

# CSS отступы и поля

Модуль 4 (2 пары)

## Внутренние и внешние отступы. Общая информация

- в CSS предусмотрены 2 вида отступов — внешние, которые называются `margin`, и внутренние с названием `padding`.
- Свойство `padding` определяет расстояние от содержимого (контента) блочного элемента до его границы, а `margin` — расстояние между границами элементов, расположенных рядом.
- Внешние отступы являются прозрачными, а внутренние можно залить фоновым цветом. Внешние отступы имеют такой эффект как схлопывание, то есть объединение значений рядом размещенных элементов. Внешние отступы могут иметь отрицательные значения, а внутренние — только положительные. Внутренние и внешние отступы можно записывать либо в сокращённой форме, либо для каждой из сторон отдельно.
- Отступы можно назначать с 4-х сторон элемента: верхней (англ. название — `top`), правой (англ. название — `right`), нижней (англ. название — `bottom`) и левой (англ. название — `left`):
  - `padding-top/margin-top` — верхний отступ (внутренний/внешний);
  - `padding-right /margin-right` — правый отступ (внутренний/внешний);
  - `padding-bottom /margin-bottom` — нижний отступ (внутренний/внешний);
  - `padding-left /margin-left` — левый отступ (внутренний/внешний);
  - `padding/margin` — отступ (внутренний/внешний) (сокращённая форма).

## Внешние отступы — margin

- Внешние отступы можно задать для всех сторон элемента, например:

```
margin: 10px;
```

- Можно указать отступы для верхнего и нижнего края элемента (первое значение) и через пробел — для левой и правой стороны элемента (второе значение):

```
margin: 10px 2em;
```

- Три значения используют для:
- 1. Верхней стороны,
- 2.левой и правой стороны
- 3. Нижней стороны элемента

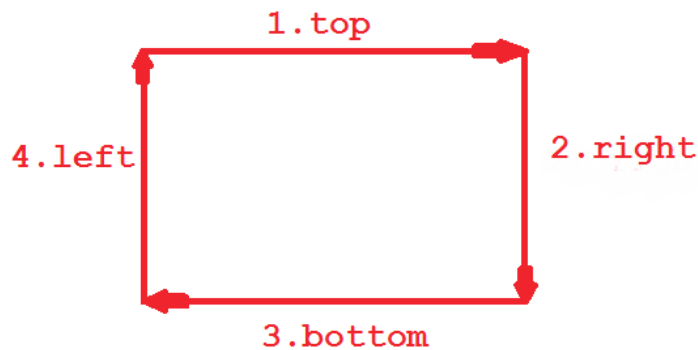
```
margin: 10px 1.5% 30px;
```

- Четыре значения определяют значения внешних отступов в следующей последовательности: сверху справа вниз налево

```
margin: 15px 10% 30pt 0;
```

## Внешние отступы — margin

- Порядок указания величин отступов начинается с левого верхнего угла и идет по часовой стрелке:



- Задают отступы в любых единицах измерений (px, pt, em, rem, % др.). Можно указывать отрицательные величины, auto (браузер рассчитывает величину автоматически) или 0 (отступа нет). В последнем случае единицы измерений не указываются, т.к. 0 хоть в %, хоть в px, хоть в em — все равно 0.

- Можно также назначить только один из отступов:

```
margin-right: 20px;  
margin-left: 2%;  
margin-top: 10pt;  
margin-bottom: 2em;
```

- Еще можно назначить общий отступ для всех сторон, а затем отменить или переопределить его для одной из них:

```
margin: 20px;  
margin-right: 0;
```

## Варианты назначения свойства margin

`margin: 20px;`

`margin: 15px 2.5em;`

`margin: 15px 10% 30px;`

`margin: 15px 10% 30pt 0;`

`width: 70%;  
margin: 30px auto;`

`margin-top: 2.3em;`

`margin-right: 12%;`

`margin-bottom: 5vh;`

`margin-left: 35px;`

`margin: 25px;  
margin-bottom: 0;`

`width: 70%;  
margin: 10px auto 20px;`

`margin: 10px auto;`

## Значение auto

- Интересным является значение auto. Используя его вместе со свойством width, вы предоставляете браузеру рассчитать величину отступа самостоятельно, исходя из размера, указанного для ширины селектора. Например, при указании для body таких свойств:

```
body {  
    width: 1000px;  
    margin: auto;  
}
```

- при ширине браузера в 1340px браузер оставит 1000px на отображение контента, а оставшиеся 340px разделит пополам и сформирует отступы в 170px справа и слева от контента, центрировав его. Так же произойдет, если ширина указана в %. Разница состоит только в том, что сначала браузер рассчитает ширину элемента в px, исходя из ширины браузера или ширины родительского элемента, а затем рассчитает величину отступов. Обычно таким образом указывают отступы не для body, а для какого-либо элемента-«обертки», который часто имеет имя класса или id wrap, wrapper, container, content и т.п. А для body margin устанавливают равными 0, т.к. по умолчанию body имеет внешние отступы в 8px.

```
body { margin: 0}  
.wrapper {width: 80%; margin: auto;}
```

# Схлопывание отступов

- Еще одной особенностью свойства `margin` является «схлопывание» вертикальных отступов у элементов, идущих друг за другом. Например, если задать такие CSS-правила для элементов с классом `block`:

```
.block { margin: 40px 0; }
```

- можно ожидать, что между блоками, идущими друг за другом, отступы будут 80px. Однако в реальности отступы накладываются, и общий отступ будет в 40px

40px

**Lorem ipsum dolor.**

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Cum itaque quaerat quia quis voluptates ullam iure autem doloribus error distinctio.

Eos in illo accusantium corrupti placeat, architecto saepe labore vero. Quibusdam mollitia itaque magnam enim officia consectetur culpa velit accusantium.

40px

**Animi, laborum ipsa.**

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Illo sed quas possimus amet recusandae similique veritatis facilis, asperiores laborum error.

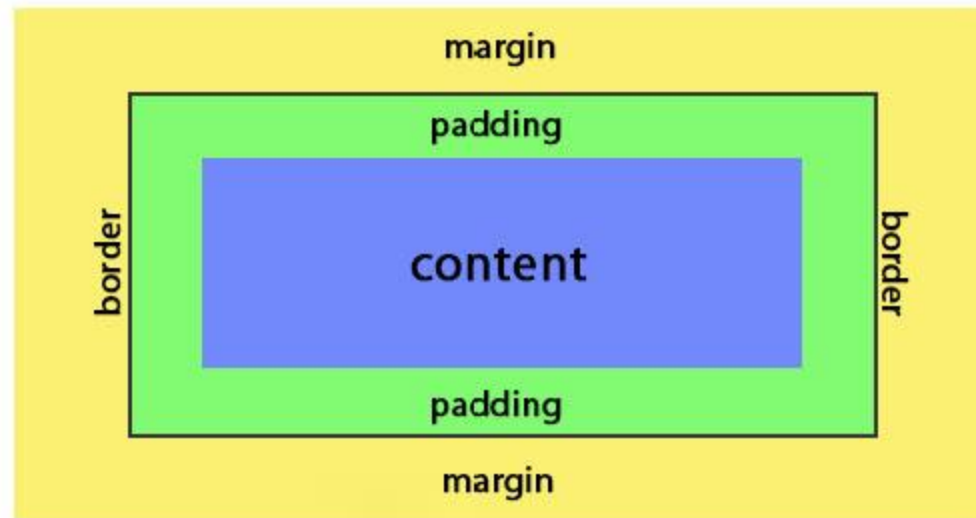
Voluptate ipsam eveniet fugiat neque, esse sint qui quis, illo optio nemo libero aliquam perferendis! Consequatur quam commodi alias quae.

- Схлопывание срабатывает только для вертикальных отступов, но не применяется для горизонтальных. Кроме того, схлопывание не срабатывает:

- ■ для тега `<html>`;
- ■ для строчных элементов;
- ■ для элементов, имеющих свойство `float: left` или `right`;
- ■ у абсолютно позиционированных элементов;
- ■ для элементов, у которых на стороне схлопывания задано свойство `padding`;
- ■ для элементов, у которых на стороне схлопывания задано свойство `border`.

## Внутренние отступы – padding

- Внутренние отступы распространяются от контента блочного элемента до его границы (css-свойство border), если она задана. При заливке элемента фоновым цветом padding обычно также имеет этот цвет



- Внутренние отступы не могут быть отрицательными и они не схлопываются, как внешние. Сходство этих двух отступов в том, что padding можно задавать в px, pt, em, rem, % и т.п., а также делать это по 4-м сторонам блока так же, как и для margin. Также можно использовать значение inherit, которое показывает, что оно наследуется у родителя.



# Внутренние отступы — padding

- Одно значение для всех сторон:

```
padding: 20px;
```

- Отступ по горизонтальным (сверху и снизу) и вертикальным (справа и слева) сторонам:

```
padding: 2em 10px;
```

- Отступ по сверху, отступ для левой и правой стороны и отступ снизу:

```
padding: 2vw 3% 10px;
```

- Отступ сверху, справа, снизу, слева:

```
padding: 1rem 5% 0 20px;
```

- Отдельный отступ по каждой из сторон:

```
padding-top: 15px;  
padding-bottom: 1rem;  
padding-left: 5%  
padding-right: 14pt;
```

- Общее значение отступов с переопределением его с одной (левой) стороны:

```
padding: 20px; padding-left: 3px;
```

# Внутренние отступы — padding

- Визуально блоки со всеми этими значениями будут выглядеть так

```
padding: 20px;
```

```
padding: 2em 10px;
```

```
padding: 2vw 3% 10px;
```

```
padding: 1rem 5% 0 20px;
```

```
padding-top: 15px;
```

```
padding-bottom: 1rem;
```

```
padding-left: 5%;
```

```
padding-right: 14pt;
```

```
padding: 20px;  
padding-left: 3px;
```

- Еще одно отличие внутренних отступов от внешних состоит в том, что свойство `padding: auto` не применяется, т.к. не существует значение `auto` для `padding`.

- Примечание:** при назначении величин отступов в % следует понимать, что проценты рассчитываются от ширины родительского элемента, а не текущего, поэтому отступ 5% для элемента шириной 200px, который помещен внутрь блока с шириной 1000px, будет составлять 50px ( $1000px * 5\% / 100\% = 50px$ ), а не 10 ( $200px * 5\% / 100\% = 10px$ ).

## Значения отступов по умолчанию

- Для того чтобы на странице различные html-элементы хорошо смотрелись даже без css-форматирования, у них есть ряд css-свойств, назначенных по умолчанию. Они подтягиваются из таблицы стилей браузера (*user agent stylesheet*), которая создается разработчиками на основе рекомендаций корпорации W3C. В том числе значения по умолчанию есть у таких свойств, как `margin` и `padding`.

- Рассмотренные элементы списков имеют такие отступы по умолчанию:

```
ul, ol {  
    margin-top: 1em;  
    margin-bottom: 1em;  
    padding-left: 40px;  
}
```

- Важно отметить, что вложенные элементы `ol` или `ul` в многоуровневых списках НЕ имеют отступов сверху и снизу.

- В списках определений значения отступов по умолчанию таковы:

```
dl { margin-top: 1em;  
    margin-bottom: 1em;  
}  
dd { margin-left: 40px;}
```

# Значения отступов по умолчанию

- Для заголовков тоже используются внешние отступы:

```
h1 { margin-top: 0.67em;
      margin-bottom: 0.67em;
    }
h2 { margin-top: 0.83em;
      margin-bottom: 0.83em;
    }
h3 { margin-top: 1em;
      margin-bottom: 1em;
    }
h4 { margin-top: 1.33em;
      margin-bottom: 1.33em;
    }
h5 { margin-top: 1.67em;
      margin-bottom: 1.67em;
    }
h6 { margin-top: 2.33em;
      margin-bottom: 2.33em;
    }
```

- Чем больше цифра заголовка, тем меньше размер шрифта, но больше верхний и нижний внешний отступ. Отступы у заголовков часто приходится изменять, т.к. в psd-макетах дизайнеры их или увеличивают или уменьшают. Абзацы по умолчанию также имеют внешние горизонтальные отступы такой же величины, что и у h3:

```
p {
  margin-top: 1em;
  margin-bottom: 1em;
}
```

## Значения отступов по умолчанию

- Для блочной цитаты отступы по умолчанию назначены со всех сторон, но с разными значениями:

```
blockquote {  
    margin-top: 1em;  
    margin-bottom: 1em;  
    margin-left: 40px;  
    margin-right: 40px;  
}
```

- Для элемента `body`, как уже говорилось ранее, внешние отступы по умолчанию составляют 8px:

```
body { margin: 8px;}
```

- Такие элементы, как `html`, `div`, `section`, `article`, `aside`, `main` по умолчанию НЕ имеют ни внешних, ни внутренних отступов, поэтому при необходимости вы можете назначать их самостоятельно. Чтобы убрать отступы, необходимо назначить 0 в качестве значения:

```
body { margin: 0;}
```

- Если нужно убрать отступы по умолчанию у всех элементов, используют «простейший сброс», устанавливая правила для универсального селектора:

```
* { margin: 0; padding: 0;}
```

# Графика в web-дизайне

## Форматы изображений для веб-использования

- Графические форматы файлов, которые поддерживаются большинством популярных веб-браузеров являются: Graphic Interchange Format (GIF), Joint Photographic Experts Group (JPEG), Portable Network Graphics (PNG) и векторная графика. Некоторые свойства графических файлов:
- **Прозрачность** – это свойство позволяет изображению быть в разной степени прозрачности от твердого состояния до полностью прозрачного.
- **Сжатие** – это свойство позволяет изображению сохраняться в гораздо меньшем файле, с помощью математических алгоритмов для обработки группы пикселей как единого элемента.
- **Переплетение** – позволяет изображению быть загруженным сначала по нечетным строкам, а затем четным. Это позволяет посетителю скорее увидеть изображение.
- **Анимация** – создает видимость движения с помощью серии последовательных снимков. Для анимированного GIF не требуется плагин в браузере и он может работать практически на всех устройствах.
- **Прогрессивная загрузка** – похожа на переплетение тем, что она загружает только часть изображения изначально, но не на основе чередующихся строк.

# GIF

GIF основан в 1980 году и принят веб-дизайнерами в начале 1990-х годов в качестве основного графического формата для веб-страниц. GIF файлы используют алгоритм сжатия, который делает размера файла маленьким для быстрой загрузки. GIF ограничен 256 цветами (8 бит), поддержкой прозрачности и черезстрочной графики. Также есть возможность создавать анимированную графику используя этот формат. Все браузеры могут отображать GIF файлы без проблем.

## **Преимущества GIF:**

- Наиболее широко поддерживаемый графический формат
- Схемы выглядят лучше в этом формате
- Поддержка прозрачности и анимации. Поэтому часто формат GIF используют для создания баннеров.
- Идеальный вариант, если в вашем изображении не большое количество цветов. Для фотографий и полноцветных изображений не подойдет, поскольку будут заметны неплавные переходы между цветами. Лучше всего использовать для оформления дизайна сайта, изображений с однотонной заливкой и там, где требуется прозрачность. Поддерживается практически всеми операционными системами и браузерами.



# JPEG

- Файлы сжаты, но поддерживают “истинные цвета” (24 бит) и являются предпочтительным форматом для фотографий, где вопрос о качестве очень важен . JPEG поддерживает прогрессивный формат, который позволяет почти мгновенно видеть изображение, которое улучшится в качество, когда закончится загрузка.
- В отличие от GIF файлов, веб-дизайнеры могут управлять сжатыми файлами JPEG, что допускает иметь различные уровни качества изображения и размеры файла.

- **Преимущества JPEG:**

- Большое сжатия означает более быструю скорость загрузки.
- Производит отличное качество для фотографий и сложных рисунков.
- Поддержка 24-битного цвета.

# PNG

- PNG является относительно недавним форматом, который был введен как альтернатива для GIF файлов. PNG поддерживает до 24 битный цвет, прозрачность, переплетение и может содержать краткое текстовое описание изображения, которое используется в поисковых системах.
- **Преимущества PNG:**
- Преодолевает 8-битный цвет ограничений в GIF
- Позволяет текстовое описание изображений для поисковых систем
- Поддерживает прозрачность
- Схемы выглядят лучше, чем в JPEG
- Из-за проблем с отображением данного формата в старых браузерах, часто приходилось либо отказываться от него, либо применять всевозможные хитрости. Сейчас, с выходом новых версий браузеров, такая проблема практически отпала.

## Векторная графика

- Большинство веб-графики является растровым изображением или рисунком, который состоит из сетки цветных пикселей. Иллюстрации должны быть созданы в векторной графике, которая состоит из математического описания каждого элемента, который составляет формы линий и цвета изображения. Векторная графика создается путем привлечения таких программ, как Adobe Illustrator или CorelDRAW. Векторная графика должна быть преобразована в любом формате GIF, JPEG или PNG для использования на веб-страницах.
- **Какой формат следует использовать?**
- Веб-дизайнер может выбрать либо GIF или JPEG формат для большинства применений. Но, так как размер файлов GIF, как правило, небольшой по сравнению с размером файла JPEG, большинство веб-дизайнеров будут использовать формат GIF для фона, коробок, кадров и любых других графических элементов, которые выглядят отлично с помощью 8-битного цвета.
- Большинство дизайнеров выберут формат JPEG для фотографий и иллюстраций, где сжатие не идет на компромисс визуальному качеству изображения.

## Цвет фона HTML страницы и отдельных элементов

- **Цвет фона HTML** страницы определяется CSS атрибутом `background-color`, который, в свою очередь, размещается внутри тега `<body>`.
- **HTML цвет фона для отдельных элементов:** блока, параграфа или ячейки таблицы определяется тем же атрибутом, расположенном внутри соответствующих тегов.
- **HTML картинка - фон** определяется с помощью атрибута `background-image` и картинки.

```
<html>
<head>
<title>HTML фон</title>
</head>
<body style="color:Yellow; background-color:#66cc66">
<h1>Заголовок 1-го уровня</h1>
<p>Первый параграф</p>
<p>Второй параграф</p>
<p style="color:#ffffff">Третий параграф</p>
</body>
</html>
```

**Заголовок 1-го уровня**

Первый параграф

Второй параграф

Третий параграф

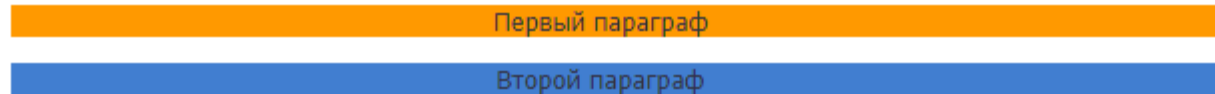
- Строка `style="color:Yellow; background-color:#66cc66"` в теге `<body>` делает цвет фона HTML страницы зеленым, текст – желтым.

# Цвет фона HTML страницы и отдельных элементов

- Определим **цвет фона** для параграфов:

```
<p style="background-color:#ff9900" align="center">Первый параграф</p>  
<p style="background-color:#417ed0" align="center">Второй параграф</p>
```

Результат:

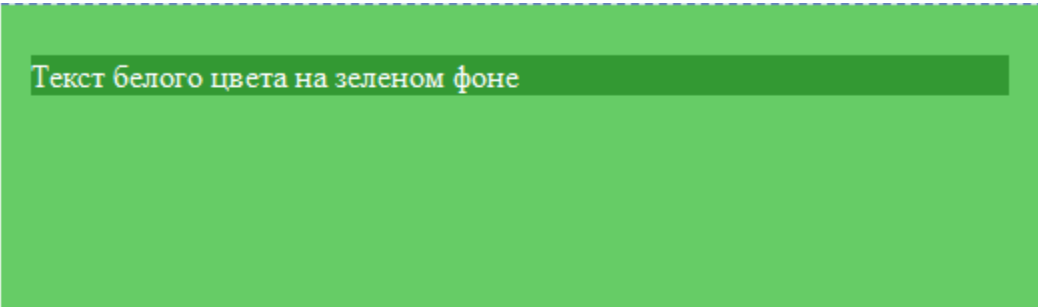


# CSS атрибуты, определяющие фон страницы или элемента

- **CSS цвет фона страницы** будет зеленым:

```
<head>
<title>Пример с цветом фона</title>
<style type="text/css">
body {
background-color:#66cc66
}
.fon_text {
background-color:#339933;
color:#ffffff
}
</style>
</head>
<body>
<p class="fon_text">Текст белого цвета на зеленом фоне</p>
</body>
```

- Атрибуты и значения
- **background-color** – определяет цвет фона.
- **color** – определяет цвет текста.
- Селектор **body** – определяет видимую часть документа.



Текст белого цвета на зеленом фоне

# HTML картинка - фон или изображение в качестве фона

- HTML фон - картинка определяется с помощью изображения
- Например, у нас есть вот такая картинка:
- Напишем следующий код для отдельной
- страницы:



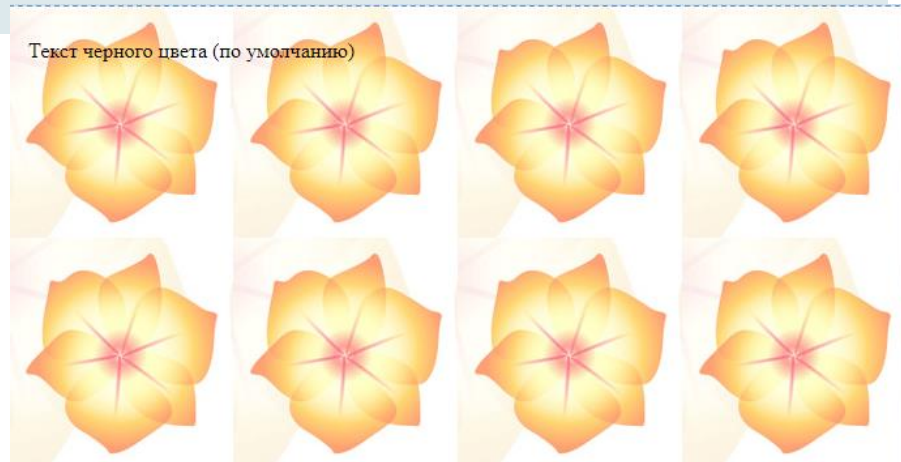
```
<html>
<head>
<title>Картинка - фон в HTML</title>
</head>
<body style="background-image:url(../images/primer-img.jpg)">
<p>Какой-то произвольный текст.</p>
</body>
</html>
```



# CSS фоновое изображение

- `background-image:url(../images/cvetok.png)` определяет изображение-фон.
- По умолчанию изображение-фон распространяется по горизонтали и по вертикали, заполняя собой все пространство, что видно из результата примера. Но возможны и другие варианты.

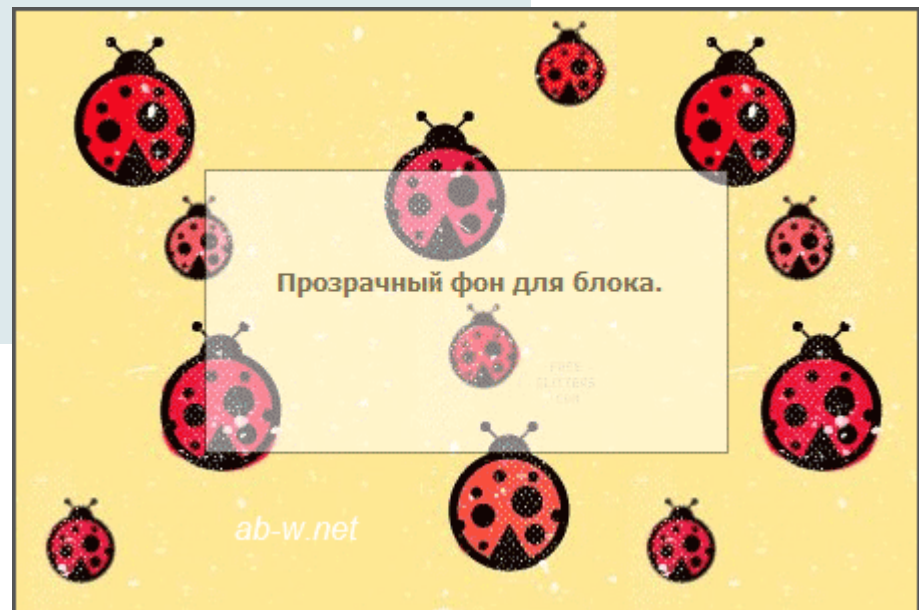
```
<head>
<title>Пример</title>
<style type="text/css">
body {
background-image:url(../images/cvetok.png)
}
</style>
</head>
<body>
<p>Текст черного цвета (по умолчанию)</p>
</body>
```





# CSS прозрачный фон для блока

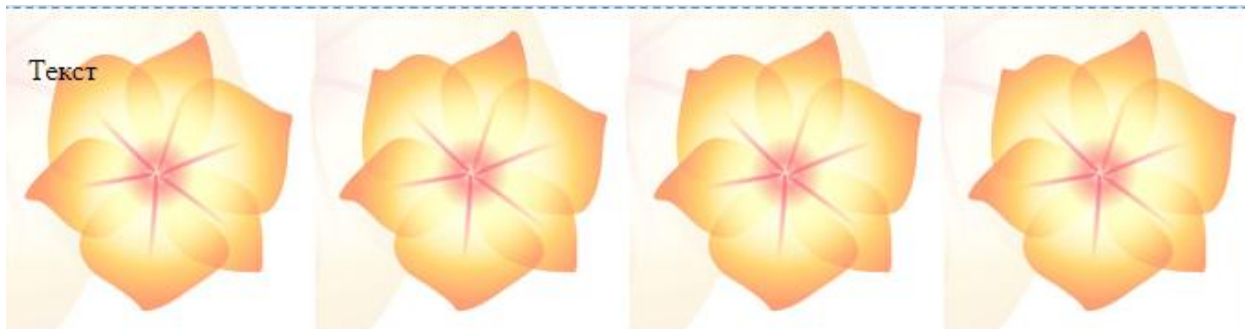
```
<head>
<title>Полупрозрачный элемент</title>
<style type="text/css">
div.page {width:450px; height:300px;
background:url(..images/beetle.jpg) repeat; border:2px solid
#555555}
div.block {width:260px; height:140px; margin-top:79px; margin-
left:94px; background-color:#ffffff; border:1px solid #333333;
/* Internet Explorer */
filter:alpha(opacity=50);
/* CSS3 standard */
opacity:0.5}
div.block p {margin:47px 0 0 10px; font-weight:bold;
color:#000000}
</style>
</head>
<body>
<div class="page">
<div class="block">
<p>Прозрачный фон для блока.</p>
</div>
</div>
</body>
```



# CSS изображение-фон распространяется по горизонтали и вертикали отдельно

```
<head>
<title>Пример</title>
<style type="text/css">
body {
background-image:url(../images/cvetok.png);
background-repeat:repeat-x
}
</style>
</head>
<body>
<p>Текст</p>
</body>
```

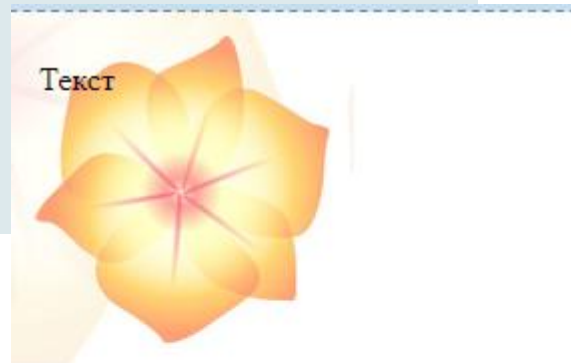
- Атрибуты и значения
- **background-repeat** – управляет распространением изображения-фона.
- Значение **repeat-x** – определяет распространение изображения-фона по горизонтали.
- **background-repeat: repeat-y** – изображение-фон распространяется по вертикали.



# CSS изображение-фон не распространяется

- Значение no-repeat запрещает распространение изображения-фона. Изображение-фон находится в верхнем левом углу, но его местоположение можно изменить

```
<head>
<title>Пример</title>
<style type="text/css">
body {
background-image:url(../images/cvetok.png);
background-repeat:no-repeat
}
</style>
</head>
<body>
<p>Текст</p>
</body>
```



## CSS атрибуты, определяющие изображение-фон и его позицию

- `background-position` определяет горизонтальное (1-е значение) и вертикальное (2-е значение) позиционирование изображения-фона. В результате позиционирования изображение-фон будет установлено по центру страницы, по горизонтали и на расстоянии 80px сверху по вертикали. Значения атрибута `background-position` могут быть определены в пикселях, в процентах или терминами: `left`, `right`, `center`, `bottom`, `top`.

```
<head>
<title>Пример</title>
<style type="text/css">
body {
background-image:url(../images/cvetok.png);
background-repeat:no-repeat;
background-position:center 80px
}
</style>
</head>
<body>
<p>Даже при изменении размеров окна,
фоновое изображение все равно останется в центре.</p>
</body>
```

- Изображение-фон изменяет свою позицию относительно границ элемента, внутри которого оно находится, то есть это могут быть границы окна web-браузера, границы ячейки таблицы или блока `<div>` `</div>`.

# CSS фон и фоновое изображение для блока

```
<head>
<title>Пример</title>
<style type="text/css">
body {
background-color:#66cc66
}
#name {
margin:0 auto;
width:550px;
height:300px;
padding:30px;
border:3px solid #ffff00;
background-color:#ffdd00;
background-image:url(../images/cvetok2.png);
background-repeat:no-repeat;
background-position:100px 100px;
color:#ffffff
}
</style>
</head>
<body>
<div id="name">Текст описуемого блока</div>
</body>
```

## Атрибуты и значения

- **margin** – определяет внешние отступы.
- **margin:0 auto** – определяет центрирование блока по горизонтали.
- **padding** – определяет поля внутри блока.
- **border** – определяет свойства границ.
- **width** – определяет ширину в пикселях или в процентах.
- **height** – определяет высоту.
- **color** – определяет цвет текста.

## CSS фон и фоновое изображение для блока

Что касается механизма позиционирования, то его можно описать так: верхний левый угол блока является началом воображаемой системы координат (0). Верхняя и левая границы соответственно оси x и y. 100 пикселей по горизонтали и 100 пикселей по вертикали определяют координату верхнего левого угла изображения-фона. Вот и все...

Блоки `<div>` `</div>`, манипулируемые CSS, способны на многое. Они активно используются при верстке, участвуют в дизайнерском оформлении web-страниц.

Текст описуемого блока



# CSS фон и фоновое изображение для блока

```
<head>
<title>Пример</title>
<style type="text/css">
body {
background-color:#66cc66
}
#name {
margin:0 auto;
width:550px;
height:300px;
padding:30px;
border:3px solid #ffff00;
background:#ffdd00
url(../images/cvetok2.png) no-repeat 100px
100px;
color:#ffffff
}
</style>
</head>
<body>
<div id="name">Текст описуемого блока</div>
</body>
```

Код из верхнего примера можно напечатать более коротко.

Результат будет аналогичен предыдущему.

background объединяет в себе все свойства фона.

Очередность записи значений следующая:

**background-color**

**background-image**

**background-repeat**

**background-attachment**

**background-position**

Не обязательно использовать все значения background, но последовательность должна быть учтена.

## CSS атрибуты и значения, определяющие неподвижный фон

- Пример того, как зафиксировать фоновое изображение в нужном месте:

```
<head>
<title>Пример</title>
<style type="text/css">
body {
background-color:BlanchedAlmond;
}
.fon_scroll {
margin:0 auto;
width:700px;
height:800px;
padding:30px;
background-color:#ffffff;
background-image:url(../images/7.jpg);
background-repeat:no-repeat;
background-attachment:fixed;
background-position:40px 40px;
}
</style>
</head>
<body>
<div class="fon_scroll">
<p>Текст документа</p>
<p>Текст документа</p>
<p>Текст документа</p>
<p>Текст документа</p>
</div>
</body>
```



# HTML изображение, вставка изображения в код страницы

- HTML изображением может быть любое изображение в формате PNG, JPEG и GIF.
- HTML код изображения определяется тегом `<img />`.
- HTML изображение может быть фоном интернет-страницы.
- HTML изображение может быть определено в роле гиперссылки.
- В папку, где у вас находится файл `index.html` поместите изображение с расширением, скажем, `.jpg`, назовите его как угодно, например, `xxx`.
- Вставка изображения в HTML код страницы:

```
<html>
<head>
<title>Вставка изображения в HTML код страницы</title>
</head>
<body>

</body>
</html>
```

- Тег `<img />` – непарный. Обратите внимание на способ его закрытия.
- Атрибуты и значения
- `src=""` – обязателен, он указывает на источник изображения.
- `alt=""` – определяет альтернативный текст, комментарий, который считывает поисковый робот при анализе содержимого web-страницы. Его также нужно обозначать.
- `width=""` – определяет ширину изображения в пикселях.
- `height=""` – определяет высоту изображения в пикселях.
- Указывайте реальные размеры – так вы сохраните первоначальное качество изображения.

# Отступы по горизонтали и по вертикали или расстояние по горизонтали и по вертикали между изображением и текстом

- Атрибуты и значения
- `hspace=""` – определяет расстояние между изображением и текстом по горизонтали.
- `vspace=""` – определяет расстояние между изображением и текстом по вертикали.

```
<html>
<head>
<title>Отступы между изображением и текстом</title>
</head>
<body>
<p>Текст сверху изображения на дополнительном расстоянии в 20
пикселей</p>

<p>Текст справа от изображения на дополнительном расстоянии в
50 пикселей</p>
<p>Текст</p>
<p>Текст</p>
<p>Текст</p>
<p>Текст</p>
<p>Текст</p>
<p>Текст внизу изображения на
дополнительном расстоянии в 20 пикселей</p>
</body>
</html>
```

Текст сверху изображения на дополнительном расстоянии в 20 пикселей



Текст справа от изображения на

Текст

Текст

Текст

Текст

Текст

Текст внизу изображения на дополнительном расстоянии в 20 пикселей

# HTML изображение - ссылка

- Ссылка наверх текущей страницы:

```
<title>HTML изображение - ссылка</title>
</head>
<body>
<p>
<a href="#"></a>
</p>
```



- `border="0"` – отменяет границу графической ссылки.

# HTML изображение по центру страницы

- Или все возможные способы размещения изображений по центру
- Код примера:

```
<html>
<head>
<title>HTML изображение по центру
страницы</title>
</head>
<body>
<p align="center">

</p>
<div style="text-align:center">

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



- На что здесь необходимо обратить внимание? → Во-первых, на то, что обозначены размеры — это ускоряет загрузку изображения. Во-вторых, прописаны атрибуты alt="", что также крайне желательно делать, даже если альтернативный текст отсутствует. В первом случае центрирование было определено HTML параметром, а во втором — с помощью линейного включения каскадных таблиц стилей.

# Текст сверху, внизу, по центру изображения

```
<html>
<head>
<title>HTML текст сверху, внизу, по центру изображения</title>
</head>
<body>
<p>Текст сверху изображения</p>
<p> Текст по центру изображения</p>
<p> Текст внизу изображения</p>
</body>
</html>
```



Текст сверху изображения



Текст по центру изображения



Текст внизу изображения

- Атрибуты и значения
- align="top" – выравнивает изображение и текст по верху.
- align="middle" – выравнивает изображение и текст по центру, по вертикали.
- align="bottom" – выравнивает изображение и текст по низу.
- Обратите внимание на способ подгрузки изображения: ../images/2121.png. Во-первых, использован формат PNG (.png). Во-вторых, изображение находится в отдельной папке, то есть документ у меня в одной папке, а изображение в другой. В таких ситуациях очень важно правильно указать путь от документа к подгружаемой картинке, что и было сделано: первые две точки .. определяют выход из папки где находится документ (все уроки, они же страницы, они же документы курса по HTML у меня находятся в одной папке, CSS – в другой, изображения – в третьей и так далее); /images/ – не что иное, как название папки с изображениями, а 2121.png – полное имя файла самого изображения.

# HTML изображение слева - текст справа

```
<html>
<head>
<title>HTML изображение слева - текст
справа</title>
</head>
<body>

<p>Изображение обтекает текст слева</p>
<p>Изображение обтекает текст слева</p>
<p>Изображение обтекает текст слева</p>
<p>Изображение обтекает текст слева</p>
</body>
</html>
```



Изображение обтекает текст слева

Изображение обтекает текст слева

Изображение обтекает текст слева

Изображение обтекает текст слева

## Придание изображению структуры и установка заголовка

- Есть множество путей как вы можете добавить заголовок к своему изображению. Для примера, нет ничего, что может вас остановить сделать

```
<div class="figure">
  

  <p>A T-Rex on display in the Manchester University Museum.</p>
</div>
```

- Это нормально. Это содержит всё что вам нужно, и красиво стилизуется с помощью CSS. Но есть проблема: здесь нет ничего, что семантически связывает изображение с его заголовком, и это может вызвать сложности для читателей. Например, когда у вас есть 50 изображений и заголовков, какой заголовок идёт вместе с каким изображением?

## Придание изображению структуры и установка заголовка

- Лучшим решением будет использование HTML5 `<figure>` и `<figcaption>` элементов. Они были созданы исключительно для этой цели: предоставление семантического контейнера для иллюстраций, и прозрачного связывания иллюстрации с её заголовком. Наш пример выше, мог быть переписан как то так:

```
<figure>
  
  <figcaption>A T-Rex on display in the Manchester University Museum.</figcaption>
</figure>
```

- `<figcaption>` элемент говорит браузерам, и вспомогательной технологии, что заголовок описывает содержимое `<figure>` элемента.
- С точки зрения доступности, заголовки и `alt` имеют различные предназначения. Заголовки помогают даже тем, кто имеет возможность просматривать изображение, тогда как `alt` предусматривает замену функционала отсутствующего изображения. Таким образом, заголовки и `alt` не подразумевают под собой одни и те же вещи, потому что оба используются браузером при отсутствии изображения. Попробуйте отключить изображения в своём браузере, чтобы увидеть как это выглядит.
- Тег `<figure>` не является изображением. Он представляет собой независимый структурный элемент, который:
  - Может использоваться в различных местах страницы.
  - Предоставляет ценную информацию, поддерживающую основной текст.
  - Тег `<figure>` может быть несколькими изображениями, куском кода, аудио, видео, уравнением, таблицей, либо чем-то другим.



## Придание изображению структуры и установка заголовка

- Есть множество путей как вы можете добавить заголовок к своему изображению. Для примера, нет ничего, что может вас остановить сделать

```
<figure>
  <p>
  <figcaption>Масштабированная модель Эйфелевой башни в Парке
    Мини-Франция</figcaption>
</figure>
```

- Например, чтобы сместить изображение вправо на расстояние, равное 30% от ширины окружающих абзацев, используйте следующие правила:

```
figure {
  float: right;
  width: 30%;
  text-align: center;
  font-style: italic;
  font-size: smaller;
  text-indent: 0;
  border: thin silver solid;
  margin: 0.5em;
  padding: 0.5em;
}
```



- На самом деле, действительно необходимы только две первые декларации (float и width), а остальные использованы исключительно для оформления.

## Масштабирование изображения

- Здесь только одна проблема, и она заключается в том, что изображение может быть слишком широким. В этом случае, ширина изображения всегда будет составлять 136 px и иллюстрация будет занимать 30% от окружающего текста. И если вы сузите окно, то изображение может не поместиться и вылезти за рамку (попробуйте!).
- Если вы знаете ширину всех изображений в документе, вы можете указать минимальную ширину иллюстрации следующим образом:

```
figure {  
  min-width: 150px;  
}
```

- Другой способ — это задать масштаб самого изображения. Именно это мы и сделали с изображением справа. Как вы, возможно, видите, если вы сделаете окно браузера слишком широким, изображения в формате JPEG масштабируются не очень хорошо. Но если это изображение — диаграмма или график в формате SVG, то масштабирование работает просто великолепно.



Сен-Тропе и его форт в  
вечернем солнце

# Масштабирование изображения

```
figure {  
  float: right;  
  width: 30%;  
  text-align: center;  
  font-style: italic;  
  font-size: smaller;  
  text-indent: 0;  
  border: thin silver solid;  
  margin: 0.5em;  
  padding: 0.5em;  
}  
img.scaled {  
  width: 100%;  
}  
  
<figure>  
  <p>  
  <figcaption>Сен-Тропе и его форт в вечернем солнце</figcaption>  
</figure>
```

- **Правило:** этот приём делает изображение настолько широким, насколько позволяет пространство внутри иллюстрации (область внутри рамок и отступов — `border` и `padding`).

## Размещение подписи сверху

- HTML позволяет элементу `figcaption` быть либо первым, либо последним элементом внутри иллюстрации. Если не применять каких-либо правил CSS, это приведёт к тому, что подпись будет размещена либо над иллюстрацией, либо под ней соответственно.
- Однако, независимо от разметки текста, вы можете указать в CSS, чтобы подпись появилась либо над изображением, либо под ним. Этого можно достичь, указав браузеру, что изображение должно быть отформатировано как таблица, в которой картинка является единственной ячейкой, а подпись становится заголовком таблицы. Просто добавьте эти правила в таблицу стилей из предыдущей секции:

```
figure {  
    display: table;  
}  
figcaption {  
    display: table-caption;  
    caption-side: top;  
}
```



## Размещение подписи сверху

- Стил, который мы использовали на этой странице, содержит рамку серого цвета. Она обрамляет иллюстрацию. К сожалению, когда мы используем табличную разметку, чтобы поместить подпись сверху или снизу, мы должны указать рамку другим способом, потому что подпись размещена за пределами границы таблицы. Мы можем исправить это, поместив часть границы на саму подпись:

```
figure {  
  border-top: none;  
  padding-top: 0;  
}  
figcaption {  
  padding: 0.5em;  
  border: thin silver solid;  
  border-bottom: none;  
}
```

# Адаптивные изображения с помощью CSS

- Существует не мало различных решений сделать изображения адаптивными, все они различаются и по сложности, и степени поддержки браузерами. Примером сложного пути реализации адаптивных картинок, является использование атрибута `srcset`, для которого требуется несколько изображений, больше разметки, а также зависимость от поддержки браузерами.

- Современные спецификации позволяют нам сделать изображения, используемые на страницах сайтов, гибкими и корректно отображающимися на экранах различных пользовательских устройств, для этого достаточно использовать всего лишь несколько свойств из обоемы CSS.

- Все варианты основаны на использовании процентных значений для свойства **width** (ширины) и значения **auto** для свойства **height** (высоты) изображений.

## Адаптивные картинки в колонках

Эти изображения отображаются рядом друг с другом. Они будут пропорционально изменяться в зависимости от размера окна просмотра, при зом сохраняя свои позиции.

Две Колонки



Три Колонки



## Условная растановка изображений

Эти изображения будут отображаться в одну колонку на небольших устройствах (например, смартфонах), в две колонки на средних устройствах (например, планшет), и в четыре колонки на экранах (например, ПК или ноутбук).





## Базовые значения адаптивного изображения

- Начнём с рассмотрения базового примера, когда нам необходимо сделать одиночные картинки используемые в записях, или других отдельных блоках, полностью адаптивными.
- Например, у нас есть контейнер, которому мы задали базовую ширину `width: 96%`; и выставили максимальную ширину в `max-width: 960px`;, в этом блоке нам необходимо вывести адаптивное изображение.
- Для этого элементу `<img>` внутри контейнера определяем ширину в `100%`, так, что его ширина всегда будет равна ширине контейнера, независимо от размера области просмотра. Высоту, соответственно, переводим в автоматический режим, в итоге изображение будет изменяться пропорционально.

```
<div class="container">
  
</div>
div.container {
  width: 96%;
  max-width: 960px;
  margin: 0 auto; /* центрируем основной контейнер */
}
img {
  width: 100%; /* ширина картинки */
  height: auto; /* высота картинки */
}
```

- Обратите внимание, что `<img>` элемент будет адаптивным, даже если были заданы фиксированные значения HTML-атрибутов ширины и высоты непосредственно в разметке.

## Адаптивные изображения в колонках

- Иногда мы хотим видеть изображения выстроенные в ряд бок о бок, или например, в виде сетки, для организации простейшей галереи картинок.
- Для этого, необходимы лишь внести небольшие изменения в код, который использовали выше, первое, это уменьшить ширину свойство `width` и задать элементу `<img>` значение `inline-block` для свойства `display`, т.е. сделать его встроенным.
- Давайте рассмотрим две компоновочные схемы: расположение картинок в две колонки и макет из трёх столбцов.
- 1. Макет изображений в две колонки
- Для двух-колоночного макета изображений, мы можем установить ширину в 48%, или примерно половину контейнера. Не устанавливаем значения в 50%, для того, чтобы были боковые отступы.

```
<div class="container">
  
  
</div>
img {
  width: 48%;
  display: inline-block;
}
```



## Адаптивные изображения в колонках

- 2. Три колонки изображений
- С трёх-колоночным макетом концепция та же, необходимо распределить ширину базового контейнера на три картинки, для этого достаточно установить значения ширины изображений около одной трети ширины контейнера: 32%.

```
<div class="container">
  
  
  
</div>
```

```
img {
  width: 32%;
  display: inline-block;
}
```

# Условная расстановка адаптивных изображений

- В следующем примере, мы рассмотрим вариант использования адаптивных картинок с различной расстановкой в зависимости от устройств просмотра, т.е. при просмотре на смартфонах изображения будут отображаться в одну колонку, в две колонки на планшетах, и выстраиваться в четыре колонки на больших экранах.
- Для реализации задуманного, применим медиа-запросы @media, указав тип носителя, для которого будет применяться то или иное максимальное значение ширины изображений max-width.

```
<div class="container">
  
  
  
  
</div>
/* Для небольших устройств (смартфоны) */
img {
  max-width: 100%;
  display: inline-block;
}
/* Для средних устройств (планшеты) */
@media (min-width: 420px) {
  img {
    max-width: 48%;
  }
}
/* Для больших устройств (ноуты, ПК) */
@media (min-width: 760px) {
  img {
    max-width: 24%;
  }
}
```

## Адаптивное изображение на всю ширину экрана

- Для того, чтобы сделать широко-форматные адаптивные изображения, которые заполняют 100% размера окна просмотра, необходимо просто удалить свойство максимальной ширины контейнера `max-width` (значение в 960px) и установить ему ширину `width` в 100%. Ширина изображения, так же выставляется в значение 100%.

```
.container {  
  width: 100%;  
}  
img {  
  width: 100%;  
}
```

- Несмотря на то что данная техника очень проста в использовании и имеет устойчивую поддержку браузерми, следует помнить о том, что изображения всегда будут показаны в полный размер, т.е. большие, с высоким разрешением изображения показываются заполняя всё пространство, что для небольших мобильных устройств, не всегда в тему, если только картинка не используется в качестве фонового изображения.

# Полезные ссылки

- <https://old.fotostars.me/> - фоторедактор
- <https://encycolorpedia.ru/> - подбор цвета
- <http://freeanalogs.ru/Online/Gimp> -  
растровый графический редактор,  
бесплатная замена фотошопу

**Спасибо за внимание.**