**Задача**

Добавить изображение на страницу так, чтобы оно обтекалось рядом лежащим текстом.

**Решение**

Обтекание картинки текстом обычно используется для компактного размещения материала и связывания между собой иллюстраций и текста. Само обтекание создаётся с помощью стилевого свойства float, добавляемого к селектору **IMG**. Значение left выравнивает изображение по левому краю, right — по правому. Обтекание при этом происходит по другим, свободным сторонам.

**Пример 1. Обтекание картинок**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Обтекание**</title>**

**<style>**

.leftimg {

float:left; /\* Выравнивание по левому краю \*/

margin: 7px 7px 7px 0; /\* Отступы вокруг картинки \*/

}

.rightimg {

float: right; /\* Выравнивание по правому краю \*/

margin: 7px 0 7px 7px; /\* Отступы вокруг картинки \*/

}

**</style>**

**</head>**

**<body>**

**<h2>**Доклад лейтенанта Бокатуева**</h2>**

**<p><img** src="images/1.jpg" alt="Лейтенант Бокатуев" width="132" height="174" class="leftimg"**>**

Вчера во время проведения разведоперации наша группа подверглась нападению неизвестного

противника в камуфляжной форме Алиенов. В результате эффективной обороны и стремительной

контратаки многочисленная группа боевиков была смята и отброшена. Среди личного состава

потерь нет. Бойцы разведгруппы проявили недюжие навыки владения оружием. Особо отличился

в бою взводный Кудряшев&nbsp;М.А., грамотно использовавший человеческие ресурсы

своего взвода. В результате операции были захвачены элементы внеземной культуры, которые

переданы аналитической группе.**</p>**

**<h2>**Пресс-релиз аналитической группы**</h2>**

**<p><img** src="images/2.jpg" alt="Учёные, находящиеся в состоянии аффекта" class="rightimg"**>**

В наших секретных лабораториях в рамках проекта &laquo;Пандора&raquo; разрабатывалось

психотропное оружие. В результате неудачного эксперимента большинство ученых, работавших

над прибором, подверглись воздействию психотропного излучения, и они, находясь в состоянии

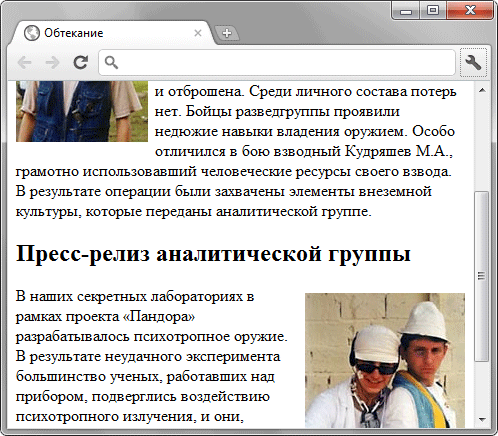
аффекта, растащили прототип по деталям. Возможно, наши ученые до сих пор находятся в

состоянии аффекта.**</p>**

**</body>**

**</html>**

Результат данного примера показан на рис. 1.



*Рис. 1. Текст с иллюстрациями*

Использование свойства float заставляет текст плотно прилегать к изображению. Поэтому в примере введёно универсальное свойство margin, которое добавляет отступы между картинкой и текстом. Это свойство одновременно задаёт отступ сверху, справа, снизу и слева от фотографий.

**Задача**

Растянуть фоновую картинку на всю ширину окна браузера с помощью CSS3.

**Решение**

Для масштабирования фона предназначено свойство background-size, в качестве его значения указывается 100%, тогда фон будет занимать всю ширину окна браузера. Для старых версий браузеров следует использовать специфические свойства с префиксами, как показано в примере 1.

**Пример 1. Растягиваемый фон**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Растягиваемый фон**</title>**

**<style>**

**body** {

background: **url(**images/sun2.png**)** no-repeat;

-moz-background-size: 100%; /\* Firefox 3.6+ \*/

-webkit-background-size: 100%; /\* Safari 3.1+ и Chrome 4.0+ \*/

-o-background-size: 100%; /\* Opera 9.6+ \*/

background-size: 100%; /\* Современные браузеры \*/

}

**</style>**

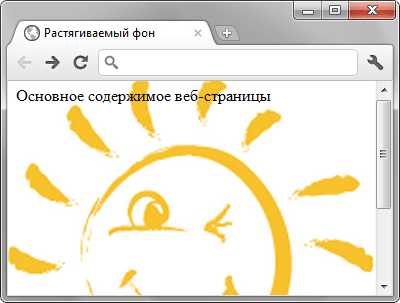
**</head>**

**<body>**

**</body>**

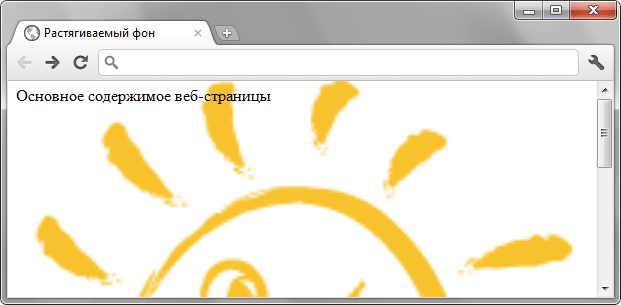
**</html>**

Результат данного примера показан на рис. 1.



*Рис. 1. Вид фона при уменьшенном размере окна*

При увеличении размера окна браузера фон также начнёт расширяться, это приведет к ухудшению вида картинки (рис. 2).



*Рис. 2. Вид фона при увеличенном размере окна*

**Задача**

Вывести маркированный список в одну строку и убрать отображение маркеров.

**Решение**

Элементы списка по умолчанию выстраиваются вертикально друг под другом, но иногда требуется разместить их по горизонтали, например, для создания различных меню. В таком случае к селектору **LI**следует добавить стилевое свойство display со значением inline, оно преобразует блочный элемент в строчный (пример 1).

**Пример 1. Горизонтальный список**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Горизонтальный список**</title>**

**<style>**

**ul**.hr {

margin: 0; /\* Обнуляем значение отступов \*/

padding: 4px; /\* Значение полей \*/

}

**ul**.hr **li** {

display: inline; /\* Отображать как строчный элемент \*/

margin-right: 5px; /\* Отступ слева \*/

border: 1px solid #000; /\* Рамка вокруг текста \*/

padding: 3px; /\* Поля вокруг текста \*/

}

**</style>**

**</head>**

**<body>**

**<ul** class="hr"**>**

**<li>**Мчади**</li>**

**<li>**Када на мацони**</li>**

**<li>**Пахлава**</li>**

**<li>**Кчуч**</li>**

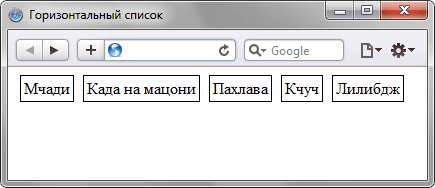
**<li>**Лилибдж**</li>**

**</ul>**

**</body>**

**</html>**

Результат данного примера показан на рис. 1.



*Рис. 1. Горизонтальный список*

Из-за того, что вокруг списка автоматически добавляются отступы, их следует изменить, добавляя свойства margin и padding к селектору **UL**, как показано в примере.

### Задача

Заменить одну картинку на другую при наведении на неё курсора мыши.

### Решение

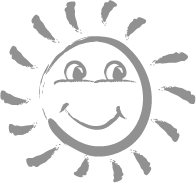
Смена одной картинки на другую при наведении на неё курсора мыши называется эффектом перекатывания. Обычно для создания эффекта перекатывания применяется JavaScript, но в большинстве случаев вполне достаточно и CSS.

#### Псевдокласс :hover

Вначале разберём, как делается собственно сам эффект. Для этой цели используется псевдокласс :hover, он добавляется к желаемому селектору. Этот псевдокласс определяет стиль элемента, когда на него наводится курсор мыши, что нам как раз и требуется. Сразу отметим, что в браузере Internet Explorer версии 6 и ниже :hover работает только со ссылками, тогда как другие браузеры понимают :hover и для других элементов. Так что для универсальности придётся помещать изображение внутрь контейнера **<a>**. Сам рисунок добавляется и меняется с помощью стилевого свойства background. Алгоритм действий следующий.

1. Готовятся два изображения одинакового размера, одно для исходного вида, а второе — для его замены при эффекте перекатывания.
2. Исходная картинка добавляется в качестве фонового рисунка к селектору **A** через стилевое свойство background со значением url(URI), здесь URI — путь к графическому файлу.
3. Присоединяем псевдокласс к селектору **A** и снова включаем свойство background, но в качестве значения указываем замещающее изображение.
4. Чтобы ссылка была по размеру картинки, её следует превратить в блочный элемент с помощью свойства display со значением block, а также задать высоту и ширину ссылки.

На рис. 1 приведены два изображения: исходное (рис. 1а) и замещающее (рис. 1б).

а б

*Рис. 1. Картинки для создания эффекта перекатывания*

Остаётся собрать весь код воедино, как показано в примере 1. Чтобы ссылка не вела на какой-то определенный файл, что в данном случае излишне, добавим «заглушку» в виде символа решетки (#) к атрибуту href тега **<a>**.

**Пример 1. Использование псевдокласса :hover**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Эффект перекатывания**</title>**

**<style>**

**a**.rollover {

background: **url(**images/sun1.png**)**; /\* Путь к файлу с исходным рисунком \*/

display: block; /\* Рисунок как блочный элемент \*/

width: 196px; /\* Ширина рисунка \*/

height: 183px; /\* Высота рисунка \*/

}

**a**.rollover:hover {

background: **url(**images/sun2.png**)**; /\* Путь к файлу с заменяемым рисунком \*/

}

**</style>**

**</head>**

**<body>**

**<p><a** href="#" class="rollover"**>** **</a></p>**

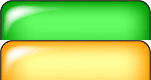
**</body>**

**</html>**

Приведённый способ хотя и прост в реализации, но имеет определённый недостаток. Поскольку второй рисунок загружается только при наведении курсора на ссылку, то отображение картинки происходит с некоторой паузой. Для устранения этой задержки можно использовать не два изображения, а лишь одно.

#### Эффект перекатывания с одним рисунком

Казалось бы имеется противоречие, ведь эффект перекатывания состоит в замене одного рисунка на другой, как же тогда обойтись одной картинкой? На самом деле изображений будет два (рис. 2), но храниться они будут в одном графическом файле.



*Рис. 2. Изображение с двумя картинками*

Смена одного рисунка на другой происходит сдвигом изображения по вертикали за счёт использования универсального свойства background или background-position, как показано в примере 2.

**Пример 2. Изменение положения рисунка**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Эффект перекатывания**</title>**

**<style>**

**a**.rollover {

background: **url(**images/mark.png**)**; /\* Путь к файлу с исходным рисунком \*/

display: block; /\* Рисунок как блочный элемент \*/

width: 151px; /\* Ширина рисунка в пикселах \*/

height: 40px; /\* Высота рисунка \*/

}

**a**.rollover:hover {

background-position: 0 -40px; /\* Смещение фона \*/

}

**</style>**

**</head>**

**<body>**

**<p><a** href="link.html" class="rollover"**></a></p>**

**</body>**

**</html>**

Для селектора **A** устанавливается фоновое изображение через свойство background, ширина (width) совпадает с рисунком, а высота (свойство height) равна половине высоты изображения. В итоге первоначально должна отображаться зеленая вкладка целиком.

При наведении на ссылку фон смещается вверх на половину высоты картинки (40 пикселов) с помощьюbackground-position и становится видна оранжевая вкладка.

**Задача**

Выровнять фотографию с подписью по центру горизонтали веб-страницы.

**Решение**

Иллюстрации к тексту часто выравнивают по центру веб-страницы, при этом текст располагается до и после изображения. Такое расположение элементов позволяет разбить большой текст на смысловые блоки и привлечь внимание к рисункам.

Вначале рассмотрим выравнивание изображения по центру. Для этого к селектору **P** следует добавить стилевое свойство text-align со значением center. При этом тег **<img>** должен располагаться внутри абзаца (тег **<p>**). Чтобы не получилось так, что все абзацы на странице начали выравниваться по центру, введем свой класс fig, и все действия будем производить с ним. В примере 1 показано, как это сделать.

**Пример 1. Использование text-align**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Выравнивание фотографии по центру**</title>**

**<style>**

.fig {

text-align: center; /\* Выравнивание по центру \*/

}

**</style>**

**</head>**

**<body>**

**<p>**Текст до изображения**</p>**

**<p** class="fig"**><img** src="images/photo1.jpg"

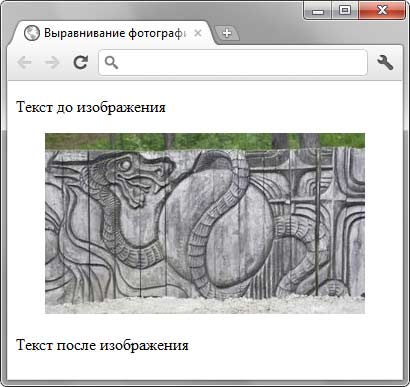
 width="320" height="181" alt="Фотография"**></p>**

**<p>**Текст после изображения**</p>**

**</body>**

**</html>**

Результат данного примера продемонстрирован на рис. 1.



*Рис. 1. Изображение, выровненное по центру веб-страницы*

Также к фотографии можно добавить подрисуночную подпись. Текст должен располагаться сразу же после изображения и аналогично ему выравниваться по центру. Здесь всё просто, опять воспользуемся нашим классом, но уже применим его к тегу **<figure>**. Чтобы текст подписи отличался по своему виду от обычных абзацев, сделаем его курсивным и выделим другим цветом (пример 2).

**Пример 2. Подрисуночная подпись**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Фотография с подписью**</title>**

**<style>**

.fig {

display: block; /\* Блочный элемент (для старых браузеров) \*/

text-align: center; /\* Выравнивание по центру \*/

font-style: italic; /\* Курсивное начертание \*/

margin-top: 0; /\* Отступ сверху \*/

margin-bottom: 5px; /\* Отступ снизу \*/

color: #666; /\* Цвет подрисуночной подписи \*/

}

**</style>**

<!--[if lt IE 9]>

<script>

document.createElement('figure');

document.createElement('figcaption');

</script>

<![endif]-->

**</head>**

**<body>**

**<figure** class="fig"**>**

**<img** src="images/photo2.jpg" width="320" height="179"

 alt="Барельеф"**>**

**<figcaption>**

Пещерный человек добывает огонь.

Но что это за лунная поверхность на заднем плане? Нет, не так все просто.

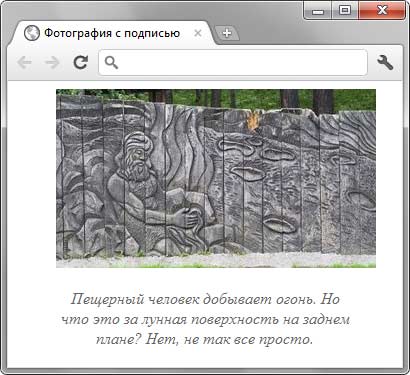
**</figcaption>**

**</figure>**

**</body>**

**</html>**

Результат данного примера показан на рис. 2.



*Рис. 2. Подпись под рисунком*

Для блока с фотографией используется тег **<figure>**, а для подписи к нему тег **<figcaption>**. Старые версии IE не понимают эти теги, поэтому специально для них добавляется небольшой скрипт.

# Как изменить расстояние между строками текста?

﻿

Для управления межстрочным расстоянием (оно еще называется интерлиньяжем) применяется стилевое свойство line-height. В качестве значения обычно указывается число, которое определяет межстрочный интервал. Например, line-height: 2 устанавливает двойной интервал, а line-height: 1.5 — полуторный (пример 1).

**Пример 1. Интерлиньяж в тексте**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Интерлиньяж**</title>**

**<style>**

**p**.dline {

line-height: 1.5;

}

**P** {

line-height: 0.9em;

}

**</style>**

**</head>**

**<body>**

**<p** class="dline"**>**Некоторые сайты сохраняют техническую информацию на

локальном компьютере посетителя в виде небольших текстовых файлов, которые

называются сookie.**</p>**

**<p>**Данный термин переводится как «печенье», но поскольку это название лишь

сбивает с толку, его оставляют без перевода. Cookie можно использовать для

сохранения даты последнего посещения читателя, паролей, а также любой

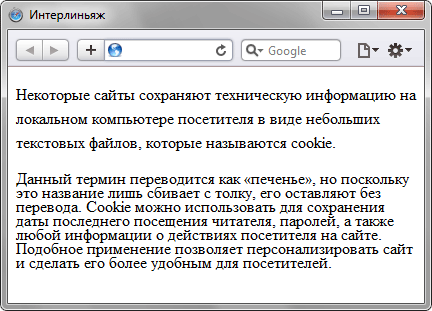
информации о действиях посетителя на сайте. Подобное применение позволяет

персонализировать сайт и сделать его более удобным для посетителей.**</p>**

**</body>**

**</html>**

Результат данного примера показан на рис. 1.



*Рис. 1. Вид текста при разном межстрочном расстоянии*

Значение line-height допустимо задавать и в других единицах CSS, например, пикселах (px), процентах (%) и др. В данном примере для первого абзаца установлен полуторный интерлиньяж, а для второго — межстрочное расстояние равно 0.9em.

**Задача**

Изменить цвет рамки вокруг изображений-ссылок.

**Решение**

Рамка вокруг изображения появляется в том случае, когда к тегу **<img>** добавляется атрибут border с ненулевым значением и при создании изображения-ссылки. При этом тег **<img>** хранится внутри контейнера **<a>**. Независимо от ситуации цвет рамки устанавливается автоматически, однако через стили можно задать любой желаемый цвет границы.

По умолчанию цвет рамки вокруг изображений-ссылок совпадает с цветом текстовых ссылок. В стилях достаточно использовать свойство border, добавляя его к конструкции **a img**, которая сообщает, что рамку следует отображать только у изображений, находящихся внутри контейнера **<a>** (пример 1).

**Пример 1. Цвет рамки**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Изображение-ссылка**</title>**

**<style>**

**a** **img** {

border: 3px solid #c79316; /\* Рамка вокруг ссылок \*/

}

**a**:visited **img** {

border: 3px solid #c716bd; /\* Рамка вокруг посещенных ссылок \*/

}

**</style>**

**</head>**

**<body>**

**<p>**

**<a** href="photo1.html"**><img** src="images/thumb1.jpg" alt="Фотография 1"**></a>**

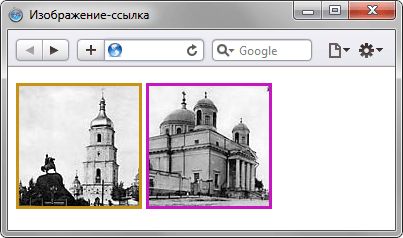
**<a** href="photo2.html"**><img** src="images/thumb2.jpg" alt="Фотография 2"**></a>**

**</p>**

**</body>**

**</html>**

Результат данного примера показан на рис. 1. Чтобы цвет рамки менялся при посещении ссылки, в примере используется псевдокласс :visited.



*Рис. 1. Цвет рамки вокруг непосещенной и посещенной ссылки*

**Задача**

Изменить параметры рамки вокруг изображения при наведении на него курсора мыши и восстановить обратно, когда курсор уходит прочь.

**Решение**

За изменение стиля элемента при наведении на него указателя мыши отвечает псевдокласс :hover. Для управления стилем ссылок этот псевдокласс добавляется к селектору **A**.

a { стиль обычной ссылки }  
a:hover { стиль ссылки при наведении на нее курсора }

Также псевдокласс :hover допускается применять аналогичным способом к изображениям, но в браузере Internet Explorer версии 6.0 и ниже этот подход работать не будет. Все потому, что эта версия Internet Explorer понимает :hover только для ссылок. Поэтому для начала рассмотрим универсальный метод, работающий во всех браузерах.

Изображение превращается в ссылку, если тег **<img>** поместить внутрь контейнера **<a>**, при этом вокруг картинок автоматически добавляется рамка, показывающая что перед нами ссылка. Вид таких рамок управляется с помощью стилевого свойства border добавляемого к конструкции **a img**, которая сообщает, что стили следует применять к тегу **<img>** расположенному внутри **<a>**. Параметры рамки при наведении на рисунок курсора задаются конструкцией **a:hover img**, как показано в примере 1.

**Пример 1. Изменение цвета рамки**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Цвет рамки**</title>**

**<style>**

**a** **img** {

border: 3px solid #53da3f; /\* Рамка вокруг изображения-ссылки \*/

}

**a**:hover **img** {

border: 3px solid #f26522; /\* Рамка при наведении на ссылку курсора мыши \*/

}

**</style>**

**</head>**

**<body>**

**<p>**

**<a** href="vinnie-the-pooh.html"**>**

**<img** src="images/figure.jpg" alt="Винни-Пух" width="100" height="111"**></a>**

**<a** href="vinnie-the-pooh.html"**>**

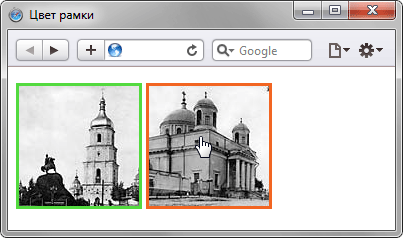
**<img** src="images/figure.jpg" alt="Винни-Пух" width="100" height="111"**></a>**

**</p>**

**</body>**

**</html>**

Результат данного примера показан на рис. 1.



*Рис. 1. Вид рамки до и после наведения курсора на изображение*

Если хочется сделать изменение цвета рамки без применения ссылок, то свойство border допустимо применить непосредственно к селектору **img**, при этом в браузере Internet Explorer до версии 6.0 код работать не будет (пример 2).

**Пример 2. Изменение параметра рамки вокруг изображения**

**img** {

border: 3px solid #53da3f; /\* Рамка вокруг изображения \*/

}

**img**:hover {

border: 3px solid #f26522; /\* Рамка при наведении на рисунок курсора мыши \*/

}

В данном примере рамка добавляется ко всем изображениям веб-страницы, если с помощью рамки требуется выделить только некоторые из них, добавьте к ним стилевые классы.

**Задача**

Добавить на страницу несколько изображений, чтобы они располагались рядом с друг другом по горизонтали.

**Решение**

Изображения являются строчными элементами, поэтому написание в коде подряд несколько тегов **<img>**автоматически выстроит картинки по горизонтали. Тем не менее, с помощью стилей этим построением можно управлять более изящно.

Для начала рассмотрим простой пример, когда изображения выстраиваются по горизонтали без стилевого оформления (пример 1).

**Пример 1. Картинки по горизонтали**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Фотографии**</title>**

**</head>**

**<body>**

**<p>**

**<img** src="images/thumb1.jpg" alt="Фотография 1" width="120" height="120"**>**

**<img** src="images/thumb2.jpg" alt="Фотография 2" width="120" height="120"**>**

**<img** src="images/thumb3.jpg" alt="Фотография 3" width="120" height="120"**>**

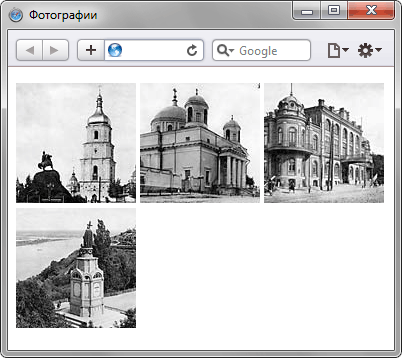
**<img** src="images/thumb4.jpg" alt="Фотография 4" width="120" height="120"**>**

**</p>**

**</body>**

**</html>**

Результат данного примера показан на рис. 1. Изображения выстраиваются в одну строку по горизонтали, а если некоторые из них не помещаются по ширине в окне, то переносятся на следующую строку.



*Рис. 1. Фотографии, расположенные по горизонтали*

Обратите внимание на пробел между фотографиями, он появляется за счет переноса тега **<img>** в коде на новую строку. Чтобы убрать пустой промежуток достаточно написать теги **<img>** в примере в одну строку.

Для регулирования горизонтальных и вертикальных отступов между фотографиями применяется стилевое свойство margin-right и margin-bottom. В примере 2 кроме отступов к фотографиям также добавлена рамка и цветная область вокруг изображений.

**Пример 2. Отступы у фотографий**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Фотографии**</title>**

**<style>**

.thumb **img** {

border: 2px solid #55c5e9; /\* Рамка вокруг фотографии \*/

padding: 15px; /\* Расстояние от картинки до рамки \*/

background: #666; /\* Цвет фона \*/

margin-right: 10px; /\* Отступ справа \*/

margin-bottom: 10px; /\* Отступ снизу \*/

}

**</style>**

**</head>**

**<body>**

**<p** class="thumb"**>**

**<img** src="images/thumb1.jpg" alt="Фотография 1" width="120" height="120"**>**

**<img** src="images/thumb2.jpg" alt="Фотография 2" width="120" height="120"**>**

**<img** src="images/thumb3.jpg" alt="Фотография 3" width="120" height="120"**>**

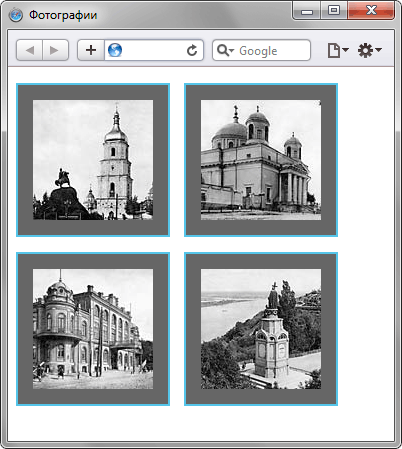
**<img** src="images/thumb4.jpg" alt="Фотография 4" width="120" height="120"**>**

**</p>**

**</body>**

**</html>**

Результат данного примера показан на рис. 2.



*Рис. 2. Вид фотографий, оформленных с помощью стилей*

**Задача**

Отразить изображение по горизонтали или вертикали.

**Решение**

Для отражения изображения или другого элемента, в том числе текстового, применяется свойствоtransform и функция scale() с отрицательным значением. Варианты такие:

* transform: scale(-1, 1) — отражение по горизонтали;
* transform: scale(1, -1) — отражение по вертикали;
* transform: scale(-1, -1) — одновременное отражение по горизонтали и вертикали.

Есть также отдельные функции scaleX() и scaleY(), они отвечают за масштабирование по соответствующим координатам.

Хотя функция scale() предназначена для изменения масштаба элемента, отрицательное значение также позволяет сделать отражение. В примере 1 добавляется обычное изображение, а затем оно же, но отражённое по вертикали. Для этого к **<img>** добавляется класс с именем mirrorY.

**Пример 1. Отражение фотографии**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Отражение**</title>**

**<style>**

.mirrorY { transform: **scale(**1, -1**)**; }

**</style>**

**</head>**

**<body>**

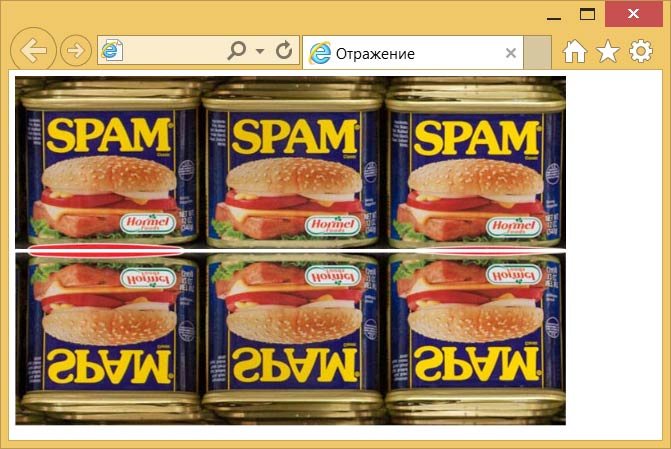
**<img** src="images/spam.jpg" alt="Спам"**>**

**<img** src="images/spam.jpg" alt="Спам" class="mirrorY"**>**

**</body>**

**</html>**

Результат данного примера показан на рис. 1.



*Рис. 1. Обычная фотография и фотография, отражённая по вертикали*

**Задача**

Изменить значение прозрачности изображения.

**Решение**

За управление прозрачностью элемента на странице отвечает стилевое свойство opacity, которое относится к CSS3. В качестве значения применяются дробные числа от 0 до 1, где ноль соответствует полной прозрачности, а единица, наоборот, непрозрачности объекта.

Старые версии Internet Explorer не поддерживает opacity, поэтому для этого браузера следует использовать специфическое свойство filter со значением alpha(Opacity=X), где под X подразумевается целое число от 0 до 100. Это значение определяет уровень прозрачности: 0 — полная прозрачность; 100 — наоборот, непрозрачность объекта.

Соединяя воедино два свойства, получим универсальный код, который устанавливает заданную прозрачность для изображений (пример 1).

**Пример 1. Полупрозрачное изображение**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Прозрачность**</title>**

**<style>**

.transparent75 {

filter: **alpha(**Opacity=75**)**; /\* Полупрозрачность для IE \*/

opacity: 0.75; /\* Полупрозрачность для других браузеров \*/

}

.transparent50 {

filter: **alpha(**Opacity=50**)**;

opacity: 0.5;

}

.transparent25 {

filter: **alpha(**Opacity=25**)**;

opacity: 0.25;

}

**</style>**

**</head>**

**<body>**

**<p>**

**<img** src="images/cat.jpg" alt="Оригинальное изображение"

 width="250" height="243"**>**

**<img** src="images/cat.jpg" alt="Непрозрачность 75%"

 width="250" height="243" class="transparent75"**>**

**<img** src="images/cat.jpg" alt="Непрозрачность 50%"

 width="250" height="243" class="transparent50"**>**

**<img** src="images/cat.jpg" alt="Непрозрачность 25%"

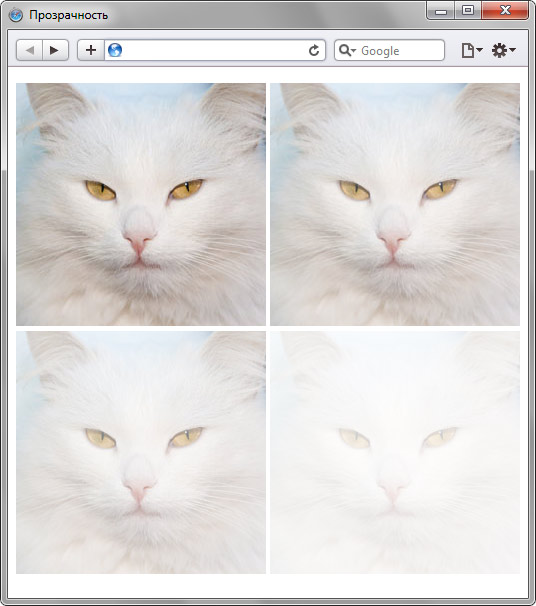
 width="250" height="243" class="transparent25"**>**

**</p>**

**</body>**

**</html>**

Результат примера показан на рис. 1.



*Рис. 1. Фотография с различными значениями прозрачности*

В данном примере вначале приводится исходная фотография, к которой не применяются настройки прозрачности, последующие фотографии отображаются с уровнем непрозрачности 75%, 50% и 25%.

**Задача**

Сделать горизонтальное меню, пункты которого имеют произвольный наклон (рис. 1).

Вид меню с наклонными пунктами

*Рис. 1. Вид меню с наклонными пунктами*

**Решение**

За трансформацию элемента отвечает стилевое свойство transform, в качестве его значения используется функция skewX, которая и задаёт нужный наклон. Угол наклона проще устанавливать в градусах, к примеру, 30deg — это 30º от вертикали. Трансформация действует и на все дочерние элементы, поэтому текст внутри элемента также будет наклонён, что не лучшим образом отражается на его читабельности и красоте. Поэтому к тексту важно ещё раз применить наклон, но уже в другую сторону, для этого достаточно просто поменять знак перед значением градусов.

Браузеры поддерживают свойство transform преимущественно со своими префиксами, так что для универсальности в стилях следует повторить его несколько раз с одним и тем же значением, добавляя -moz- для Firefox, -webkit- для Safari и Chrome, -o- для Opera и -ms- для браузера Internet Explorer как показано в примере 1.

**Пример 1. Наклон пункта меню**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Меню**</title>**

**<style>**

**li** {

display: inline-block; /\* Строчно-блочные элементы \*/

background: #CA181A; /\* Цвет фона \*/

margin-right: 3px; /\* Расстояние между пунктами меню \*/

-webkit-transform: **skewX(**-30deg**)**; /\* Для Safari и Chrome \*/

-moz-transform: **skewX(**-30deg**)**; /\* Для Firefox \*/

-o-transform: **skewX(**-30deg**)**; /\* Для Opera \*/

-ms-transform: **skewX(**-30deg**)**; /\* Для IE \*/

transform: **skewX(**-30deg**)**; /\* CSS3 \*/

}

**a** {

color: #fff; /\* Цвет ссылок \*/

display: block; /\* Блочный элемент \*/

padding: 5px 15px; /\* Поля вокруг текста \*/

text-decoration: none; /\* Убираем подчёркивание \*/

-webkit-transform: **skewX(**30deg**)**; /\* Для Safari и Chrome \*/

-moz-transform: **skewX(**30deg**)**; /\* Для Firefox \*/

-o-transform: **skewX(**30deg**)**; /\* Для Opera \*/

-ms-transform: **skewX(**30deg**)**; /\* Для IE \*/

transform: **skewX(**30deg**)**; /\* CSS3 \*/

}

**li**:hover {

background: #333; /\* Цвет фона при наведении курсора мыши \*/

}

**</style>**

**</head>**

**<body>**

**<ul>**

**<li><a** href="1.html"**>**Джокер**</a></li>**

**<li><a** href="2.html"**>**Пазузу**</a></li>**

**<li><a** href="3.html"**>**Палпатин**</a></li>**

**<li><a** href="4.html"**>**Доктор Дум**</a></li>**

**</ul>**

**</body>**

**</html>**

В данном примере горизонтальное меню создаётся с помощью маркированного списка. Чтобы список был именно горизонтальным, в стилях к элементу **li** добавляется свойство display со значением inline-block. Для наклона применяется свойство transform с функцией skewX и значением -30º. К ссылкам внутри списка также применяется это свойство, но уже с положительным значением 30º, это обеспечивает правильное, а не наклонное отображение букв.

Браузеры Chrome, Safari и Firefox используют свойство transform для блочных или строчно-блочных элементов, поэтому для ссылок задано свойство display со значением block.

**Задача**

Задать цвет маркеров в списке не изменяя цвет текста.

**Решение**

Существует два способа изменения цвета маркеров, условно назовем их простой и хитрый. Простой метод состоит в том, что внутрь **<li>** вкладываем тег **<span>**, а уже внутрь него помещаем текст. Иными словами вместо традиционной схемы <li>текст</li> создаем конструкцию <li><span>текст</span></li>. При этом цвет маркеров определяется стилевым свойством color для селектора **LI**, а цвет текста — для селектора **SPAN** (пример 1).

**Пример 1. Использование вложенных тегов**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Цвет текста и маркеров в списке**</title>**

**<style>**

**li** {

color: red; /\* Цвет маркеров \*/

}

**li** **span** {

color: navy; /\* Цвет текста \*/

}

**</style>**

**</head>**

**<body>**

**<ul>**

**<li><span>**Скрипка**</span></li>**

**<li><span>**Гитара**</span></li>**

**<li><span>**Волынка**</span></li>**

**<li><span>**Шарманка**</span></li>**

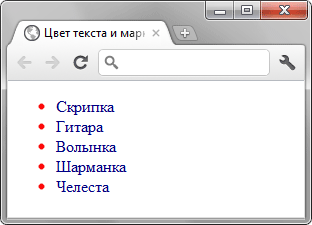
**<li><span>**Челеста**</span></li>**

**</ul>**

**</body>**

**</html>**

Результат данного примера показан ниже (рис. 1).



*Рис. 1. Маркеры, отличающиеся по цвету от основного текста*

Несмотря на простоту, метод неудобен, особенно при объёмных списках, ведь к каждому пункту списка теперь придётся добавлять тег **<span>**. Поэтому разберём хитрый способ, полностью основанный на работе CSS.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Internet Explorer | Chrome | Opera | Safari | Firefox | Android | iOS |
| 8.0+ | 1.0+ | 7.0+ | 3.1+ | 1.0+ | 1.0+ | 1.0+ |

Смысл в следующем — убираем оригинальные маркеры списка через свойство list-style-type и добавляем свои собственные маркеры с помощью псевдоэлемента :before и свойства content. Такая связка позволяет вставлять любой текст или символ перед элементом, в данном случае **LI**. Причём видом текста (цвет, шрифт, фон и др.) также можно управлять через стили, что и продемонстрировано в примере 2. Здесь в качестве маркеров используется символ параграфа ¶.

**Пример 2. Использование псевдоэлемента :before**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Цвет маркеров в списке**</title>**

**<style>**

**li** {

list-style-type: none; /\* Прячем исходные маркеры \*/

}

**li**:before {

color: red; /\* Цвет маркера \*/

content: "¶ "; /\* Сам маркер \*/

padding-right: 10px; /\* Расстояние от маркера до текста \*/

}

**</style>**

**</head>**

**<body>**

**<ul>**

**<li>**Север**</li>**

**<li>**Юг**</li>**

**<li>**Запад**</li>**

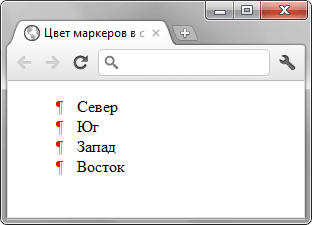
**<li>**Восток**</li>**

**</ul>**

**</body>**

**</html>**

Результат данного примера показан на рис. 2.



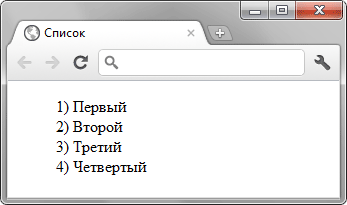
*Рис. 2. Маркеры, созданные с помощью стилей*

**Задача**

Изменить вывод чисел в нумерованном списке и сделать, чтобы после номера шла круглая скобка.

**Решение**

По умолчанию нумерованный список имеет определённый вид: вначале идет число, затем точка и после этого через пробел отображается текст. Такая форма написания наглядна и удобна, но некоторые разработчики предпочитают видеть другой способ оформления нумерации списков. А именно, чтобы вместо точки шла закрывающая скобка, как продемонстрировано на рис. 1 или нечто подобное.



*Рис. 1. Вид нумерованного списка со скобкой*

Стили позволяют поменять вид нумерации списков с помощью стилевых свойств content иcounter-increment. Вначале для селектора **OL** требуется задать counter-reset: item, это нужно для того, чтобы нумерация в каждом новом списке начиналась заново. В противном случае, нумерация будет продолжена и вместо 1,2,3 можно будет увидеть 5,6,7. Значение item это уникальный идентификатор счетчика, мы его выбираем сами. Далее необходимо спрятать исходные маркеры через стилевое свойство list-style-type со значением none.

Свойство content само по себе не используется, как правило, оно работает в сочетании с псевдоэлементами :after и :before. Так, конструкция **li:before** говорит о том, что перед каждым элементом списка необходимо добавить какое-то содержание (пример 1).

**Пример 1. Создание своей нумерации**

**li**:before {

content: **counter(**item**)** ') '; /\* Добавляем к числам скобку \*/

counter-increment: item; /\* Задаём имя счетчика \*/

}

content со значением counter(item) выводит число; добавляя скобку, как показано в данном примере, получим требуемый вид нумерации. Свойство counter-increment необходимо для увеличения номера списка на единицу. Обратите внимание, что везде используется один и тот же идентификатор с именемitem. Окончательный код приведен в примере 2.

**Пример 2. Изменение вида списка**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Список**</title>**

**<style>**

**ol** {

list-style-type: none; /\* Убираем исходные маркеры \*/

counter-reset: item; /\* Обнуляем счетчик списка \*/

}

**li**:before {

content: **counter(**item**)** ') '; /\* Добавляем к числам скобку \*/

counter-increment: item; /\* Задаём имя счетчика \*/

}

**</style>**

<!--[if lt IE 8]>

<style>

ol {

list-style-type: decimal;

}

</style

<![endif]-->

**</head>**

**<body>**

**<ol>**

**<li>**Первый**</li>**

**<li>**Второй**</li>**

**<li>**Третий**</li>**

**<li>**Четвертый**</li>**

**</ol>**

**</body>**

**</html>**

Браузер Internet Explorer до версии 8.0 не поддерживает свойство content. Чтобы не получилось явного различия при выводе списка в разных браузерах, специально для Internet Explorer в данном примере через условные комментарии добавлено свойство list-style-type со значением decimal. Эта запись выводит традиционную нумерацию строк.

Приведённым способом можно делать любой вид нумерованного списка, например, брать число в квадратные скобки, в стилях при этом изменится только одна строка.

content: '[' counter(item) '] ';

**Задача**

Через стили установить цвет, размер, шрифт и другие параметры у чисел в нумерованном списке.

**Решение**

В браузере Firefox определение стилей нумерации в списке происходит с помощью псевдокласса::-moz-list-number, который добавляется к селектору **li**. После чего достаточно написать желаемые правила, как показано в примере 1.

**Пример 1. Использование ::-moz-list-number**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Список**</title>**

**<style>**

**li**::-moz-list-number {

color: red; /\* Цвет \*/

font-style: italic; /\* Курсив \*/

}

**</style>**

**</head>**

**<body>**

**<ol>**

**<li>**Первый**</li>**

**<li>**Второй**</li>**

**<li>**Третий**</li>**

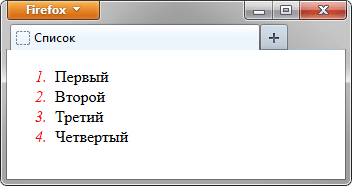
**<li>**Четвертый**</li>**

**</ol>**

**</body>**

**</html>**

Результат данного примера в Firefox показан на рис. 1.



*Рис. 1. Стиль нумерации в списке*

Область применения данного псевдокласса ограничена лишь Firefox; в других браузерах, к сожалению, его аналога нет. Поэтому для универсальности следует пойти другим путём — вообще убрать исходную нумерацию и сделать её с помощью набора свойств counter-reset и counter-increment. Вывод такой нумерации делается через псевдоэлемент ::before и свойство content, как показано в примере 2.

**Пример 2. Изменение вида списка**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="utf-8"**>**

**<title>**Список**</title>**

**<style>**

**ol** {

list-style-type: none; /\* Убираем исходные маркеры \*/

counter-reset: num; /\* Задаём имя счетчика \*/

}

**li**::before {

content: **counter(**num**)** '. '; /\* Выводим число \*/

counter-increment: num; /\* Увеличиваем значение счётчика \*/

color: red;

font-style: italic;

}

**</style>**

**</head>**

**<body>**

**<ol>**

**<li>**Первый**</li>**

**<li>**Второй**</li>**

**<li>**Третий**</li>**

**<li>**Четвертый**</li>**

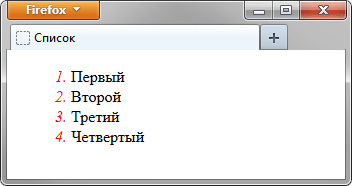
**</ol>**

**</body>**

**</html>**

Для начала у списка прячем родную нумерацию, задавая значение none у свойства list-style-type. Можно вместо него также использовать универсальное свойство list-style, в данном случае результат будет одинаковый. Далее инициируем счётчик, задавая произвольное имя у свойства increment-reset. Это имя нам понадобится в дальнейшем для вывода номера и его инкрементирования (увеличения на единицу). Сам вывод происходит свойством content, оно работает в связке с псевдоэлементом ::before. К нему же добавляем произвольные свойства для стилизации чисел списка.

Поскольку вставка чисел происходит перед текстом пунктов списка, то наша нумерация оказывается сдвинута чуть правее, чем оригинальная (рис. 2). Учитывайте этот момент при создании списка.



*Рис. 2. Нумерация, сделанная через ::before*