Введение в CSS

CSS (каскадные таблицы стилей)

До появления CSS

- До появления CSS, дизайнеры и верстальщики прописывали стили внутри html тегов. Это довольно неудобно, так как при желании изменить какой-либо стиль, необходимо делать изменения во многих местах, да и кроссбраузерность никто гарантировать не мог.
- После своего появления каскадные таблицы стилей решили все эти проблемы.

- Рассмотрим способы подключения стилей к странице.

Способы подключения стилей к странице (1)

1. Подключение CSS отдельным внешним файлом с расширением .css. Самый удобный способ. Он гарантирует максимально лёгкое и удобное редактирование стилей и не нагружает ими страницу. К самой странице подключается так:
   Между тегами

<head></head>

вставляется код:

```
<link href="style.css" rel="stylesheet" type=text/css>
```

Соответственно, файл стилей должен называться style.css и находиться в одной папке с самим файлом html.

Способы подключения стилей к странице (2,3)

2. Встраивание блока стилей в саму страницу. Удобство этого метода состоит в том, что не надо создавать дополнительных файл, а главный недостаток - нагрузка стилями html кода. Подключается так:

```
<style type="text/css">Стили… </style>
```

3. Встраивание стилей в сами теги. Самый неудобный способ, так как затрудняет последующее редактирование и нагружает страницу. Подключается так: внутри кода пишется

```
style=&#8221;стили&#8221;
```

Преимущества использования CSS

- Очищение html-кода от стилей
- Ускорение загрузки страниц сайта
- Максимальная кроссбраузерность
- Расширенные возможности работы со стилями
- Возможность достижения интересных графических эффектов

Оформление заголовков

- Заголовок - один из важнейших аспектов преставления текста. Он даёт читателю понятие об основной теме текста, его направлении и т.д.

- В языке html, заголовки задаются тегом
- <h1>, <h2>, <h3>....
- Цифра после буквы h называется уровнем заголовка и определяет высоту и ширину символов. (h1 - самый большой).
- Оформлять заголовок можно непосредственно в коде, но это оказывается крайне неудобным и громоздким.
- Посмотрим, как это можно сделать оптимальным образом - с помощью CSS.
- Существует довольно много стилей, которые могут быть применены к заголовку. Мы можем изменить размер шрифта, его положение на странице, цвет, начертание и т.д. Попробуем сделать это в несколько этапов.

1. Задание цвета. Создадим html файл и запишем туда такой код:

```
<html>
    <head>

    <style type="text/css" media="screen">
    h1 {color:#003366;}
    h2 {color:#e2b500;}
    h3 {color:#9d0000;}
    </style>

    </head>
    <body>
    <h1>Основной заголовок</h1>
    <h2>Заголовок 2-го уровня</h2>
    <h3>Заголовок 3-го уровня</h3>
    </body>
    </html>
```

Все стили прописываются в фигурных скобках через ;

2.Выравнивание

- Теперь поставим заголовок 1-го уровня по центру страницы.
- Для этого в списке стилей h1, после **color:#.....;** добавим такой стиль: **text-align:center;**

```
<style type="text/css" media="screen">
   h1 {color:#003366; text-align:center;}
   h2 {color:#e2b500;}
   h3 {color:#9d0000;}
   </style>
```

3. Начертание

- Теперь изменим начертание шрифта для заголовка 2-го уровня. Для примера возьмём шрифт Verdana. Добавим только для h2 такой стиль: **font-family: Verdana;**

```
<style type="text/css" media="screen">
   h1 {color:#003366; text-align:center;}
   h2 {color:#e2b500; font-family: Verdana;}
   h3 {color:#9d0000;}
   </style>
```

4. Фон и граница

- Для примера, выделим заголовок 3-го уровня фоновым цветом. и границей (сделаем вид блока). Добавим для него такие стили: **background: #d4e6ff; border: solid 1px #006cff; width: 200px;**

```
<style type="text/css" media="screen">
   h1 {color:#003366; text-align:center;}
   h2 {color:#e2b500; font-family: Verdana;}
   h3 {color:#9d0000; background: #d4e6ff;
   border: solid 1px #006cff; width: 200px;}
   </style>
```

Задание стилей основного блока (пример 1)

- /* Задание стилей основного блока */
  .body {
  color:#333; /* Задание цвета текста */
  background-color: #d2efff; /* Задание цвета фона */
  border: 2px solid #ffba00; /* Задание сплошной границы блока шириной в 2 пикселя и её цвета */
  border-top-style: none; /* Удаление верхней границы блока */
  font-size: 12px; /* Задание размера шрифта */
  padding: 5px; /* Задание отступа в 5 пикселей со всех сторон */
  }

Пример 2 использования файла CSS

```css
@charset "windows-1251";
/* CSS Document */
.body{
    padding:0px;
    margin:0px;
    font-family: Verdana, serif;
    font-size: 12px;
    font-weight: normal;
    letter-spacing: normal;
    background-color: #f7f7f7;
    background-image:
url(../../images/bg_body.jpg);
    background-repeat: repeat-x;
    background-position: left top;
}

.justify {
    text-align: justify;
    padding: 10px;
}
```

текст текст текст 0 текст текст текст текст текст текст
текст текст текст текст текст0 текст текст текст текст
текст текст текст текст текст 2 текст текст текст текст
текст текст текст текст текст текст текст текст текст
текст текст текст текст текст текст текст текст текст 3
текст текст текст текст текст текст текст текст 8 текст
текст текст текст текст текст текст текст текст текст
текст

lkj lk lj lk lkj lk lk lk lkj lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk j
lkj lk lj lk lkj lk lk lk lkj lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk
jlkj lk lj lk lkj lk lk lk lkj lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk
jlkj lk lj lk lkj lk lk lk lkj lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk
jlkj lk lj lk lkj lk lk lk lkj lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk
jlkj lk lj lk lkj lk lk lk lkj lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk
jlkj lk lj lk lkj lk lk lk lkj lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk
jlkj lk lj lk lkj lk lk lk lkj lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk
jlkj lk lj lk lkj lk lk lk lkj lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk
jlkj lk lj lk lkj lk lk lk lkj lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk j

для html-файла

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
<title>Страница</title>
<link rel="stylesheet" type="text/css" href="css/222.css">
</head>
<body class="body" >
<p>текст текст текст 0 текст текст текст текст текст текст текст текст текст текст текст0  текст
текст текст текст текст текст текст текст текст 2 текст текст текст текст текст текст текст текст
текст текст текст текст текст текст текст текст текст текст текст текст текст текст  3 текст текст
текст текст текст текст текст текст  8 текст текст текст текст текст текст текст текст текст текст
текст </p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p class="justify"> lkj lk lj lk lkj lk lk lk lkj lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk j lkj lk lj lk lkj lk lk lk lkj
lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk jlkj lk lj lk lkj lk lk lk lkj lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk jlkj lk lj lk lkj lk lk lk
lkj lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk jlkj lk lj lk lkj lk lk lk lkj lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk jlkj lk lj lk lkj lk
lk lk lkj lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk jlkj lk lj lk lkj lk lk lk lkj lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk jlkj lk lj lk
lkj lk lk lk lkj lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk jlkj lk lj lk lkj lk lk lk lkj lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk jlkj lk
lj lk lkj lk lk lk lkj lkjlkjlkjlkjl lkjlkj lkj ljlkjlk jlk jlk jlk j</p>
</body>
</html>
```

Вставка swf-файла

```
<object
classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000
" codebase=
"http://download.macromedia.com/pub/shockwave/cabs/fla
sh/swflash.cab#version=6,0,0,0" width="100%"
height="180">
  <param name="movie" value="intro_gal_2008.swf" >
  <param name="quality" value="high" >
  <param name="wmode" value="opaque"/>
  <embed src="intro_gal_2008.swf" wmode="opaque"
width="100%" height="180" quality="high"
pluginspage="http://www.macromedia.com/go/getflashplay
er" type="application/x-shockwave-flash"></embed>
</object>
```

Урок

Создание простого блока

Создание простого блока (css-файл)

```
/* Задание стилей всего блока */
#block{
    width: 250px; /* Задание ширины блока */
}
/* Задание стилей заголовка */
.head {
    text-align:center; /* Выравнивание заголовка по центру блока */
    color: #fff; /* Задание цвета заголовка (тут - белый) */
    background-color: #0274b0; /* Задание цвета фона (тут - синий) */
    border: 2px solid #ffba00; /* Задание сплошной границы блока шириной в
2 пикселя и её цвета */
    font-size: 15px; /* Задание размера шрифта заголовка */
    font-weight:bold; /* Задание полужирного начертания шрифта */
    padding: 7px 0 7px 0; */ Задание верхнего и нижнего отступов текста
заголовка от границ блока */
}
```

Задание стилей основного блока

- /* Задание стилей основного блока */
  .body {
  color:#333; /* Задание цвета текста */
  background-color: #d2efff; /* Задание цвета фона */
  border: 2px solid #ffba00; /* Задание сплошной границы блока шириной в 2 пикселя и её цвета */
  border-top-style: none; /* Удаление верхней границы блока */
  font-size: 12px; /* Задание размера шрифта */
  padding: 5px; /* Задание отступа в 5 пикселей со всех сторон */
  }

```
<html>
<head>
<link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<div id="block">
<div class="head">
ПРОСТОЙ БЛОК
</div>
<div class="body">
Уроки CSSвёрстки. <br>
Начнем с простого, и рассмотрим один из способов создания стандартного блока, состоящего из "шапки" и основной части...
</div>
</div>
</body>
</html>
```

Урок

Прижимаем footer к низу страницы

Прижимаем footer к низу страницы (css)

```
html, body { margin: 0;  padding: 0;  height: 100%; /* не забываем это свойство для html и body */ }
#conteiner { position: relative;  min-height: 100%; /* минимальная высота контейнера, ее понимают все браузеры за исключением IE6 */ }
/* ниже фильтр * html так называемый CSS хак, через который можно задать любое свойство предназначенное для IE6 */
* html #conteiner { height: 100%;}
#header {position: relative; height: 2.5em;}
```

```
/* элемент ниже позволяет не наезжать подвалу на контент, он должен быть не меньше подвала по высоте */
.end_content { position: relative; height: 2.6em;}
#footer {
    background-color:#1C7FFF;
    color:#FFF;
    position: relative;
    margin-top: -2.5em; /* свойство должно быть равным высоте элемента, не забываем про отрицательное значение */
    height: 2.5em;
}
```

```
<html>
<head>
<link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
  <div id="conteiner">
    <!-- Обращаю Ваше внимание на то что Header (шапка) внутри контейнера -->
    <div id="header">title</div>
    content
    <div class="end_content"></div>
  </div>
  <!-- footer (подвал) за контейнером -->
  <div id="footer">footer</div>
</body>
</html>
```

Урок

Пример безтабличной верстки

Простой шаблон

http://www.css-school.ru/css_verst/44-urok-css-vjorstki-4.-prostojj-shablon.html

Часть 2

CSS-навигация

Правила оформления списков

- В html различается 2 вида списков: нумерованный и ненумерованный (маркированный). Они задаются тегами "ol" и "ul" соответственно. Элементы обоих видов списков задаются тегом "li".
- Для примера, составим простой маркированный список (именно с ними чаще всего приходится работать в CSS):

```
<ul>
   <li>Пункт 1</li>
   <li>Пункт 2</li>
   <li>Пункт 3</li>
</ul>
```

Для нумерованных (ol)

- Оформление (стиль) задаётся одним атрибутом: list-style-type.
- Значения этого атрибута – фиксированы и различны для нумерованных и маркированных списков.

  Для нумерованных (ol):
  decimal – арабские цифры (по умолчанию);
  lower-roman – малые римские цифры;
  upper-roman – большие римские цифры;
  lower-alpha – малые (строчные) латинские буквы;
  upper-alpha – большие (прописные) латинские буквы

Для маркированных (ul)

- Для маркированных (ul):
  disk – простой круг (по умолчанию);
  circle – кольцо;
  square – квадрат;
  lower-alpha – малые (строчные) латинские буквы;
  none – отмена оформление (благодаря этому значению атрибута маркированные списки удобно использовать как меню и т.д.).

Маркер-рисунок

Также, можно задать рисунок как маркер списка.

Делается это так:

list-style-image: url(Адрес рисунка)

Положение маркера

С помощью стилей можно задать и положение маркера относительно текста. Это задаётся атрибутом **list-style-position**. Он имеет следующие значения:

outside – вне текста (по умолчанию);
inside – внутри текста;

Все стили могут быть заданы в одном атрибуте **list-style**.
Например:

list-style: inside

Меню с помощью списка (1)

- Создадим вертикальное меню с помощью списка. Начнем с создания файла стилей style.css и файла index.html с таким начальным кодом (содержанием):

```
<html>
<head>
<link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<ul>
    <li><a href="#">Пункт 1</a></li>
    <li><a href="#">Пункт 2</a></li>
    <li><a href="#">Пункт 3</a></li>
</body>
</html>
```

- Пункт 1
- Пункт 2
- Пункт 3

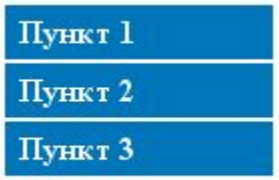
Меню с помощью списка (2)

Дальше работаем только с css файлом.
Запишем в css файл такой код:

```
ul {list-style:none;}

    li a {color:#fff; text-decoration:none; padding:
    4px 7px; width:120px; background:#0076ba;
    display:block; border:1px solid #fff;
    font-weight:bold;}
```

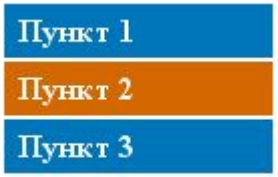
Мы задали сразу стили для всего списка и его элементов в случае, если они – ссылки. Должно получиться так:



Меню с помощью списка (3)

Теперь добавим эффектов при наведении мыши на кнопку:

```
li a:hover {
  background:#d36800;
  }
```



Вот, и всё. Наше меню готово.

Атрибут CSS 2.0 «z-index» (1)

Атрибут z-index допустимо применять только к свободно позиционированным объектам, то есть у всех слоёв должен стоять атрибут position:relative.

Как можно догадаться из вида атрибута z-index, он определяет положение объектов по оси Z и создает, таким образом, иллюзию трёхмерности страницы. Любой элемент может быть выше или ниже другого.

Атрибут CSS 2.0 «z-index» (2)

По умолчанию значение атрибута z-index – auto. При этом значении элементы идут один за другим по порядку написания в коде. Атрибут может быть задан положительным, отрицательным числом, или нулём. Например:

```
z-index:5;
```

Объект с большим значением атрибута z-index будет перекрывать атрибут с меньшим его значением. То есть объект с z-index равным 5 перекроет объект с z-index равным 3.

Атрибут поддерживается Internet Explorer начиная с версии 4.0. Для тегов frame и iframe поддерживается Internet Explorer начиная с версии 5.5.

Применение атрибута z-index

- Атрибут применяется чаще всего для позиционирования сложных и/или активных блоков. В остальных случаях, когда код более-менее прост, стараются обходиться без этого атрибута, ведь ничего не мешает расположить, в коде, объект, находящийся выше над объектом, находящимся ниже. Ещё есть некоторые проблемы при использовании атрибута z-index к активным блокам, да и есть множество более простых способов оформления.

  Итак, не смотря на то, что без атрибута z-index можно легко прожить, он остаётся достаточно востребованным в современной вёрстке.

z-index. Пример. index.html

Сделаем эффект лёгкой тени для текста без использования картинок, зато с использованием слоёв и атрибута z-index.

Как обычно, создадим файлы index.html и style.css. Запишем в веб-страницу стандартный начальный код:

```
<html>
    <head>
    <link rel="stylesheet" href="style.css" type="text/css" />
    </head>
    <body>
   //сам слой
<div id="text">Любой текст</div>
    //слой тени
<div id="shadow">Любой текст</div>
    </body>
    </html>
```

z-index. Пример. style.css

В style.css запишем:

```
#text { font:30px Tahoma;
        position: relative;
        color:#002f5f;
        z-index:2;}
#shadow {font:30px Tahoma;
        position: relative;
        color:#ccc;
        top:-35px;
        left:2px;
        z-index:1;}
```

Заметим, что z-index тени меньше, чем у самого текста. Это значит, что тень будет именно там, где и должна быть – под текстом. Если сделали всё правильно должно получиться так:

Сравнение двух типов верстки (блочного и табличного)

Рассмотрим их достоинства и недостатки, попробуем найти оптимальный и максимально эффективный тип верстки…
Итак, начнём с описания старейшего типа верстки – табличного. Он основан на теге “table” и имеет примерно такую структуру:

```
<table>
    <tr>
        <td>Содержание первого столбца</td>
        <td>Содержание второго столбца</td>
    </tr>
</table>
```

Таким образом, мы задали таблицу, состоящую из 1 строки и 2 столбцов. Для ясности: “tr” обозначает строку, а “td” – столбец.
- По умолчанию, для таблиц применены следующие стили:
- Граница (border): 0 пикселей (нет)
- Поля ячеек (cellpadding): 1 пиксель
- Интервал ячеек (cellspacing): 2 пикселя

Достоинства и недостатки табличного позиционирования

- У таблиц множество положительных сторон, многие из которых субъективны. Однако, есть и бесспорные:
  - С помощью таблиц можно добиться практически любого расположения элементов
  - Таблицы имеют привычный для пользователя вид
  - С помощью таблиц можно точно закрепить любой элемент
  - Вёрстка с помощью таблиц проста для понимания и освоения

- Однако, у противников таблиц есть свои аргументы. Некоторые из них вполне убедительны:
  - Таблицы бывают очень громоздки
  - Таблицы не всегда способны обеспечить полную кроссбраузерность и поддержку различных разрешений экрана
  - Таблицы замедляют скорость загрузки страниц сайта (спорное утверждение, о нём поговорим ниже).

Достоинства и недостатки блочной верстки

- Блочной вёрсткой, называется вёрстка, выполненная без использования таблиц.
- В HTML-коде используются такие теги как div, span и многие другие. Сама же форма страницы формируется с помощью стилей (CSS).
- Положительные стороны блоков:
  - Малый размер кода
  - Основа дизайна – CSS
  - Широкие возможности позиционирования элементов
  - Быстрая загрузка страниц
- Отрицательные стороны блоков:
  - Практически никогда не удаётся достигнуть полной кроссбраузерности
  - Метод сложен для понимания новичками
  - Большой файл стилей

Что выбрать?

- Очень многие веб-разработчики, которые считают себя «продвинутыми» призывают к полному отказу от табличной верстки, как от устаревшего способа. Но эти люди частенько забывают одну очень важную деталь – многие посетители их сайтов будут пользоваться устаревшими браузерами, которые могут поддерживать CSS только на базовом уровне (Internet Explorer 6, например). В некоторых типах браузеров и ОС могут происходить неприятные искажения.
  Вот тогда и начинаются различные скрипты, исправляющие эти изъяны и практически сводящие на нет все преимущества CSS.
  В практике были случаи, когда идеальные с точки зрения HTML и CSS блоки вели себя неадекватно от некоторых типов активного содержимого.
  Есть ещё, конечно, проблема свободно позиционированных объектов, а точнее проблема не в них, а в понимании таких объектов браузерами Internet Explorer.
  Итак, понятно, что чистая блочная вёрстка не эффективна.
- Как на счёт табличной? Действительно, многие браузеры загружают таблицу как единое целое, замедляя таким образом работу сайта. Действительно, таблицы делают код излишне объемным. Действительно, табличная вёрстка в какой-то степени ограничивает фантазию дизайнера.
- Вот и закончен наш анализ. Он показал, что мы, без сомнения, можем создать шаблон только на блоках, или только на таблицах. Однако, нужно заметить, что и тот, и другой шаблоны не будут совершенными (есть исключения, разумеется).

- Самым логичным типом вёрстки – будет грамотный «симбиоз» таблиц и блоков, их дополнение друг друга. Именно такой подход позволит создавать действительно качественные сайты.

Что такое спрайт?

- Спрайт – одно изображение, в состав которого включено некоторое количество более мелких изображений.
- По структуре, спрайт можно сравнить с мозаикой или пазлом. С помощью специального атрибута, для каждого объекта выбирается часть этой картинки, которая и показывается.

  Конечно, Вы имеете полное право спросить: «Зачем создавать спрайт, если можно просто воспользоваться несколькими картинками, как это делается обычно?».
- Ответ будет довольно простым: спрайт подгружается один раз и при наведении курсора на элемент изменения происходят мгновенно, без подгрузки другой картинки. Это существенно улучшает его восприятие и уменьшает количество HTTP запросов, что в свою очередь ускоряет загрузку страницы. Само собой, у спрайтов есть ограничения. Наш объект должен быть фиксирован по ширине и /или высоте.

  Спрайты очень часто используются в меню, или других элементах, изменяющихся при наведении мыши. В практическом примере, мы будем делать именно меню для сайта.

Пример. Спрайт-меню. index.html

- Создадим вертикальное меню, состоящее из 5 элементов. Сделаем его, используя маркированный список. Создадим файлы index.html и style.css. В код веб-страницы запишем стандартное начало:

```
<html >
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
    <title>Спрайт-меню</title>
    <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
    <ul id="menu">
      <li><a href="#">Ссылка 1</a></li>
      <li><a href="#">Ссылка 2</a></li>
      <li><a href="#">Ссылка 3</a></li>
      <li><a href="#">Ссылка 4</a></li>
      <li><a href="#">Ссылка 5</a></li>
    </ul>
</body>
</html>
```

Пример. Спрайт-меню. style.css (1)

Перейдём к оформлению.

Сначала в style.css отменим оформление для всего списка:

```
ul {list-style:none;}
```

Потом зададим вид пункта меню, без наведения на него мыши:

```
#menu li a {
        padding:7px 10px;
        color:#666;
        text-decoration: none;
        display: block;
        font:13px Arial;
        font-weight:bold;
        background: url(menu.png);
        width:200px;
        margin:3px;}
```

Здесь – мы использовали изображение menu.png.

Это – и есть спрайт. Он выглядит следующим образом:

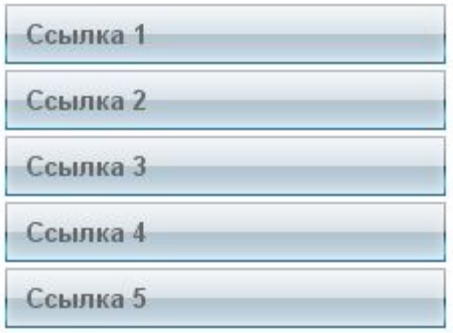
Пример. Спрайт-меню. style.css (2)

Теперь зададим стили для меню при наведении на него мыши и при клике на него:

```
#menu li a:hover {
        color: #fff;
        background: url(menu.png)
    0 -30px;
    }

    #menu li a:active {
        color: #fff;
        background: url(menu.png)
    0 -60px;
    }
```

- Заметим, что -30px и -60 px – указание, на сколько пикселей надо сместиться по спрайту.
  Если всё сделано правильно, то получится такое меню:



Бестабличная верстка – обтекания, или CSS Float

Обтекание картинки текстом использовалось с самого начала формирования стандартов верстки.

Позже – в CSS появился атрибут float, призванный обеспечить такую возможность. Всё казалось бы логичным, но это – не так, в чем Вы и убедитесь в этом примере.
Сам атрибут выглядит так:

float:left – обтекание по левому краю
float:right – обтекание по правому краю
float:none – нет обтекания (по умолчанию)

С использованием обтекания связано очень много искажений и проблем. Это, в первую очередь, связано с неинуитивностью самой концепции float в CSS. Тем не менее, атрибут помогает реализовать различные идеи, многие из которых, просто не имеют другого исполнения.

Сферы использования CSS Float

- Обтекание фиксированного объекта другими объектами
- Формирование колонок, заключенных в блоки
- Принудительное выравнивание объекта по левому или правому краю родителя
- Выделение блока (объекта) из основного потока (содержания)
- Создание частей сложных конструкций

Основные особенности работы с float

- Объект со стилем float позиционируется по левому/правому краю, вне общего потока и игнорируя другие установки
- Блоки с float игнорируются основным потоком. Например: при создании фона – их содержание не учитывается
- Для обтекаемых объектов необходимо задавать ширину (любым способом). Крайне желательно задавать высоту (см. предыдущий пункт)
- Разные браузеры по разному понимают простейшие операции с такими блоками. Особые проблемы возникают с устаревшими версиями IE
- Блоки с атрибутом Float выравниваются по верхней границе элемента - родителя
- Возможны проблемы с отображением ссылок внутри Float блоков.

Основные проблемы работы с float

- Основная проблема при реальной верстке с помощью обтеканий – то, что блок с таким стилем как бы сжимается, не обращая никакого внимания на содержание. Решить эту проблему можно задав высоту блока, или задав фон с помощью самой страницы (стиль для html,body).
  Для предотвращения наезда нижних элементов основного потока на блок с float, используют специальный атрибут – clear. Он имеет следующие варианты:
- clear:none – нет контроля обтекания
- clear:left – контроль обтекания по левому краю
- clear:right – контроль обтекания по правому краю
- clear:both – контроль обтекания по обоим краям

Атрибут clear чаще всего используется для создания «подвала».

Шаблон персонального сайта – основа дизайна float. index.html

```
<!-- Левая колонка -->
    <div id="leftcolumn">
    <h3>Мои друзья</h3>
    <div id="rcs"><a
href="http://www.css-school.ru/">Россиcая
школа CSS</a></div>
    <div id="gzweb"><a
href="http://www.gzweb.ru/">GZweb.ru</a></div>
    <div id="wallday"><a
href="http://www.wallpapersday.ru/">Wallpapers
Day</a></div>
        <hr />
          <h3>Интересные события</h3>
     <ul>
     <li><a href="/">Российская школа CSS
оживилась</a></li>
     <li><a href="/">У GZweb.ru новый
дизайн</a></li>
     <li><a href="/gall.html">Моя
галерея</a></li>
     </ul>
     <br /><br />
    </div>
    <!-- Конец левой колонки -->
```

```
<!-- Основное содержание
сайта -->
    <div id="rightcolumn">
      <h1>Мой дневник
появился</h1>
       <img
src="1.jpg"  alt="Мой
дневник появился" /><br />
        Рад ссобщить Вам,
что сегодня начал
работать мой собственый
дневник!!!
    ;)
<div id="undernew">Я
опубликовал этот
материал <b>12 октября
2008 года</b>. Категория -
<b>Новости
дневника</b>.</div>
    </div>
    <!-- Конец основного
содержания сайта -->
```

Шаблон персонального сайта – основа дизайна float. style.css

Без стилей, эти блоки располагались бы один под другим, но это можно исправить, задав в style.css

```
#leftcolumn {
    background:#e7e7e7;
    display: inline;
    color: #333;
    margin-right:10px;
    padding:10px;
    width: 246px;
    height:auto;
    [b] float: left;[/b]
    border:2px solid #ddd;
    }
```

```
#rightcolumn {
    [b] float: right;[/b]
    color: #333;
    padding:0;
    width: 680px;
    display: inline;
    position: relative;
    text-align:center;
    }
```

2 вариант. index.html

Попытаемся сделать простую структуру из двух колонок, но вместо

float: right;

для правой ставим

margin-left:***px;

Вот код для html файла:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
<title>Российская школа CSS. Использование Float.</title>
<link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<div id="left">Левая колонка</div>
<div id="right">Правая колонка</div>
</body>
</html>
```

2 вариант. style.css

```
html,body {
   width:1000px;
   margin:0 auto;
   padding-top:10px;
 }

#left {
   float:left;
   width:250px;
   background:#cbcbcb;
   border:3px solid #004167;
   height:300px;
   padding:10px;
 }
```

```
#right {
   width:690px;
   background:#ffd97e;
   border:3px solid #004167;
   height:300px;
   padding:10px;
   margin-left:276px;
 }
```

В Опере и подобных браузерах это будет выглядеть так

А в IE6 будет так

На рисунке ясно видно, что браузер добавил три пикселя между блоками, хотя по логике, их там не должно быть. Можно, конечно предположить, что это произошло из-за влияния границы и/или внутреннего отступа. Уберём их и посмотрим, что получится:

Проблема решена

Итак, простой манипуляцией со стилями исправить проблему не удалось. Зато, исправить её можно значительно проще.

Добавим в html такой код:

```
<![if ! IE]>
    <style>
    #right {
        margin-left: 280px;
    }
```

Этот код значит, что для всех браузеров, кроме IE будет приниматься отступ в 280 пикселей.

Соответственно в файл стилей пишем так:

```
margin-left:277px;
```

Вот и всё.

Проблема – решена.