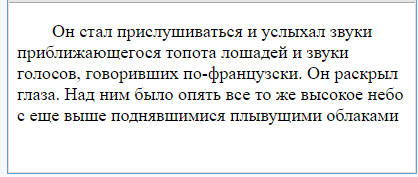
}



### **text-indent**

Свойство **text-indent** задает отступ первой строки абзаца. Для установки отступа могут применяться стандартные единицы измерения, например, em или пиксели:

|  |  |
| --- | --- |
| 1  2  3 | p{  text-indent: 35px;  } |



**Стилизация списков**

CSS предоставляет специальные свойства по стилизации списков. Одним из таких свойств является **list-style-type**. Оно может принимать следующие значения для нумерованных списков:

* decimal: десятичные числа, отсчет идет от 1
* decimal-leading-zero: десятичные числа, которые предваряются нулем, например, 01, 02, 03, … 98, 99
* lower-roman: строчные латинские цифры, например, i, ii, iii, iv, v
* upper-roman: заглавные латинские цифры, например, I, II, III, IV, V…
* lower-alpha: строчные латинские буквы, например, a, b, c..., z
* upper-alpha: заглавные латинские буквы, например, A, B, C, … Z

Для ненумерованных списков:

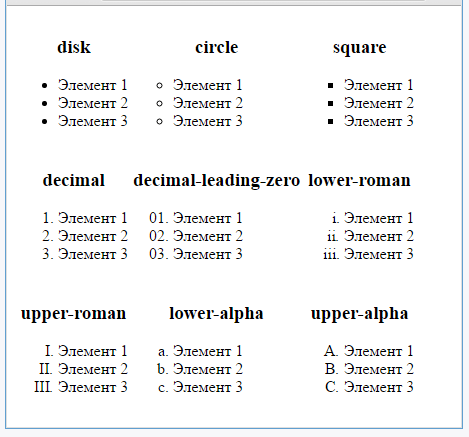
* disc: черный диск
* circle: пустой кружочек
* square: черный квадратик

Например:

ul{

list-style-type: square;

}



Чтобы вообще отключить маркеры у элементов списка, используется значение none:

ul{

list-style-type: none;

}

Данное свойство может применяться как ко всему списку, так и к отдельным элементам. Например:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Стилизация списков в CSS3</title>

<style>

.decimal{

list-style-type: decimal;

}

ol{

list-style-type: lower-roman;

}

</style>

</head>

<body>

<ol>

<li>Элемент 1</li>

<li class="decimal">Элемент 2</li>

<li>Элемент 3</li>

<li>Элемент 4</li>

</ol>

</body>

</html>

### **list-style-position**

Веб-браузеры обычно отображают маркеры списка слева от элементов списка. С помощью свойства **list-style-position** мы можем настроить их позиционирование. Это свойство принимает два значения: outside (по умолчанию) и inside (обеспечивает равномерное распределение по ширине).

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Стилизация списков в CSS3</title>

<style>

ul.outside{

list-style-position: outside;

}

ul.inside{

list-style-position: inside;

}

</style>

</head>

<body>

<h3>Inside</h3>

<ul class="inside">

<li>Он стал прислушиваться и услыхал звуки приближающегося топота лошадей и звуки голосов...</li>

<li>Он стал прислушиваться и услыхал звуки приближающегося топота лошадей и звуки голосов...</li>

</ul>

<h3>Outside</h3>

<ul class="outside">

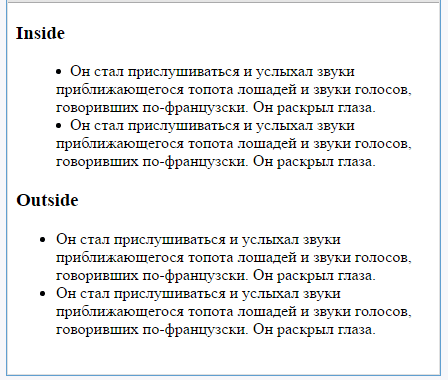
<li>Он стал прислушиваться и услыхал звуки приближающегося топота лошадей и звуки голосов...</li>

<li>Он стал прислушиваться и услыхал звуки приближающегося топота лошадей и звуки голосов...</li>

</ul>

</body>

</html>



### **list-style-image**

Свойство **list-style-image** позволяет задать в качестве маркера изображение:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Стилизация списков в CSS3</title>

<style>

ul{

list-style-image:url(phone\_touch.png);

}

</style>

</head>

<body>

<ul>

<li>iPhone 6S</li>

<li>Galaxy S7</li>

<li>Nexus 5X</li>

<li>Lumia 950</li>

</ul>

</body>

</html>

|  |  |
| --- | --- |
|  | Свойство list-style-image в качестве значения принимает путь к изображению url(phone\_touch.png), где "phone\_touch.png" – это название файла изображения. То есть в данном случае предполагается, что в одной папке с веб-страницей находится файл изображения phone\_touch.png. |

**Стилизация таблиц**

CSS предоставляет ряд свойств, которые помогают стилизовать таблицу:

* **border-collapse**: устанавливает, как будет стилизоваться граница смежных ячеек
* **border-spacing**: устанавливает промежутки между границами смежных ячеек
* **caption-side**: устанавливает положение элемента caption
* **empty-cells**: задает режим отрисовки для пустых ячеек
* **table-layout**: определяет размеры таблицы

### **Установка таблицы**

Ранее для установки границы в таблице широко использовался атрибут border, например:

<table border="2px" >

Сейчас же тенденция для стилизации использовать только стили CSS, поэтому граница также задается через CSS с помощью стандартного свойства **border**:

table {

border: 1px solid #ccc; /\* граница всей таблицы \*/

}

tr {

border: 1px solid #ccc; /\* границы между строками \*/

}

td, th {

border: 1px solid #ccc; /\* границы между столбцами \*/

}

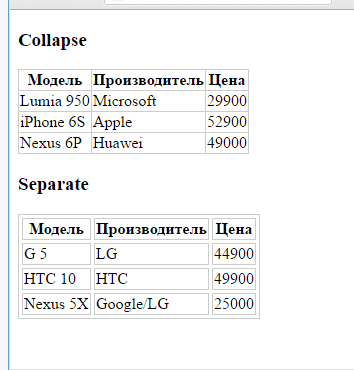
При установке границ между столбцами с помощью свойства border-collapse можно установить общую или раздельную границу между смежными ячейками:

* collapse: смежные ячейки имеют общую границу
* separate: смежные ячейки имеют отдельные границы, которые разделяются пространством

Если смежные ячейки имеют раздельные границы, то с помощью свойства border-spacing можно установить пространство между границами:







### **Пустые ячейки**

Свойство **empty-cells** позволяет стилизовать пустые ячейки с помощью одного из следующих значений:

* show: пустые ячейки отображаются, значение по умолчанию
* hide: пустые ячейки не отображаются







### **Позиционирование заголовка**

Свойство caption-side управляет позицией заголовка и может принимать следующие значения:

* top: позиционирование заголовка вверху (значение по умолчанию)
* bottom: позиционирование заголовка внизу

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



### **Управление размером таблицы**

С помощью свойства **table-layout** можно управлять размером таблицы. По умолчанию это свойство имеет значение auto, при котором браузер устанавливает ширину столбцов таблицы автоматически, исходя из ширины самой широкой ячейки в столбце. А из ширины отдельных столбцов складывается ширина всей таблицы.

Однако с помощью другого значения – fixed можно установить фиксированную ширину:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | table {  border: 1px solid #ccc;  border-spacing: 3px;  table-layout: fixed;  width:350px;  } |

### **Вертикальное выравнивание ячеек**

Как правило, содержимое ячеек таблицы выравнивается по центру ячейки. Но с помощью свойства **vertical-align** это поведение можно переопределить. Это свойство принимает следующие значения:

* top: выравнивание содержимого по верху ячейки
* baseline: выравнивание первой строки текста по верху ячейки
* middle: выравнивание по центру (значение по умолчанию)
* bottom: выравнивание по низу

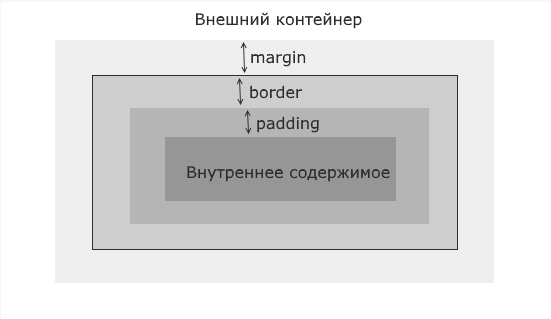
Свойство vertical-align применяется только к элементам <th> и <td>:

|  |  |
| --- | --- |
| 1  2  3  4  5 | td, th{  border: solid 1px #ccc;  vertical-align: bottom;  height: 30px;  } |

**Блочная модель**

Для веб-браузера элементы страницы представляют небольшие контейнеры или блоки. Такие блоки могут иметь различное содержимое – текст, изображения, списки, таблицы и другие элементы. Внутренние элементы блоков сами выступают в качестве блоков.

Схематично блочную модель можно представить следующим образом:



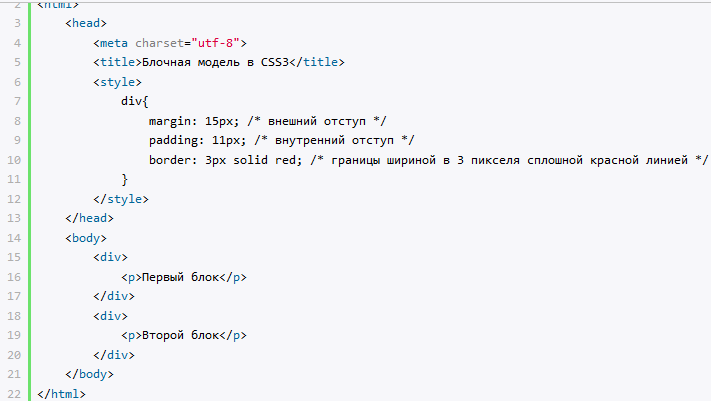
Пусть элемент расположен в каком-нибудь внешнем контейнере. Это может быть элемент body, div и так далее. От других элементов он отделяется некоторым расстоянием – внешним отступом, которое описывается свойством CSS **margin**. То есть свойство margin определяет расстояние от границы текущего элемента до других соседних элементов или до границ внешнего контейнера.

Далее начинается сам элемент. И в начале идет его граница, которая в CSS описывается свойством **border**.

После границы идет внутренний отступ, который в CSS описывается свойством **padding**. Внутренний отступ определяет расстояние от границы элемента до внутреннего содержимого.

Далее идет внутреннее содержимое, которое также реализует ту же блочную модель и также может состоять из других элементов, которые имеют внешние и внутренние отступы и границу.

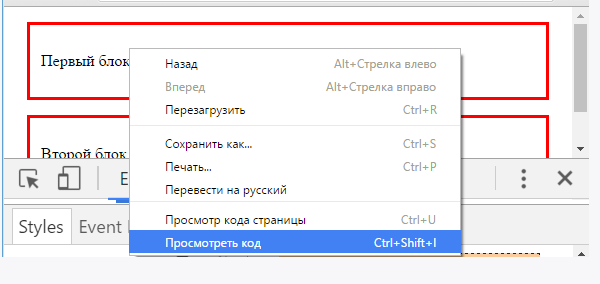
Например, определим следующую веб-страницу:



После запуска веб-страницы в браузере мы можем посмотреть блочную модель конкретных элементов.

Для этого надо нажать на нужный элемент правой кнопкой мыши и открывающемся контекстном меню выбрать пункт, который позволяет просмотреть исходный код элемента.

Для разных браузеров этот пункт может называться по разному. К примеру в Google Chrome это **Посмотреть код**:



В Mozilla Firefox этот пункт называется **Исследовать элемент**.

И по выбору данного пункта браузер откроет панель, где будет показан код элемента его стили и блочная модель:

|  |  |
| --- | --- |
|  | В этой модели мы можем увидеть, как задаются отступы элемента, его граница, посмотреть отступы от других элементов и при необходимости динамически поменять значения их стилей.  Если мы явным образом не указываем значения свойств margin, padding и border, то браузер применяет предустановленные значения. |

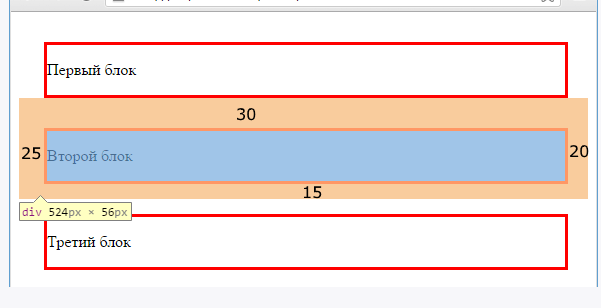
**Внешние отступы**

Свойство **margin** определяет отступ элемента от других элементов или границы внешнего контейнера. Существуют специальные свойства CSS для задания отступов для каждой стороны:

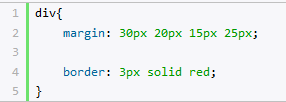
* **margin-top**: отступ сверху
* **margin-bottom**: отступ снизу
* **margin-left**: отступ слева
* **margin-right**: отступ справа

Например:

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



Можно вместо четырех свойств задать одно:



Свойство задается в формате:



Если значения для всех четырех отступов совпадает, то можем указать только одно значение:



В этом случае для всех четырех отступов будет использоваться 25 пикселей.

Для установки отступов можно использовать точные значения в пикселях (px) или em, либо процентные отношения, либо значение auto (автоматическая установка отступов).

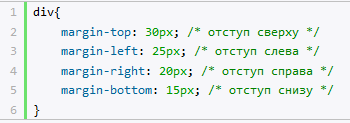
Например:



Значение 2 em определяет расстояние, которое в два раза больше размера шрифта элемента.

При использовании процентов веб-браузеры вычисляют размер отступов на основе ширины элемента-контейнера, в который заключен стилизуемый элемент.

В то же время, если несколько элементов у нас соприкасаются, то браузер выбирает наибольший отступ элемента, который затем и используется. Так, выше в примере использовался следующий стиль:



Между первым и вторым блоками расстояние будет равно 30 пикселям – значение свойства margin-top второго блока div, несмотря на то, что у первого блока div имеется свойство margin-bottom, равное 15 пикселям.

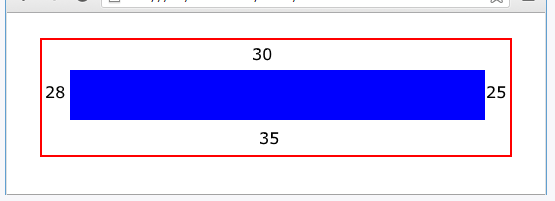
**Внутренние отступы**

Свойство **padding** задает внутренние отступы от границы элемента до его внутреннего содержимого. Как и для свойство margin, в CSS имеются четыре свойства, которые устанавливают отступы для каждой из сторон:

* **padding-top**: отступ сверху
* **padding-bottom**: отступ снизу
* **padding-left**: отступ слева
* **padding-right**: отступ справа

Например:

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

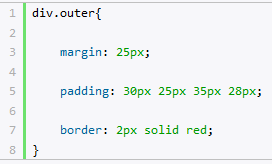


Для установки значения отступов, как и в margin, могут применяться либо конкретные значения в пикселях, так и процентные значения (относительно размеров элементов).

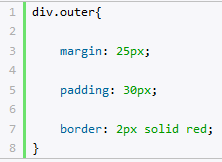
Для записи отступов также можно использовать сокращенную запись:



Например:



Если все четыре значения совпадают, то можно писать указать только одно значение для всех отступов:



**Границы**

Граница отделяется элемент от внешнего по отношению к нему содержимого. При этом граница является частью элемента.

Для настройки границы могут использоваться сразу несколько свойств:

* **border-width**: устанавливает ширину границы
* **border-style**: задает стиль линии границы
* **border-color**: устанавливает цвет границы

Свойство border-width может принимать следующие типы значений:

* Значения в единицах измерения, таких как em, px или cm

|  |  |
| --- | --- |
| 1 | border-width: 2px; |

* Одно из константных значений: thin (тонкая граница – 1px), medium (средняя по ширине – 3px), thick (толстая – 5px)

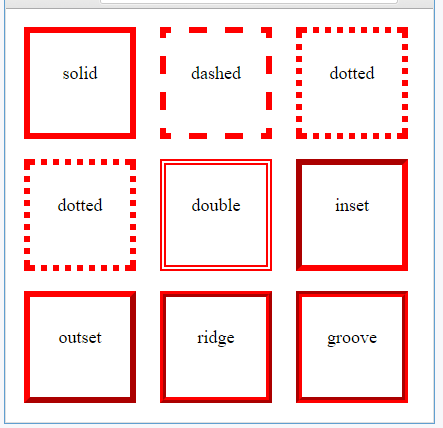
|  |  |
| --- | --- |
| 1 | border-width: medium; |

Свойство border-color в качестве значения принимает цвет CSS:

|  |  |
| --- | --- |
| 1 | border-color: red; |

Свойство border-style оформляет тип линии границы и может принимать одно из следующих значений:

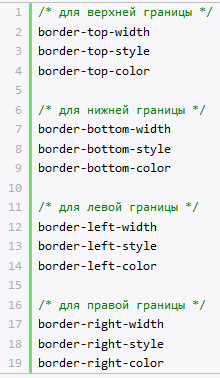
* none: граница отсутствует
* solid: граница в виде обычной линии
* dashed: штриховая линия
* dotted: линия в виде последовательности точек
* double: граница в виде двух параллельных линий
* groove: граница имеет трехмерный эффект
* inset: граница как бы вдавливается во внутрь
* outset: аналогично inset, только граница как бы выступает наружу
* ridge: граница также реализует трехмерный эффект



Например:

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

При необходимости мы можем определить цвет, стиль и ширину границы для каждой из сторон используя следующие свойства:



### **Свойство border**

Вместо установки по отдельности цвета, стиля и ширины границы мы можем использовать одно свойство – **border**:

|  |  |
| --- | --- |
| 1 | border: ширина стиль цвет |

Например:

|  |  |
| --- | --- |
| 1 | border: 2px solid red; |

Для установки границы для отдельных сторон можно использовать одно из свойств:

|  |  |
| --- | --- |
| 1  2  3  4 | *border-top*  *border-bottom*  *border-left*  *border-right* |

Их использование аналогично:

|  |  |
| --- | --- |
| 1 | border-top: 2px solid red; |

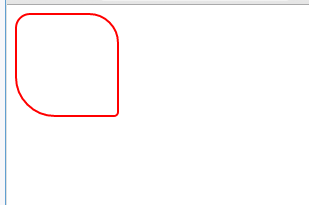
### **Радиус границы**

Свойство **border-radius** позволяет округлить границу. Это свойство принимает значение радиуса в пикселях или единицах em.

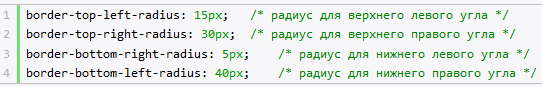
|  |  |
| --- | --- |
|  | Теперь каждый угол будет скругляться по радиусу в 30 пикселей: |

Так как у элемента может быть максимально четыре угла, то мы можем указать четыре значения для установки радиуса у каждого углов:

|  |  |
| --- | --- |
|  | border-radius: 15px 30px 5px 40px; |



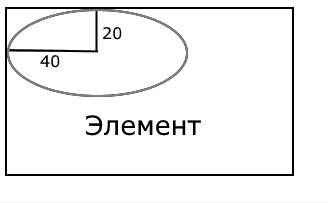
Вместо общей установки радиусов для всех углов, можно их устанавливать по отдельности. Так, предыдущее значение border-radius можно переписать следующим образом:



Также border-radius поддерживает возможность создания эллиптических углов. То есть угол не просто скругляется, а использует два радиуса, образуя в итоге душу эллипса:

|  |  |
| --- | --- |
| 1 | border-radius: 40px/20px; |

В данном случае полагается, что радиус по оси X будет иметь значение 40 пикселей, а по оси Y – 20 пикселей.



**Размеры элементов**

Размеры элементов задаются с помощью свойств **width** (ширина) и **height** (высота).

Значение по умолчанию для этих свойств – auto, то есть браузер сам определяет ширину и высоту элемента. Можно также явно задать размеры с помощью единиц измерения (пикселей, em) или с помощью процентов:

|  |  |
| --- | --- |
| 1  2  3 | width: 150px;  width: 75%;  height: 15em; |

Пиксели определяют точные ширину и высоту. Единица измерения **em** зависит от высоты шрифта в элементе. Если размер шрифта элемента, к примеру, равен 16 пикселей, то 1 em для этого элемента будет равен 16 пикселям. То есть если у элемента установить ширину в 15em, то фактически она составит 15 \* 16 = 230 пикселей. Если же у элемента не определен размер шрифта, то он будет взят из унаследованных параметров или значений по умолчанию.

Процентные значения для свойства width вычисляются на основании ширины элемента-контейнера.

Если, к примеру, ширина элемента body на веб-странице составляет 1000 пикселей, а вложенный в него элемент <div> имеет ширину 75%, то фактическая ширина этого блока <div> составляет 1000 \* 0.75 = 750 пикселей.

Если пользователь изменит размер окна браузера, то ширина элемента body и соответственно ширина вложенного в него блока div тоже изменится.

Процентные значения для свойства height работают аналогично свойству width, только теперь высота вычисляется по высоте элемента-контейнера.

Например:

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

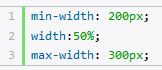
В то же время фактические размеры элемента могут в итоге отличаться от тех, которые установлены в свойствах width и height. Например:



|  |  |
| --- | --- |
|  | Как видно на скриншоте, в реальности значение свойства width - 200px - определяет только ширину внутреннего содержимого элемента, а под блок самого элемента будет выделяться пространство, ширина которого равна ширине внутреннего содержимого (свойство width) + внутренние отступы (свойство padding) + ширина границы (свойство border-width) + внешние отступы (свойство margin).  То есть элемент будет иметь ширину в 230 пикселей, а ширина блока элемента с учетом внешних отступов составит 250 пикселей.  Подобные расчеты следует учитывать при определении размеров элементов. |

С помощью дополнительного набора свойств можно установить минимальные и максимальные размеры:

* min-width: минимальная ширина
* max-width: максимальная ширина
* min-height: минимальная высота
* max-height: максимальная высота



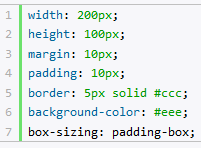
В данном случае ширина элемента равна 50% ширины элемента-контейнера, однако при этом не может быть меньше 200 пикселей и больше 300 пикселей.

### **Переопределение ширины блока**

Свойство box-sizing позволяет переопределить установленные размеры элементов.

Оно может принимать одно из следующих значений:

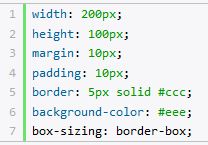
* content-box: значение свойства по умолчанию, при котором браузер для определения реальных ширины и высоты элементов добавляет к значениям свойств width и height ширину границы и внутренние отступы
* padding-box: указывает веб-браузеру, что ширина и высота элемента должны включать внутренние отступы как часть своего значения. Например, пусть у нас есть следующий стиль:



Здесь реальная ширина внутреннего содержимого блока будет равна 200px (width) – 10px (padding-left) – 10px (padding-right) = 180px.

*Стоит отметить, что большинство современных браузеров не поддерживают данное свойство.*

* border-box: указывает веб-браузеру, что ширина и высота элемента должны включать внутренние отступы и границы как часть своего значения. Например, пусть у нас есть следующий стиль:

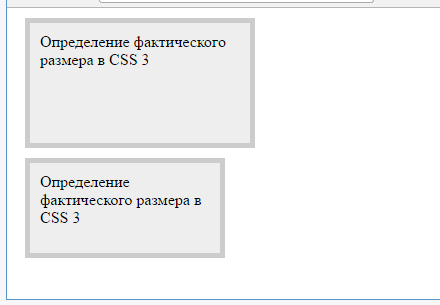


Здесь реальная ширина внутреннего содержимого блока будет равна 200px (width) – 10px (padding-left) – 10px (padding-right) – 5px (border-left-width) – 5px (border-right-width) = 170px.

Например, определим два блока, которые отличаются только значением свойства box-sizing:

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

В первом случае при определении размеров блока к свойствам width и height будут добавляться толщина границы, а также внутренние и внешние отступы, поэтому первый блок будет иметь большие размеры:



**Фон элемента**

Фон элемента описывается в CSS свойством background. Фактически это свойство представляет сокращение набора следующих свойств CSS:

* background-color: устанавливает цвет фона



* background-image: в качестве фона устанавливается изображение



* Это свойство принимает одно значение: ключевое слово url, после которого в скобках идет путь к файлу изображения. В данном случае имеется в виду, что в одной папке рядом с веб-страницей находится файл dubi.png.
* Также можно использовать абсолютные адреса URL, например:



* Либо можно использовать относительные адреса – относительно html-документа или корневого каталога сайта:



* background-repeat: устанавливает режим повторения фонового изображения по всей поверхности элемента
* background-size: устанавливает размер фонового изображения
* background-position: указывает позицию фонового изображения
* background-attachment: устанавливает стиль прикрепления фонового изображения к элементу
* background-clip: определяет область, которая вырезается из изображения и используется в качестве фона
* background-origin: устанавливает начальную позицию фонового изображения

Например:

|  |  |
| --- | --- |
|  | Первый блок окрашен в оттенок красного цвета, а второй блок устанавливает в качестве фона изображение. Все содержимое блока накладывается поверх фона: |

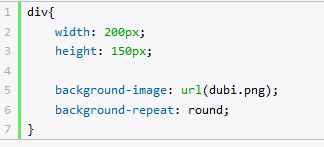
### Повторение изображения

На выше приведенном скриншоте видно, что CSS должным образом масштабирует изображение, чтобы наиболее оптимально вписать его в пространство элемента. Однако в связи с масшатбированием изображение может не полностью покрывать поверхность элемента, и поэтому для полного покрытия автоматически CSS начинает повторять изображение.

С помощью свойства background-repeat можно изменить механизм повторения. Оно может принимать следующие значения:

* repeat-x: повторение по горизонтали
* repeat-y: повторение по вертикали
* repeat: повторение по обеим сторонам (действие по умолчанию)
* space: изображение повторяется для заполнения всей поверхности элемента, но без создания фрагментов
* round: изображение должным образом масштабируется для полного заполнения всего пространства
* no-repeat: изображение не повторяется

Например:



### Размер изображения

Свойство background-size позволяет установить размер фонового изображения. Для установки размера можно использовать либо единицы измерения, например, пиксели, либо проценты, либо одно из предустановленных значений:

* contain: масштабирует изображение по наибольшей стороне, сохраняя аспектное отношение
* cover: масштабирует изображение по наименьшей стороне, сохраняя аспектное отношение
* auto: значение по умолчанию, изображение отображается в полный размер

Если нужно масштабировать изображение таким образом, чтобы оно оптимальнее было вписано в фон, то для обеих настроек можно установить значение 100%:



Если задаются точные размеры, то вначале указывается ширина, а потом высота изображения:

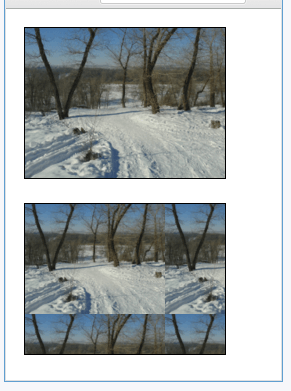


Можно задать точное значение для одного измерения – ширины или высоты, а для другого задать автоматические размеры, чтобы браузер сам выводил точные значения:



Например:

|  |  |
| --- | --- |
|  | Во втором случае изображение будет масштабироваться до размеров 140х110. Поскольку у нас еще остается место на элементе, то по умолчанию изображение будет повторяться для заполнения всей поверхности: |



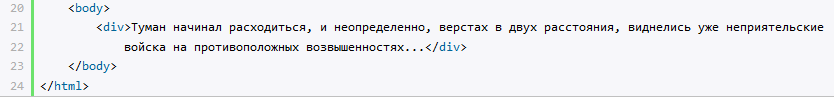
### Позиция изображения

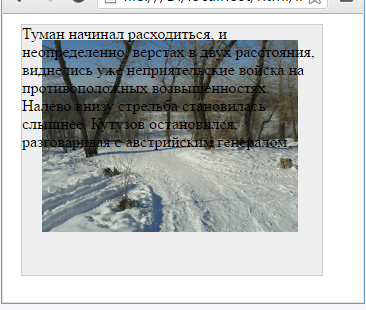
Свойство background-position управляет позицией фонового изображения внутри элемента. Оно может принимать отступы от верхнего левого угла элемента в единицах измерения, например, в пикселях в следующем формате:



Например:







Кроме того, данное свойство может принимать одно из следующих значений:

* top: выравнивание по верхнему краю элемента
* left: выравнивание по левому краю элемента
* right: выравнивание по правому краю элемента
* bottom: выравнивание по нижнему краю элемента
* center: изображение располагается по центру элемента

Например:



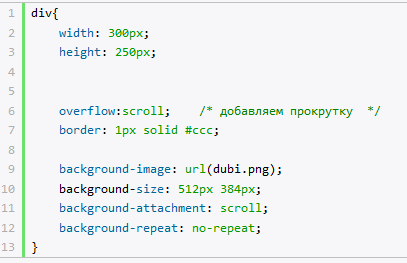
Здесь изображение выравнивается по верху и правому краю, то есть будет располагаться в правом верхнем углу элемента.

### **background-attachment**

Свойство background-attachment управляет, как фоновое изображение будет прикреплено к элементу. Это свойство может принимать следующие значения:

* fixed: фон элемента фиксирован вне зависимости от прокрутки внутри элемента
* local: по мере прокрутки внутри элемента фон изменяется
* scroll: фон фиксирован и не меняется при прокрутке, но в отличие от fixed несколько элементов могут использовать свой фон, тогда как при fixed создается один фон для всех элементов

Например:



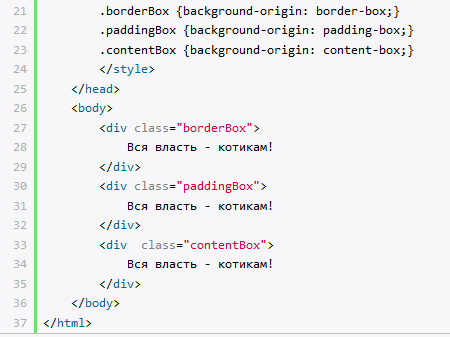
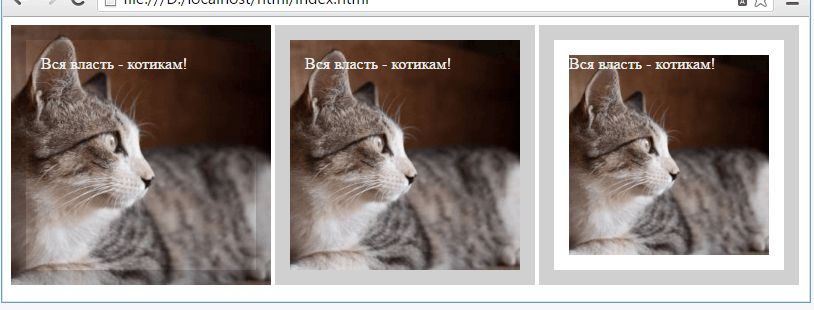
### **background-origin**

Свойство background-origin указывает позицию на изображении, с которой будет начинаться собственно фоновое изображение для элемента. Оно может принимать следующие значения:

* border-box: фон у элемента устанавливается начиная с его внешней границы, определяемой свойством border
* padding-box: фон устанавливается с учетом внутренних отступов
* content-box: фон устанавливается по содержимому элемента

Используем все три значения:



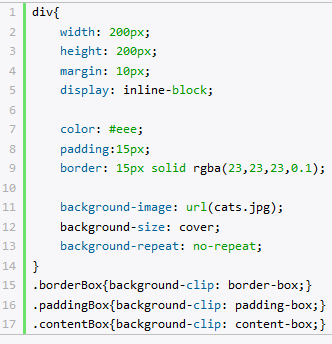
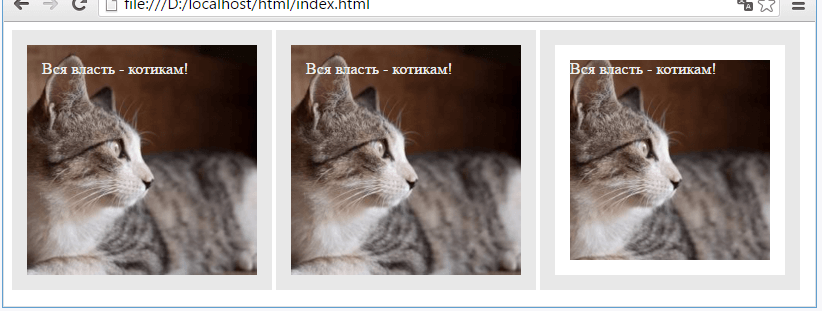


### **background-clip**

Свойство background-clip определяет, какая часть изображения используется для фона. Он принимает те же значения:

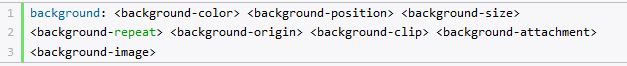
* border-box: изображение обрезается по границам элемента
* padding-box: из изображения исключается та часть, которая находится под границами элемента
* content-box: изображение обрезается по содержимому с учетом внутренних отступов

Например, если к предыдущей разметке мы применим следующие стили:



### **Свойство background**

Свойство background по сути является сокращением всех ранее рассмотренных свойств CSS в формате:



Например, если у нас есть следующий набор свойств:



То мы их можем сократить следующим образом:



## Создание тени у элемента

Свойство box-shadow позволяет создать у элемента тень. Это свойство может принимать сразу несколько значений:



* hoffset: горизонтальное смещение тени относительно элемента. При положительном значении тень смещается вправо, а при отрицательном – влево
* voffset: вертикальное смещение тени относительно элемента. При положительном значении тень смещается вниз, а при отрицательном – вверх
* blur: необязательное значение, которое определяет радиус размытия тени. Чем больше это значение, тем более размытыми будут края тени. По умолчанию имеет значение 0.
* spread: необязательное значение, которое определяет направление тени. Положительное значение распространяет тень во вне во всех направлениях от элемента, а отрицательное значение направляет тень к элементу
* color: необязательное значение, которое устанавливает цвет тени
* inset: необязательное значение, которое заставляет рисовать тент внутри блока элемента

Через запятую можно определить несколько различных теней:



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

## Контуры элементов

Концепция контуров похожа на использование границ у элементов. Не стоит путать или заменять границы контурами, они имеют разное значение. Контуры полезны тем, что позволяют выделить элемент, чтобы привлечь к нему внимание в какой-то ситуации. Контуры располагаются вне элемента сразу за его границами.

Контур в CSS 3 представлен свойством outline, хотя данное свойство является сокращением следующих свойств:

* outline-color: цвет контура
* outline-offset: смещение контура
* outline-style: стиль контура. Оно принимает те же значения, что и border-style:
  + none: контур отсутствует
  + solid: контур в виде обычной линии
  + dashed: штриховая линия
  + dotted: линия в виде последовательности точек
  + double: контур в виде двух параллельных линий
* outline-width: толщина контура

С помощью свойства outline данное определение контура можно сократить следующим образом:



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

## Обтекание элементов

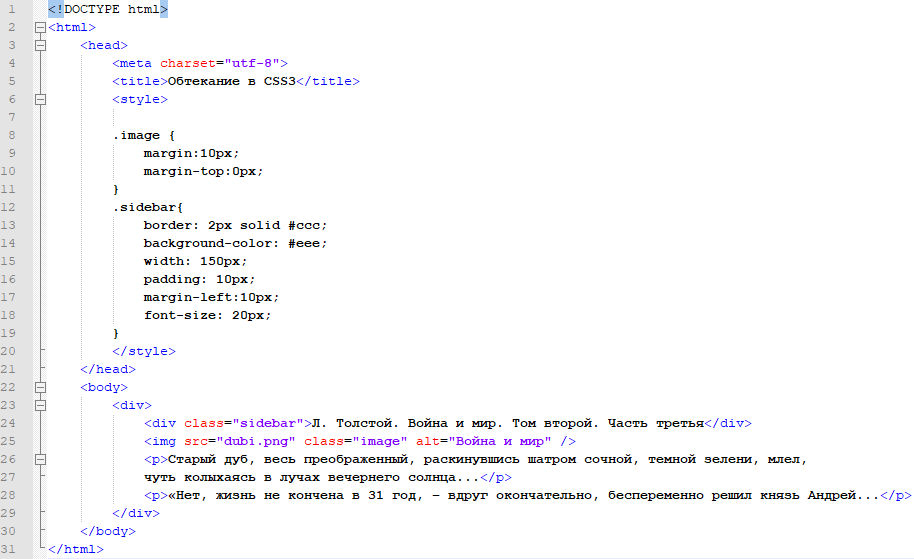
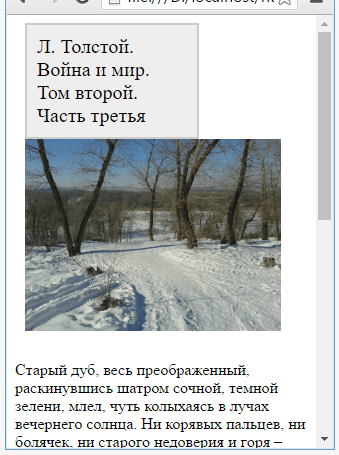
Как правило, все блоки и элементы на веб-странице в браузере появляются в том порядке, в каком они определены в коде html. Однако CSS предоставляет специальное свойство float, которое позволяет установить обтекание элементов, благодаря чему мы можем создать более интересные и разнообразные по своему дизайну веб-страницы.

Это свойство может принимать одно из следующих значений:

* left: элемент перемещается влево, а все содержимое, которое идет ниже его, обтекает правый край элемента
* right: элемент перемещается вправо
* none: отменяет обтекание и возвращает объект в его обычную позицию

При применении свойства float для стилизуемых элементов, кроме элемента img, рекомендуется установить свойство width.

Итак, представим, что нам надо на странице вывести слева от основного текста изображение, справа должен быть сайдбар, а все остальное место должно быть занято основным текстом статьи. Определим интерфейс страницы сначала без свойства float:



Теперь на той же странице применим свойство float, изменив стили следующим образом:

|  |  |
| --- | --- |
|  | Соответственно изменится и размещение элементов на странице: |

Элементы, к которым применяется свойство float, еще называют floating elements или плавающими элементами.

**Запрет обтекания. Свойство clear**

Иногда возникает необходимость запретить обтекания. Подобная задача может быть актуальна, если какой-то блок должен переноситься вниз на новую строку, а не обтекать плавающий элемент.

Например, футер, как правило, должен находиться строго внизу и растягиваться по всей ширине страницы. Если же перед футером находится плавающий элемент, то футер может обтекать этот элемент, что не желательно.

Для запрета обтекания элементов в CSS применяется свойство clear, которое указывает браузеру, что к стилизуемому элементу не должно применяться обтекание.

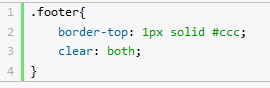
Свойство clear может принимать следующие значения:

* left: стилизуемый элемент может обтекать плавающий элемент справа. Слева же обтекание не работает
* right: стилизуемый элемент может обтекать плавающий элемент только слева. А справа обтекание не работает
* both: стилизуемый элемент может обтекать плавающие элементы и относительно них смещается вниз
* none: стилизуемый элемент ведет себя стандартным образом, то есть принимает участие в обтекании справа и слева

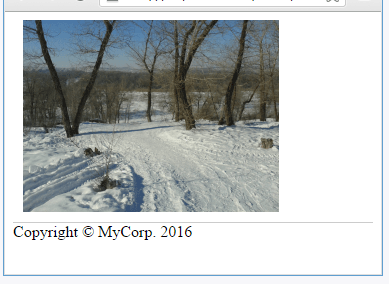
Например, пусть на веб-странице будет определен футер:

|  |  |
| --- | --- |
|  | Наличие обтекания будет создавать некорректное отображение, при котором футер смещается вверх: |

Изменим стиль футера:



Теперь футер не будет обтекать изображение, а будет уходить вниз.



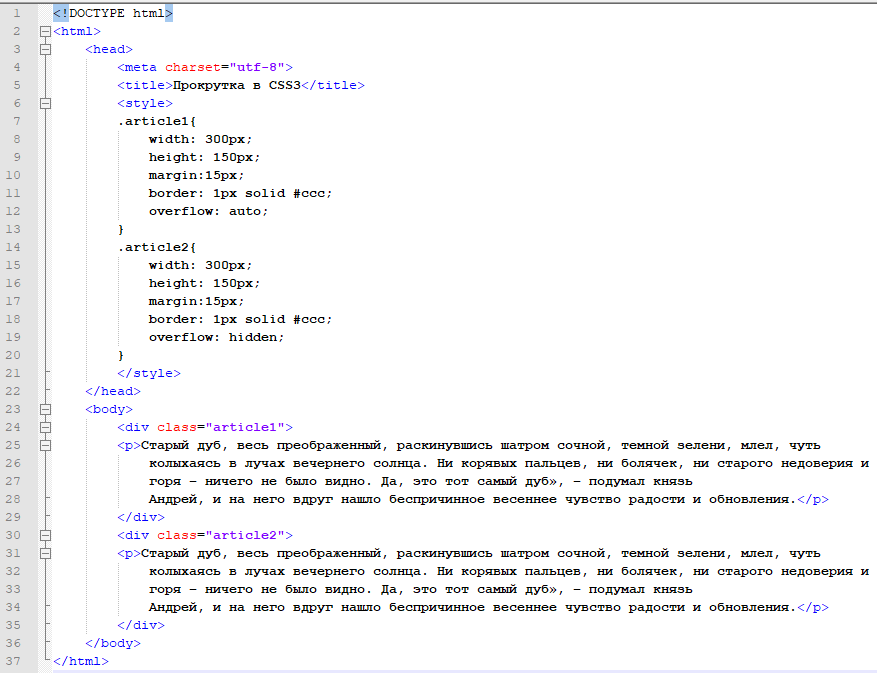
## Прокрутка элементов

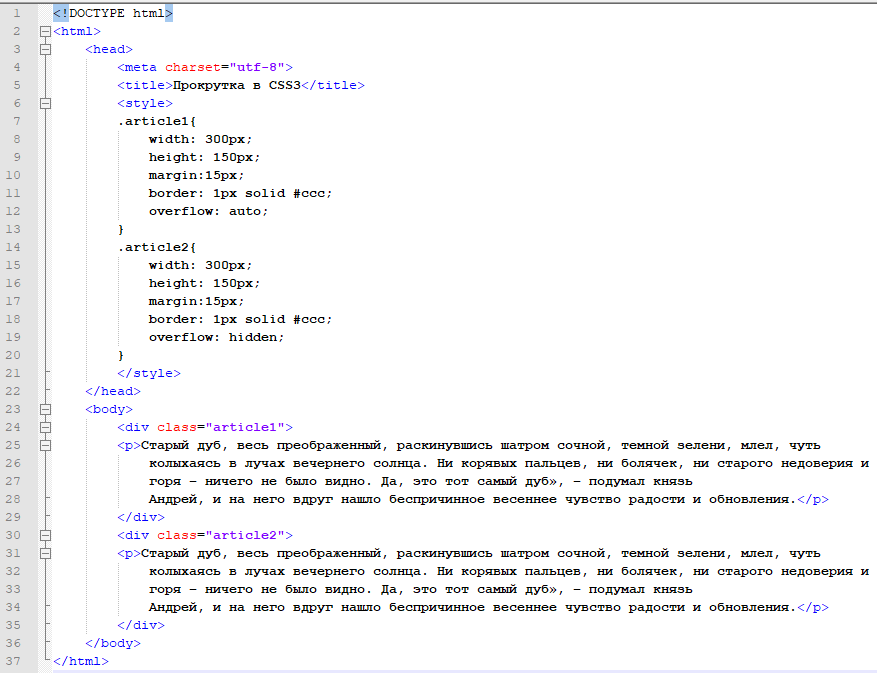
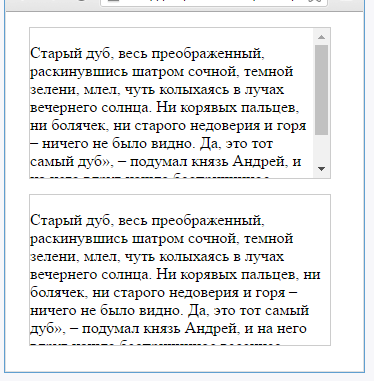
Нередко при создании веб-страниц можно столкнуться с ситуацией, когда содержимое блока занимает гораздо больше места, чем сам определено шириной и высотой блока. В этой ситуации по умолчанию браузер все равно отображает содержимое, даже если оно выходит за границы блока.

Однако свойство overflow позволяет настроить поведение блока в подобной ситуации и добавить возможность прокрутки. Это свойство может принимать следующие значения:

* auto: если контент выходит за границы блока, то создается прокрутка. В остальных случаях полосы прокрутки не отображаются
* hidden: отображается только видимая часть контента. Контент, который выходит за границы блока, не отображается, а полосы прокрутки не создаются
* scroll: в блоке отображаются полосы прокрутки, даже если контент весь помещается в границах блока, и таких полос прокрутки не требуется
* visible: значение по умолчанию, контент отображается, даже если он выходит за границы блока

Рассмотрим применение двух значений:





Свойство **overflow** управляет полосами прокрутки как по вертикали, так и по горизонтали. С помощью дополнительных свойств **overflow-x** и **overflow-y** можно определить прокрутку соответственно по горизонтали и по вертикали. Данные свойства принимают те же значения, что и overflow:



## Линейный градиент

Градиенты представляют плавный переход от одного цвета к другому. В CSS3 имеется ряд встроенных градиентов, которые можно использовать для создания фона элемента.

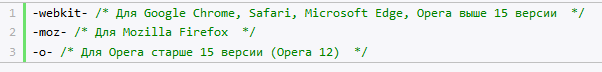
Градиенты в CSS не представляют какого-то специального свойства. Они лишь создают значение, которое присваивается свойству **background-image**.

Линейный градиент распространяется по прямой от одного конца элемента к другому, осуществляя плавный переход от одного цвета к другому. Для создания градиента нужно указать его начало и несколько цветов, например:



В данном случае началом градиента будет левый край элемента, который обозначается значением left. Цвета градиента: черный (black) и белый (white). То есть начиная с левого края элемента до правого будет плавно идти переход из черного цвета в белый.

В использовании градиентов есть один недостаток – многообразие браузеров вынуждает использовать префикс вендора:



Так, полноценное использование градиента будет выглядеть следующим образом:

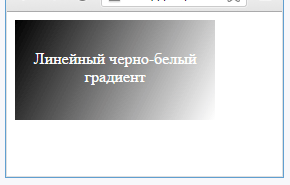
|  |  |
| --- | --- |
|  |  |

Для установки начала градиента можно использовать следующие значения: left (слева направо), right (справа налево), top (сверху вниз) или bottom (снизу вверх). Например, вертикальный градиент будет выглядеть следующим образом:



Также можно задать диагональное направление с помощью двух значений:





Кроме конкретных значений типа top или left также можно указать угол от 0 до 360, который определит направление градиента:



После величины углы указывается слово deg.

К примеру, 0deg означает, что градиент начинается в левой части и перемещается в правую часть, а при указании 45deg он начинается в нижнем левом углу и перемещается под углом 45° в верхний правый угол.

После определения начала градиента, можно указать применяемых цветов или опрные точки. Цветов не обязательно должно быть два, их может быть и больше:

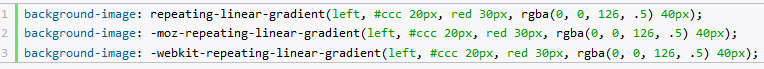


Все применяемые цвета распределяются равномерно. Однако можно также указать конкретные места фона для цветовых точек. Для этого после цвета добавляется второе значение, которое и определяет положение точки.



### Повторяющийся градиент

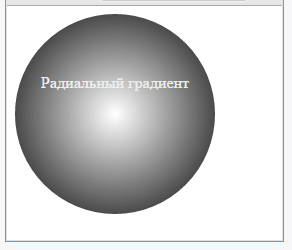
С помощью repeating-linear-gradient можно создавать повторяющиеся линейные градиенты. Например:



В данном случае градиент начинается с левого края элемента с полоски серого цвета (#ccc) шириной 20 пикселей, далее до 30 пикселей идет переход к красному цвету, а затем до 40 пикселей выполняется обратный переход к светло-синему цвету (rgba(0, 0, 126, .5)). После этого браузер повторяет градиент, пока не заполнит всю поверхность элемента.

## Радиальный градиент

Радиальные градиенты в отличие от линейных распространяются от центра наружу по круговой схеме. Для создания радиального градиента достаточно указать цвет, который будет в центре градиента, и цвет, который должен быть снаружи. Эти цвета передаются в функцию radial-gradient(). Например:



Как и в случае с линейным градиентом здесь также надо использовать префиксы вендоров для поддержки браузерами.

Радиальный градиент может иметь две формы: круговую и эллиптическую. Эллиптическая форма представляет распространение градиента в виде эллипса и задается с помощью ключевого слова ellipse:



Поскольку это значение для градиента по умолчанию, то оно может опускаться при использовании.

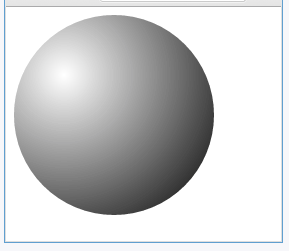
Круговая форма представляет распространение градиента в виде кругов от центра во вне. Для этого используется ключевое слово circle:



Как правило, центр радиального градиента расположен в центре элемента, но это поведение можно переопределить, указав значение для параметра background-position:



Числа 25% 30% означают, что центр градиента будет находиться на расстоянии в 25% от левой границы и в 30% от верхней границы элемента.



С помощью специальных дополнительных значений можно задать размер градиента:

* closest-side: градиент распространяется из центра только до ближайшей к центру стороне элемента. То есть градиент остается внутри элемента
* closest-corner: ширина градиента вычисляется по расстоянию из его центра до ближайшего угла элемента, поэтому градиент может выйти за пределы элемента.
* farthest-side: градиент распространяется из центра до самой дальней стороны элемента
* farthest-corner: ширина градиента вычисляется по расстоянию из его центра до самого дальнего угла элемента

