Учебник CSS для начинающих.

Версия для печати раздела "Учебник CSS" сайта www.webremeslo.ru

Введение

Что такое CSS?

Если Вы уже прошли курс обучения по **учебнику HTML** (**http://webremeslo.ru/html/glava0.html**) или же хорошо знакомы с HTML почерпнув необходимые знания из других источников, то настало время взяться за изучение CSS.

CSS (Cascading Style Sheets) - Каскадные таблицы стилей - это свод стилевых описаний, тех или иных HTML тегов (далее элементов HTML), который может быть применён как к отдельному тегу - элементу, так и одновременно ко всем идентичным элементам на всех страницах сайта. CSS по сути своего рода дополнение к

HTML, которое значительно расширяет его возможности.

Ну и что? Спросите Вы.. Зачем мне этот пресловутый CSS? Я и HTML-ом в чистом виде неплохо обходился!

Приведу ряд доводов в пользу использования CSS:

HTML в чистом виде имеет весьма ограниченный набор инструментов не позволяющий решать те или иные дизайнерские и функциональные задумки веб-мастера. Ну вот хотя бы, к примеру, взять больной вопрос всех начинающих веб-ремесленников "Как убрать подчеркивание ссылки?" или "Как сделать чтобы наведя курсором на эту самую ссылку она меняла цвет и подсвечивалась?" с помощью одного HTML этого никак не сделать!! А сколько их еще таких "больных вопросов"? - тьма.. Тут то и приходит на помощь CSS, который решает большинство задач касающихся дизайна сайта.

Предположим, Вы написали сайт в нем 100 страниц.. хороший сайт, информативный, люди на него ходят.. И вдруг по каким либо причинам Вам понадобилось изменить его дизайн, ну не знаю, мода изменилась, Вы нашли более лучшее дизайнерское решение, заказчику пришлось не по душе.. да мало ли еще почему.. Сколько времени и сил у Вас уйдёт на то что бы полностью переделать все 100 страниц сайта? CSS предлагает разумное решение этой задачи. А что если один раз в отдельном файле полностью описать весь дизайн сайта? Допустим: все заголовки **<h1>** делать красным цветом, параграфы **<p>** писать курсивам, ссылки **<a>** не подчёркивать :) фон на всех страницах залить зелёным, и т. д. … а потом просто заставить эти 100 страниц HTML обращаться к файлу CSS и черпать из него нужную информацию? Теперь когда Вам вздумается, к примеру, перекрасить все заголовки из красного в зеленый, Вам ненужно

открывать все 100 страниц находить в них теги **<h1>** и указывать в каждом что ты теперь не красный а зелёный! Вам нужно всего лишь открыть файл описание и изменить в нем цвет элемента **<h1>** на зелёный и всё!! Все заголовки на всех страницах сайта как по взмаху волшебной палочки станут зелеными.

Ввиду того, что CSS позволяет выносить повторяющиеся стилевые описания одних и тех же элементов в один файл происходит значительная "разгрузка" документов HTML, а это экономия объема, трафика, времени, денег.. HTML код становится лёгким, удобным для чтения и редакции.

Ну как? Заинтриговал? Если да то рекомендую перейти к непосредственному изучению CSS. В главах этого учебника Вы научитесь внедрять каскадные таблицы стилей на страницы Вашего сайта, познакомитесь с основными стилевыми свойствами элементов на

примере создания сайта с использованием CSS, вникните в тонкости и хитрости дела. Если Вы уже знакомы с каскадными таблицами стилей и Вас интересует исключительно справочная информация, то предлагаю заглянуть в **справочник CSS** (**http://webremeslo.ru/spravka/spravka4.html**) где собранны и кратко описаны свойства CSS и их возможные значения.

Глава 1

Внедрение CSS в HTML документ.

В этой главе речь пойдет о том, как внедрить CSS в документ HTML, то есть связать стилевое описание элемента непосредственно с самим элементом, каким либо HTML тегом.

Осуществить данную задачу можно тремя способами:

Написать стилевое описание непосредственно в самом элементе. Такой способ хорош лишь в том случае если таковой элемент один единственный в HTML документе который нуждается в отдельном стилевом описании.

Написать стилевое описание для всех идентичных элементов HTML документа. Такой способ оправдывает себя, если стиль страницы принципиально отличается от общего дизайна сайта (группы взаимосвязанных страниц).

Вынести стилевое описание элементов HTML в отдельный файл CSS. Это позволит управлять дизайном всего сайта целиком, каждой страницей сайта в которой указанно обращение к CSS файлу. Этот способ является наиболее эффективным использованием таблицы каскадных стилей.

Давайте более подробно рассмотрим каждый вариант, а заодно познакомимся с правилами синтаксиса написания CSS.

Атрибут style.

Практически каждый HTML тег имеет атрибут **style**, который говорит о том, что к этому тегу применяется некое стилевое описание.

Пишется так:

<p **style=""**> это параграф с индивидуальным стилем </p>

Всё что будет написано между кавычками атрибута **style** и будет

являться стилевым описанием для данного элемента, в данном случае элемента ** **

Ну например:

<p **style="color: #ff0000; font-size:12px"**> это параграф с индивидуальным стилем

В данном случае мы указали, что этот параграф должен отображаться красным цветом и иметь размер шрифта в 12 пикселей. В последующих главах я подробно расскажу о том что написано в кавычках , сейчас же речь идет о том как применить CSS к какому либо HTML тегу.

По такому же принципу можно указать индивидуальный стиль практически для каждого HTML элемента.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
```

```
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Атрибут style</title>
  </head>
  <body style="background-color:
#c5ffa0">
    <h1 style="color: #0000ff; font-
size:18px">Всё о слонах</h1>
    <p style="color: #ff0000; font-
size:14px">На этом сайте Вы найдёте
любую информацию о слонах.</p>
    <h2 style="color: #0000ff; font-
size:16px">Купить слона</h2>
    <p style="color: #ff0000; font-
size:14px">У нас Вы можете по
выгодным ценам приобрести лучших
слонов!!</p>
    <h2 style="color: #0000ff; font-
size:16px">Взять слона на
прокат</h2>
    <p style="color: #ff0000; font-
size:14px">Только у нас Вы можете
взять любых слонов на прокат!!</p>
```

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

Но еще раз повторюсь, такой способ внедрения CSS хорош лишь в том случае, если требуется задать определенный стиль малому числу HTML элементов.

Тег <style>

Для того, что бы описать необходимые элементы одновременно на всей странице в заголовок HTML документа внедряют тег **<style></style>** (не путайте с одноименным атрибутом) в котором и происходит описание нужных нам элементов.

Взгляните на пример, ниже к нему будут комментарии.

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
```

```
  <head>
    <title>Тег style</title>
    <style type="text/css">
      body {background-color:
#c5ffa0}
      h1 {color: #0000ff; font-
size:18px}
      h2 {color: #0000ff; font-
size:16px}
      p {color: #ff0000; font-size:14px}
    </style>
  </head>
  <body>
    <h1>Всё о слонах</h1>
    <p>На этом сайте Вы найдёте
любую информацию о слонах.</p>
    <h2>Купить слона</h2>
    <p>У нас Вы можете по выгодным
ценам приобрести лучших
слонов!!</p>
    <h2>Взять слона на прокат</h2>
    <p>Только у нас Вы можете взять
любых слонов на прокат!!</p>
  </body>
</html>
```

Как видно из примера мы добились точно такого же результата что и в первом случае только теперь мы не прописываем каждому элементу стиль индивидуально, а вынесли его в "голову" документа тем самым, указав, что все заголовки **<h1>,<h2>** - будут синими, а параграфы **<p>** - красными. Представьте, как мы облегчили бы себе работу, будь на странице сотня таких параграфов и штук пятнадцать заголовков, да и сам документ стал меньше весить за счет "удаления" всех повторяющихся стилевых описаний для каждого отдельно взятого элемента.

Теперь обещанные комментарии:

Тег **<style>** принято внедрять в заголовок HTML документа между тегами **<head></head>**.

Атрибут тега **<style> type** - сообщает браузеру, какой синтаксис использовать для правильной интерпретации стилей. Для

правильной интерпретации браузерами CSS значение **type** должно равняться **text/css**.

Внутри тега **<style></style>** идет непосредственное объявление стилей тех или иных HTML элементов согласно следующему синтаксису:

Если в блоке объявления стилей указывается несколько свойств элемента, то они между собой разделяются точкой с запятой.

CSS в отдельном внешнем файле.

Долго ли коротко ли, подошли мы к главному, на мой взгляд, достоинству CSS, а именно возможности выносить все сведения касающиеся дизайна сайта в отдельный внешний файл.

Итак, открываем блокнот (или другой редактор) и пишем в нем следующий текст:

```
body {background-color: #c5ffa0}
a {color:#000060; font-weight: bold;}
a:hover {color:#ff0000; font-weight: bold;
text-decoration:none}
h1 {color: #0000ff; font-size:18px}
h2 {color: #ff00ff; font-size:16px}
p {color: #600000; font-size:14px}
```

О том, что это такое странное мы написали, постараюсь подробно рассказать в последующих главах этого учебника.

Далее сохраняем этот небольшой файлик с расширением ***.css** (обычно файл со стилями называют **style.css**).

Все! файл со стилевым описанием создан! Теперь осталось совсем чуть чуть, а именно заставить нужные страницы нашего сайта черпать информацию с этого файла.

Делается это с помощью тега **<link>** (связь). Тег **<link>** многоцелевой и служит для "связывания" HTML документа с дополнительными внешними файлами, обеспечивающими его должную работу. Тег **<link>** является своего рода ссылкой, только предназначенной не для пользователей, а для программ обозревателей (браузеров). Так как **<link>** несёт в себе исключительно служебную информацию он располагается в заголовке HTML документа между тегами **<head> </head>** и не выводится браузерами на экран.

Тег **<link>** имеет атрибуты:

href - Путь к файлу.
rel - Определяет отношения между текущим документом и файлом, на который делается ссылка.

shortcut icon - Определяет, что подключаемый файл является иконкой.

stylesheet - Определяет, что подключаемый файл содержит таблицу стилей.

application/rss+xml - Файл в формате XML для описания ленты новостей.

type - MIME тип данных подключаемого файла.

Так как мы подключаем в качестве внешнего файла каскадную таблицу стилей, то наша служебная ссылка приобретает следующий вид:

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

Повторюсь, что бы уж точно развеять возможные недопонимания. Атрибуту **rel** присваиваем значение **stylesheet** так как подключаем в качестве внешнего файла каскадную таблицу стилей, указываем путь к

файлу css (в этом примере файл называется **mystyle.css** и лежит рядом с документом HTML в котором прописывается данная ссылка) так же указываем, что данный файл текстовый и содержит в себе стилевое описание **type="text/css"**

Теперь вставляем эту строчку в заголовки страниц нашего сайта и наслаждаемся результатом..

Пример:

Файл mystyle.css

```
body {background-color: #c5ffa0}
a {color:#000060; font-weight: bold;}
a:hover {color:#ff0000; font-weight:
bold; text-decoration:none}
h1 {color: #0000ff; font-size:18px}
h2 {color: #ff00ff; font-size:16px}
p {color: #600000; font-size:14px}
```

Файл index.html

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
  <head>
    <title>каскадная таблица стилей</title>
    <link rel="stylesheet" href="mystyle.css" type="text/css">
  </head>
  <body>
    <h2>Меню:</h2>
    <a href="index.html">Всё о слонах.</a>
    <a href="elephant.html">Купить слона.</a>
    <a href="elephant1.html">Взять слона на прокат.</a>
    <hr>
    <h1>Всё о слонах</h1>
    <p>На этом сайте Вы найдёте любую информацию о слонах.</p>
  </body>
</html>
```

Файл elephant.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
```

```
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>каскадная таблица стилей</title>
    <link rel="stylesheet" href="mystyle.css" type="text/css">
  </head>
  <body>
    <h2>Меню:</h2>
    <a href="index.html">Всё о слонах.</a>
    <a href="elephant.html">Купить слона.</a>
    <a href="elephant1.html">Взять слона на прокат.</a>
    <hr>
    <h1>Купить слона</h1>
    <p>У нас Вы можете по выгодным ценам приобрести лучших слонов!!</p>
  </body>
</html>
```

Файл elephant1.html

```
<!DOCTYPE HTML PUBLIC "-
```

```
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>каскадная таблица
стилей</title>
    <link rel="stylesheet"
href="mystyle.css" type="text/css">
  </head>
  <body>
    <h2>Меню:</h2>
    <a href="index.html">Всё о
слонах.</a>
    <a href="elephant.html">Купить
слона.</a>
    <a href="elephant1.html">Взять
слона на прокат.</a>
    <hr>
    <h1>Взять слона на прокат</h1>
    <p>Только у нас Вы можете взять
любых слонов на прокат!!</p>
  </body>
</html>
```

В примере выше, "сайт о слонах", на данный момент, имеется три страницы, каждая из которых связана с одним единственным внешним css файлом - mystyle.css. Таким образом, мы значительно его "разгрузили" и сделали дизайн всего сайта "мобильным". Представьте сколько б килобайт мы выиграли, будь на этом сайте сотня полноценных страниц!? А также, сколько б времени сэкономили, если бы нам понадобилось изменить что-либо в его дизайне!?

О том как присвоить какой либо группе идентичных элементов стиль отличающийся от основного стиля данного элемента, сделать отдельный класс элементов, читайте в главе **Классы и идентификаторы.**

Полезные советы:

В этой главе мы рассмотрели три способа внедрения CSS в HTML документ. Какой же лучше использовать?

Используйте атрибут **style** для какого либо элемента если этот элемент с отличным от других элементов стилем один единственный на всём сайте.

Используйте тег **<style>** со стилевым описанием, в том случае, если страница должна иметь индивидуальный дизайн в корни отличный от других страниц сайта.

В большинстве случаев разумно выносить каскадную таблицу стилей в отдельный css файл.

Глава 2

Свойства текста.

В этой главе пойдет речь о том, что можно сделать с текстом, применяя к элементам HTML содержащие в себе некий текст те или иные свойства CSS.

Ну поехали..

Выравнивание текста.

Если Вы помните, из курса HTML, для того что бы выровнять текст, например по центру экрана, мы применяли к тегу содержащему в себе текст атрибут **align**(выравнивание) и одно из его возможных значений **center**(по центру)

Запись имела такой вид:

** текст по центру **

В CSS данную задачу берет на себя свойство **text-align**, которое выравнивает текстовое содержание

относительно элемента родителя (например, блока **div**) или же окна браузера.

text-align (так же как и htmlловский атрибут **align**) имеет следующие значения:

left - Выровнять текст по левому краю элемента (по умолчанию).

right - Выровнять текст по правому краю.

center - Выровнять текст по центру.

justify - Выровнять текст по обоим краям.

Теперь для того чтобы выровнять текст того же параграфа по центру следует писать так:

** текст по центру **

- это в этом случае если мы, с помощью атрибута **style**, внедряем CSS непосредственно в HTML тег.

А вот в примере ниже используется тег **<style>** в заголовке документа:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Выравнивание текста</title>
    <style type="text/css">
      h1 { text-align: center }
      p { text-align: justify }
    </style>
  </head>
  <body>
    <h1>Всё о слонах</h1>
    <p>Слон - самое крупное
млекопитающее на нашей планете!
Самый большой слон из когда-либо
живущих на Земле был
зарегистрирован в Анголе в 1956 году.
Этот самец весил около 12 тон, а в
высоту достигал 4,2 метра, что на
метр выше среднего Африканского
слона.</p>
```

```
    <p>Слоны являются королевским символом Азиатской культуры и известны своей отличной памятью и высоким интеллектом. Аристотель однажды сказал, что слон - "животное, которое превосходит всех других в остроумии и интеллекте".</p>
  </body>
</html>
```

Оформление текста.

Свойство **text-decoration** позволяет декорировать текст, присвоив ему одно или несколько значений из ниже представленных вариантов оформления текста.

Возможные значения:

blink - Текст будет мигать.

line-through - Делает текст перечеркнутым.

overline - Надчёркивание текста.

underline - Подчеркивание текста.

none - Текст без оформления.

Пишется так:

Ссылка без подчёркивания

Пример:

Файл mystyle.css

```
h1 {text-align: center}
h3 {text-align: left; text-decoration: underline}
a {text-decoration: underline}
a:hover {text-decoration:none}
p {text-align: justify}
```

Файл index.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Оформление текста</title>
    <link rel="stylesheet" href="mystyle.css" type="text/css">
```

```
  </head>
  <body>
    <h3>Меню:</h3>
    <a href="index.html">Всё о
слонах.</a><br>
    <a href="elephant.html">Купить
слона.</a>
    <hr>
    <h1>Всё о слонах</h1>
    <p>Слон - самое крупное … …
…</p>
    <p>Слоны являются … … …</p>
  </body>
</html>
```

Обратите внимание на внешний файл CSS в нем мы "декорировали" ссылку элемент **<a>**, причем делали это два раза первый раз **a {text-decoration: underline}** сделали её подчёркнутой, хотя можно было этого и не делать, так как тег **<a>** подчёркнут по умолчанию, а второй раз использовали так называемый псевдокласс **hover** и запретили

подчеркивание **a:hover {text-decoration:none}**

Данный псевдокласс указывает на то, что применять к нему стилевое описание стоит лишь в том случае если пользователь навел курсор на этот элемент. Так если в примере навести курсор на одну из ссылок в меню то подчеркивание исчезнет, что создаёт определенный динамический эффект.. меню становиться "живым".

Впрочем, мы немного забежали вперёд.. о псевдоклассах речь пойдёт в отдельной главе.

Отступ первой строки.

Свойство **text-indent** - задаёт отступ первой строки в текстовом блоке с левой стороны, проще говоря делает "красную строку".

Расстояние от левого края окна браузера или же элемента родителя (блока в который помещен блок с текстом) может быть заданно в

процентах от ширины окна браузера или же единицах измерения принятых в CSS.

В примере ниже расстояние отступа от левого края задаётся в пикселях **(px)**:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Красная строка</title>
    <style type="text/css">
      h1 {text-align: center}
      p {text-align: justify; text-indent:
20px}
    </style>
  </head>
  <body>
    <h1>Всё о слонах</h1>
    <p>Слон - самое крупное
млекопитающее на нашей планете!
Самый большой слон из когда-либо
живущих на Земле был
```

```
зарегистрирован в Анголе в 1956 году. Этот самец весил около 12 тон, а в высоту достигал 4,2 метра, что на метр выше среднего Африканского слона.</p>
    <p>Слоны являются королевским символом Азиатской культуры и известны своей отличной памятью и высоким интеллектом. Аристотель однажды сказал, что слон - "животное, которое превосходит всех других в остроумии и интеллекте".</p>
  </body>
</html>
```

Трансформация текста

Своийство **text-transform** трансформирует символы в указанном текстовом блоке, делая их заглавными или прописными по одному из правил в зависимости от присужденного значения данному свойству.

Значения:

none - Текст отображается без каких-либо изменений.(по умолчанию)

capitalize - Каждое слово в тексте отображается с заглавного символа.

lowercase - Все символы преобразуются в нижний регистр.

uppercase - Все символы преобразуются в верхний регистр.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Трансформация текста</title>
  </head>
  <body>
    <p style="text-transform:
capitalize">союз советских
социалистических республик</p>
    <p style="text-transform:
```

```
lowercase">CCCP cccp</p>
    <p style="text-transform:
uppercase">cccp CCCP</p>
  </body>
</html>
```

Вертикальное выравнивание

Вертикальное выравнивание текста в строке устанавливает свойство **vertical-align**

Возможные значения свойства **vertical-align:**

baseline - Выравнивает базовую линию элемента по базовой линии родителя.

bottom - Выравнивает элемент по нижней части строки.

middle - Выравнивает средину элемента по базовой линии родителя и прибавляет половину высоты родительского элемента.

sub - Нижний индекс (размер шрифта не меняется).

super - Верхний индекс (размер шрифта не меняется).

text-bottom - Нижняя граница элемента выравнивается по нижнему краю строки.

text-top - Верхняя граница элемента выравнивается по верхнему краю строки.

top - Выравнивает элемент по верхней части строки.

Базовая линия - это линия, на которой располагаются "сидят" символы в текстовой строке, Например буква "А" сидит прямо на этой линии, а вот строчная буква "у" сидит на ней же, но свесив ноги..

Взгляните на рисунок с разметкой строки:

Так же вертикальное выравнивание элемента относительно строки может выражаться в процентах, пикселях или любых других принятых в CSS единицах измерения, причем эти единицы могут принимать как положительные, так и отрицательные значения.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Вертикальное выравнивание
текста</title>
  </head>
  <body>
    <font size="+3">А и Б </font>
    <span style="vertical-align:
+5px">сидели на трубе </span>
    <span style="vertical-align:
bottom">А упало </span>
    <span style="vertical-align: top">Б
```

```
пропало.. </span>
    <span style="vertical-align:
50%">что осталось на трубе?</span>
    <hr>
    формула воды: H<span
style="vertical-align: sub">2</span>O
    <hr>
    <span>н</span>
    <span style="vertical-align: -
10px">а</span>
    <span style="vertical-align: -
20px">и</span>
    <span style="vertical-align: -
30px">с</span>
    <span style="vertical-align: -
40px">к</span>
    <span style="vertical-align: -
50px">о</span>
    <span style="vertical-align: -
60px">с</span>
    <span style="vertical-align: -
70px">о</span>
    <span style="vertical-align: -
80px">к</span>
  </body>
</html>
```

Пробелы и перенос строки.

Набранный текст, в каком либо текстовом редакторе браузерами по умолчанию выводится на экран в виде сплошного текста, где переносы строк расставляются автоматически, а так же убираются лишние (более одного) пробелы между символами.

Свойство **white-space** имитирует работу тега **<pre>**, определяя показывать или нет пробелы между символов, если таковых больше чем один, а так же разрешает или запрещает перенос строки.

Может иметь следующие значения:

normal - текст выводится как обычно (лишние пробелы убираются), переносы строк определяются автоматически. (по умолчанию)

nowrap - запрещает автоматический перенос строки.

pre - показывает текст в том виде в котором он был набран. пробелы и переносы строки не удаляются.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Пробелы и перенос
строки</title>
  </head>
  <body>
    <p style="white-space: pre">
Слон.

Дали туфельки слону.
Взял он туфельку одну
И сказал: - Нужны пошире,
И не две, а все четыре!

С. Я. Маршак.
    </p>
    <hr>
```

```
    <p style="white-space: nowrap">
Это длинный предлинный текст,
который вряд ли полностью
поместится в одной строчке, по
умолчанию в нужном месте, браузер
перенес бы его на следующую строку,
однако мы принудительно запретили
это делать, с помощью значения
nowrap свойства white-space. Так что
теперь, по всей вероятности, в окне
браузера появиться горизонтальная
полоса прокрутки.. и зачем,
спрашивается, мы это сделали?
    </p>
  </body>
</html>
```

При использовании nowrap текст в нужном месте можно переносить на следующую строку используя тег **
**

Расстояние между словами.

Свойство **word-spacing** задаёт расстояние между словами (группами символов не разделенными пробелом) в строке.

Значения:

normal - Нормальное расстояние. (по умолчанию)

px - Расстояние задаётся в пикселях или любых других единицах измерения принятых в CSS.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Расстояние между
словами</title>
  </head>
  <body>
    <p align="left" style="word-spacing:
10px">Расстояние между словами
равно десяти пикселям</p>
    <p align="left" style="word-spacing:
-10px">Расстояние между словами
может иметь отрицательное
значение</p>
```

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

Межсимвольное расстояние.

А вот свойство **letter-spacing** определяет расстояние между символами в тексте и так же как и может **word-spacing** быть задано следующими значениями:

normal - Нормальное расстояние. (по умолчанию)

px - Расстояние задаётся в пикселях или любых других единицах измерения принятых в CSS.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Расстояние между
символами</title>
  </head>
```

```
  <body>
    <p style="letter-spacing:
5px">Расстояние между буковками
равно пяти пикселям</p>
    <p style="letter-spacing: -3px">А
здесь буквы, из за отрицательного
значения, будут наплывать друг на
друга</p>
  </body>
</html>
```

Интерлиньяж

Интерлиньяж - это расстояние между строками текста.

Расстояние между строками текста можно задать используя свойство **line-height**, сделать это можно следующими способами:

normal - Норма (по умолчанию).

% - Проценты. за сто процентов берется высота шрифта

0.5 - Множитель. Может быть использовано любое число больше ноля. Так, например множитель 0.5

будет равняться половинному межстрочному расстоянию, а 2 - двойному.

px - Пиксели и любые другие единицы измерения, принятые в CSS.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Интерлиньяж</title>
  </head>
  <body>
    <div style="line-height: 150%">
строка первая <br> строка вторая <br>
строка третья <br> строка четвертая
<br> строка пятая
    </div>
    <hr>
    <div style="line-height: 0.5">
строка первая <br> строка вторая <br>
строка третья <br> строка четвертая
```

```
<br> строка пятая
      </div>
      <hr>
      <div style="line-height: 25px">
строка первая <br> строка вторая <br>
строка третья <br> строка четвертая
<br> строка пятая
      </div>
   </body>
</html>
```

Полезные советы:

. При декорировании текста, свойство - **text-decoration**, будьте благоразумны используя подчеркивание текста это может ввести в заблуждение посетителя страницы, он может подумать, что данный текст является ссылкой.

. А Вам точно нужно использовать значение **nowrap** свойства **white-space**, запрет переноса строки? Появление горизонтальной полосы прокрутки мало кого вдохновляет..

. Используя псевдокласс hover в сочетании с различными элементами

и их возможными CSS свойствами можно добиться весьма интересных эффектов.

Глава 3

Свойства шрифта.

В предыдущей главе мы рассмотрели свойства текста, в этой поговорим о том что можно сделать с шрифтом используя инструменты CSS.

Стиль шрифта

Свойство **font-style**, в зависимости от выбранного значения, определяет стиль шрифта.

Шрифт может иметь следующие стили:

normal - обычный (по умолчанию)

italic - курсив

oblique - наклонный

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
  <head>
    <title>Стиль шрифта</title>
  </head>
  <body>
    <p style="font-style: italic">это курсив</p>
    <p style="font-style: oblique">а это наклонный текст</p>
    <p style="font-style: normal">И чем спрашивается, они отличаются?</p>
  </body>
</html>
```

Чем отличается курсив от наклонного текста? Курсив - это своего рода шрифт взятый из библиотеки шрифтов, а наклонный текст - это результат работы алгоритма, где каждый символ слегка наклоняется в правую сторону.

Начертание шрифта

Весьма интересное свойство шрифта **font-variant** позволяет делать

строчные буквы заглавными и уменьшенными.

Значения:

normal - нормальный (по умолчанию)

small-caps - все буквы заглавные и уменьшенные

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Все буквы заглавные</title>
  </head>
  <body>
    <p style="font-variant: small-
caps">Купи слона!!</p>
  </body>
</html>
```

Размер шрифта

Свойство CSS **font-size** - определяет размер шрифта.

Размер шрифта может быть задан в процентах или пикселях и любых других допустимых единицах измерения CSS, а так же абсолютным или относительным значением.

значения абсолютного размера шрифта:

xx-small - очень очень маленький

x-small - очень маленький

small - маленький

medium - средний

large - большой

x-large - очень большой

xx-large - очень очень большой

значения относительного размера шрифта:

larger - больше чем размер шрифта родительского элемента

smaller - меньше чем размер шрифта родительского элемента

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Размер шрифта</title>
  </head>
  <body>
    <div style="font-size: 18px;
background-color: #ecfef2; border: 5px
double #245404">
      <p>Размер шрифта
родительского элемента (блока DIV)
равен 18 пикселям</p>
      <p style="font-size: larger">Этот
шрифт будет крупнее относительно
элемента родителя</p>
      <p style="font-size: smaller">А
этот шрифт будет мельче
относительно элемента родителя</p>
    </div>
```

```
    <p style="font-size: 14px;">В блоке ниже размер штифта элемента родителя огромен (60 пунктов), однако дочерние параграфы расположенные в нём имеют собственное абсолютное значение шрифта и к размеру шрифта элемента родителя никак не привязаны.</p>
    <div style="font-size: 60pt; background-color: #ecfef2; border: 5px double #245404">
        <p style="font-size: xx-small">xx-small - очень очень маленький</p>
        <p style="font-size: x-small">x-small - очень маленький </p>
        <p style="font-size: small">small - маленький </p>
        <p style="font-size: medium">medium - средний</p>
        <p style="font-size: large">large - большой</p>
        <p style="font-size: x-large">x-large - очень большой</p>
        <p style="font-size: xx-large">xx-large - очень очень большой</p>
    </div>
```

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

Жирность шрифта

Свойство **font-weight** - определяет жирность шрифта. Насыщенность шрифта может быть задана относительно шрифта элемента родителя с помощью следующих значений:

normal - обычный шрифт

bold - полужирный шрифт

bolder - жирный шрифт

lighter - тонкий шрифт

А также выражается в условном числовом значении от 100 до 900 с шагом 100 (100, 200, 300... 900) где значение 100 тонкий шрифт, а 900 - сверх жирный.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
```

```
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Жирность шрифта</title>
  </head>
  <body>
    <div style="font-size: 18pt">
      <p style="font-weight:
bolder">жирный шрифт</p>
      <p style="font-weight:
lighter">тонкий шрифт</p>
      <p style="font-weight:
600">жирность шрифта равна 600</p>
    </div>
  </body>
</html>
```

Семейство шрифта

Атрибут CSS **font-family** - указывает одно, два или три имени шрифта из библиотеки шрифтов.

Возможность указывать до трёх имен шрифтов через запятую используется разработчиками во

избежание возможных проблем связанных с отсутствием, по тем или иным причинам, указанных имен в библиотеке шрифтов на компьютере пользователя.

Так например запись в стилевом описании **P {font-family: Times New Roman, Arial, Verdana;}** - будет указывать браузеру пользователя, что данный параграф следует писать с помощью шрифта Times New Roman, а если его по каким то мифическим причинам не окажется в его библиотеке шрифтов то следует использовать шрифт Arial, и уж если и его нет, тогда писать шрифтом Verdana.

Если же браузер не найдёт в библиотеке шрифтов пользователя ни одного шрифта из указанных то он будет использовать тот шрифт который указан в его настройках "по умолчанию"

Однако также можно указать браузеру не только какой то конкретный шрифт, но и обозначить предпочтительное семейство шрифтов из перечисленных ниже возможных

serif - шрифты с засечками

sans-serif - рубленые шрифты

cursive - курсивные шрифты

fantasy - декоративные шрифты

monospace - моноширинные шрифты

Например, если в файле CSS написать **P {font-family: Times New Roman, sans-serif;}** то это будет значить что если вдруг не окажится шрифта с именем Times New Roman, то следует использовать любой (или определённый в настройках браузера) доступный шрифт из семейства **sans-serif** - рубленых шрифтов.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Семейство шрифта</title>
    <style type="text/css">
      h3 {font-family: Times New
Roman, Verdana, sans-serif;}
      p {font-family: Monotype
Corsiva, Verdana;}
      span {font-family: Comic Sans
MS;}
    </style>
  </head>
  <body>
    <h3>Купи слона</h3>
    <p>У нас Вы можете по
<span>выгодным ценам</span>
приобрести лучших слонов!!</p>
  </body>
</html>
```

Праметры шрифта

Вы наверняка обратили внимание на тот факт, что все свойства CSS предназначенные для работы со шрифтом начинаются с английского слова **font** (собственно шрифт).. **font**-family, **font**-size и т.д..

Так вот это неспроста.. дело в том, что все эти свойства являются "дочерними" базового атрибута CSS **font** в довершении главы о нем собственно и пойдёт речь.

Речь будет недолгой..)) Итак **font** - (шрифт), являясь базовым атрибутом, может определять одновременно сразу несколько параметров шрифта принимая те или иные значения от следующих атрибутов:

font-style

font-variant

font-weight

font-size

font-family

Для более детального ознакомления с возможными параметрами смотрите каждый атрибут отдельно.

Предположим нам необходимо написать стилевое описание шрифта для тега **** и по нашей задумке шрифт для данного тега должен быть: курсивом, жирным, иметь размер 20 пикселей и использовать шрифт Arial

Все это можно осуществить, написав следующие:

```
span{
  font-style: italic;
  font-weight: bolder;
  font-size: 20px;
  font-family: Arial
}
```

А можно обойтись всего одной строчкой используя базовый атрибут **font**.

Вот так:

```
span{font: italic bold 20px Arial}
```

Мы просто перечислили нужные нам параметры "марки" **font**. Согласитесь, по-моему, так писать гораздо проще, к тому же код становится более компактным и лёгким для чтения как браузером, так и человеком.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Параметры шрифта</title>
    <style type="text/css">
      span{font: italic bold 20px Arial}
    </style>
  </head>
  <body>
    <span>Этот текст написан
курсивом, он жирный, имеет размер 20
пикселей и использует шрифт
Arial</span>
  </body>
</html>
```

Полезные советы:

Используйте базовый атрибут **font** если необходимо применить к шрифту элемента более одного свойства.

Помните что текст, прежде всего, должен быть удобным для чтения и только потом красивым и стильным, а по сему советую избегать следующих вещей:

Не стоит использовать огромные и малюсенькие шрифты, это напрягает, ищите золотую середину. Так же не стоит злоупотреблять декоративными шрифтами, конечно, все эти закорючки весьма красивы, но читать длинный текст с таким шрифтом невыносимо. Выделяйте жирным только те фразы и выражения, на которых хотели бы произвести акцент, а курсивом "особые места" в тексте, к примеру, цитаты или афоризмы.

По мнению психологов, в длинном тексте должно использоваться не менее двух шрифтов, но и не более

четырёх.. Причем выделение текста "особым" шрифтом должно иметь систематический характер.. Например все заголовки одним шрифтом, "основной текст" вторым, и "особый текст" третьим.

Глава 4

Цвет и фон.

В этой главе мы поговорим о том, как с помощью CSS присвоить цвет элементу и его фону, а так же о том, как использовать рисунок в качестве фона элемента и управлять его положением.

Перед тем как перейти непосредственно к обучению, проведу краткий экскурс на тему: "Цвета в Интернете"

Цвет в CSS может быть задан тремя методами:

Именным значением, например: **red** - красный.

Значением цвета **RGB**, например: **RGB(255,0,0)** - опять таки красный.

Шестнадцатеричным значением цвета **RGB**, например: **#ff0000** - всё тот же красный.

С именным значением цвета всё понятно **black** - черный, **green** - зелёный, **olive** - оливковый и т.д. (полную палитру базовых красок, т.е. цветов для которых зарезервированы именные значения, **смотрите здесь: (http://www.webremeslo.ru/spravka/spravka1.html)**)

Однако по понятным причинам не для всех оттенков цветов зарезервировано индивидуальное имя. Когда возникает необходимость в использовании какого либо "нестандартного" цвета необходимо определить его значение **RGB, (Red, Green , Blue)** сочетание красного, зеленого и синего цвета в числовом выражении. Каждый оттенок из основных цветов в системе **RGB** может выражаться в числе от **0** до **255**.

Например, черный цвет будет иметь значение **0,0,0** то есть отсутствие всякого

цвета.. белый - значение **255,255,255** теоретически если смешать основные цвета должен получится белый, а вот например классический синий цвет имеет значение **0,0,255** то есть на "мольберте" присутствует только синий. На рисунке наглядно показано, что происходит с красками если их смешать, так смешивая оттенки основных цветов можно добиться любого цвета из видимого спектра.

Однако в большинстве случаев "веб краски" имеют шестнадцатеричное выражение десятичного значения RGB.

В шестнадцатеричном исчислении цифры от 10 до 15 заменены латинскими буквами и числовой ряд приобретает следующий вид:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Числа больше 15ти в шестнадцатеричной системе образуются путём объединения двух и

более чисел в одно. Так, например, числу **255** в десятичной системе соответствует число **FF** в шестнадцатеричной системе.

Значит, для того чтобы выразить нужный оттенок в шестнадцатеричном виде, нам понадобиться три пары чисел, где первая пара - значение красного цвета, вторая пара значение зелёного и третья пара синего цвета. Так, например, тот же классический синий в шестнадцатеричном выражении будет выглядеть так: **#0000FF**. Знак решётки перед числом ставится для указания того что данное число является шестнадцатеричным, например в числе **#808000** нет латинских букв однако со знаком решётки понятно что оно шестнадцатеричное и выражает собой оливковый цвет.

И еще.. выражая цвет в шестнадцатеричном виде можно обойтись тремя числами, которые затем будут дублироваться, например

запись **#DAF** будет сокращённой формой **#DDAAFF**.

Фух.. затянул я с водной частью давайте же наконец учить CSS..

Цвет элемента.

Для того, что бы перекрасить текст, какого либо, элемента в нужный нам цвет необходимо воспользоваться свойством **color** и присвоив ему нужное значение - собственно цвет.

Как уже сказано выше цвет в CSS может быть задан следующими методами:

#ff0000 - шестнадцатеричное значение цвета RGB.

red - именное значение цвета.

RGB(255,0,0) - значение цвета RGB.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
```

```
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Цвет элемента</title>
  </head>
  <body>
    <div style="color: red">Блок 1</div>
    <div style="color: #ff0000">Блок 2</div>
    <div style="color: RGB(255, 0, 0)">Блок 3</div>
  </body>
</html>
```

Цвет фона элемента.

А вот свойство **background-color** - определяет цвет фона элемента.

Цвет фона может иметь следующие значения:

#ff0000 - шестнадцатеричное значение цвета RGB.

red - именное значение цвета.

RGB(255,0,0) - значение цвета RGB.

transparent - прозрачный фон. (по умолчанию)

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Цвет фона элемента</title>
  </head>
  <body style="background-color:
#fffacd">
    <div style="background-color:
white">Белый блок</div>
    <div style="background-color:
#0000ff">Синий блок</div>
    <div style="background-color:
RGB(255,0,0)">Красный блок</div>
    <div style="background-color:
transparent">Прозрачный блок</div>
  </body>
</html>
```

Фоновое изображение.

Для любого элемента можно присвоить фоновое изображение с помощью CSS свойства: **background-image**.

Возможные значения **background-image:**

url - путь к файлу с изображением.

none - изображение отсутствует. (по умолчанию)

Для того чтобы сделать некую картинку фоном для элемента необходимо указать к ней путь согласно следующего синтаксиса url(путь к файлу/имя файла). Путь к файлу указывается в том случае, если рисунок находится в другой папке.

В примере ниже в качестве основного фона (элемент **body**) используется одно графическое изображение, а для блока **div** другое, возможность использования различных фоновых изображений для разных элементов страницы

позволяет решать практически любые дизайнерские задумки.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Фоновое изображение</title>
    <style type="text/css">
      body{
        background-image:
url(fon.jpg);
      }
      div{
        background-image:
url(fon1.gif);
        border: 5px double #245404;
        height: 250px;
      }
      p{
        text-align: center;
        color: #008040;
        font: bold 24px Arial;
```

```
        }
    </style>
  </head>
  <body>
    <p>Страница с фоновым
изображением</p>
    <div><p>Блок с фоновым
изображением</p></div>
  </body>
</html>
```

Если рисунок не заполняет собой весь фон элемента, то он выкладывается "плиткой".

Фиксация фонового изображения.

Если на странице или в каком либо блоке присутствует полоса прокрутки, то фоновое изображение будет прокручиваться вместе с остальным содержанием данного блока.

Зафиксировать фоновое изображение позволяет свойство **background-attachment**, которое может принимать одно из двух значений:

fixed - фиксированный фон.

scroll - подвижный фон (по умолчанию).

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Фиксация фонового
изображения</title>
    <style type="text/css">
      body{
        background-image: url(fon.jpg);
        background-attachment: fixed;
      }
      h1{
        color: #0000ff;
        font: bold 28px;
      }
      div{
        line-height: 2;
        white-space: pre;
```

```
            color: #0000ff;
            font-size: 24px;
          }
      </style>
    </head>
    <body>
      <h1>Песенка мамонтенка</h1>
      <div>
По синему морю, к зелёной земле
Плыву я на белом своём корабле.
... ... ...
... ... ...
Музыка: В. Шаинский
Слова: Д. Непомнящий
      </div>
    </body>
</html>
```

Повторение фонового изображения.

Как уже говорилось выше, если рисунок не заполняет собой весь фон элемента, то он повторяется и выкладывается "плиткой". Свойство **background-repeat** - регулирует повторение фонового изображения.

Возможные значения:

no-repeat - запретить повторение, показать только одно изображение.

repeat - повторять изображение (по умолчанию).

repeat-x - повторять только по горизонтали.

repeat-y - повторять только по вертикали.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Повторение фонового
изображения</title>
  </head>
  <body style="background-image:
url(fon.jpg); background-repeat:
repeat-x">
    <samp style="background-color:
```

```
#ffffff">Изображение повторяется
только по оси x.</samp>
  </body>
</html>
```

Позиция фонового изображения.

При помощи CSS свойства **background-position** - можно управлять позицией фонового изображения в элементах где оно задано с помощью **background-image**.

Позицию фонового изображения, а точнее его верхнего левого угла, можно задать в процентах, пикселях, а также любых других единицах измерения CSS, с помощью двух значений, где первое значение будет являться отступом от левой границы элемента, а второе значение отступом от верхней границы элемента.

Например, запись: **background-position: 200px 100px;** будет обозначать, что фоновое изображение будет смещено по горизонтали на 200 пикселей от левой границы элемента

и по вертикали на 100 пикселей от верхней границы элемента.

Так же можно использовать следующие значения:

по горизонтали:

left - расположить слева.

center - расположить по центру.

right - расположить справа.

по вертикали:

top - расположить сверху.

center - расположить по центру.

bottom - расположить снизу.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Позиция фонового
```

```
изображения</title>
    <style type="text/css">
      body{
        background-image: url(fon.jpg);
        background-repeat: no-repeat;
        background-position: center
100px;
      }
    </style>
  </head>
  <body>
    <samp>В этом примере фоновое
изображение по горизонтали
расположено по центру, а по
вертикали в ста пикселях от верхней
границы элемента.</samp>
  </body>
</html>
```

Background.

Ну и в довершении главы немного о базовом свойстве **background**.

Являясь базовым свойством, **background** может одновременно

принимать те или иные значения от его дочерних свойств:

background-attachment

background-color

background-image

background-position

background-repeat

Все эти свойства мы рассмотрели выше, так что вдаваться в подробности нет смысла..

Нужные значения дочерних свойств указываются через пробел, в любом порядке и по мере необходимости.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Background</title>
```

```
  </head>
  <body style="background:
url(fon.jpg) fixed repeat-x center">
    <p style="background: red">Фон
этого параграфа такой же красный как
если бы мы использовали background-
color: red.</p>
    <p>А у элемента body в базовом
свойстве background указанно сразу
четыре возможных значения взятых от
его дочерних свойств</p>
  </body>
</html>
```

Полезные советы:

При использовании изображения в качестве фона с помощью свойства **background-image** заодно заливайте фон альтернативным цветом с помощью **background-color**. Так если, по каким либо причинам, не загрузится фоновое изображение или же пользователь умышленно отключит в настройках браузера загрузку рисунков, фон элемента не останется "голым".

Не используйте большие весом "мегабайтные" изображения берегите время деньги и нервы пользователей.. А если уж без этого не обойтись, то перед тем как выкладывать такие изображения, как следует, поработайте с ней в графических редакторах на предмет "лишнего веса". Почти всегда можно значительно сжать картинку особо не проиграв в качестве изображения.

Глава 5

Границы элемента.

В этой главе мы поговорим о том, как сделать с помощью CSS рамку - бордюр, вокруг того или иного элемента. В HTML эта задача лежала на плечах атрибута **border**, однако его можно было применить далеко не к каждому тегу (элементу) да и не всегда он мог решить ту или иную дизайнерскую задумку.

В CSS эту задачу берёт на себя одноимённое базовое свойство **border** и значительно расширяет круг возможностей при работе со стилем границы любого(!) элемента выводимого на экран.

Стиль границы.

Если в HTML бордюр мог быть только в виде сплошной линии вокруг элемента, то в CSS это уже достаточно широкий набор возможных стилей рамок.

Свойство **border-style** может присваивать элементу один из ниже перечисленных стилей границы.

none - граница отсутствует (по умолчанию).

dotted - граница из ряда точек.

dashed - пунктирная граница.

solid - сплошная граница

double - двойная граница

groove - граница "бороздка"

ridge - граница "гребень"

inset - вдавленная граница

outset - выдавленная граница

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
```

```
    <title>Стиль границы</title>
    <style type="text/css">
      p {
        background-color: #f5f5f5;
        text-align: center;
      }
    </style>
  </head>
  <body>
    <p style="border-style:
none;">граница отсутствует</p>
    <p style="border-style:
dotted;">граница из ряда точек</p>
    <p style="border-style:
dashed;">пунктирная граница</p>
    <p style="border-style:
solid;">сплошная граница</p>
    <p style="border-style:
double;">двойная граница</p>
    <p style="border-style:
groove;">граница "бороздка"</p>
    <p style="border-style:
ridge;">граница "гребень"</p>
    <p style="border-style:
inset;">вдавленная граница</p>
    <p style="border-style:
```

```
outset;">выдавленная граница</p>
  </body>
</html>
```

Стиль бордюра может быть задан как для всех сторон элемента одновременно, так и для каждой его стороны отдельно в зависимости от того, сколько значений присвоено свойству **border- style**. Таковых значений может быть от одного до четырёх по числу сторон элемента.

В каждом из четырёх случаев действуют свои "правила" по присуждению стиля рамки той или иной стороне элемента, которые приведены в таблице ниже:

Число значений	**Результат**
1	Пример: div {**border-style: solid;**} Первое значение - Устанавливает единый стиль бордюра для всех

	сторон элемента.
2	Пример: div {**border-style: solid double;**} Первое значение - Устанавливает стиль верхней и нижней границы элемента. Второе значение - Устанавливает стиль левой и правой границы элемента.
3	Пример: div {**border-style: solid double dashed;**} Первое значение - Устанавливает стиль верхней границы элемента. Второе значение - Устанавливает стиль левой и правой границы элемента. Третье значение - Устанавливает стиль

	нижней границы элемента.
4	Пример: div {**border-style: solid double dashed ridge;**} Первое значение - Устанавливает стиль верхней границы элемента. Второе значение - Устанавливает стиль правой границы элемента. Третье значение - Устанавливает стиль нижней границы элемента. Четвёртое значение - Устанавливает стиль левой границы элемента.

Толщина границы.

Свойство **border-width** - устанавливает ширину границы элемента.

Ширина бордюра может быть заданна с помощью следующих аргументов:

thin - тонкая граница

medium - средняя толщина границы

thick - толстая граница

А также в пикселях или любых других единицах измерения принятых в CSS.

По аналогии со стилем, толщина бордюра тоже может иметь от одного до четырёх значений и в каждом случае устанавливает ёе для тех или иных сторон бордюра как показано в таблице ниже.

Число значений	Результат
1	Пример: div {border-style: solid; **border-width: 1px;**} Первое значение - Устанавливает единую толщину бордюра со всех сторон.
2	Пример: div {border-style: solid; **border-width: 1px**

	4px;} Первое значение - Устанавливает толщину верхней и нижней границы элемента. Второе значение - Устанавливает толщину левой и правой границы элемента.
3	Пример: div {border-style: solid; **border-width: 1px 4px 7px;**} Первое значение - Устанавливает толщину верхней границы элемента. Второе значение - Устанавливает толщину левой и правой границы элемента. Третье значение - Устанавливает толщину

	нижней границы элемента.
4	Пример: div {border-style: solid; **border-width: 1px 4px 7px 5px;**} Первое значение - Устанавливает толщину верхней границы элемента. Второе значение - Устанавливает толщину правой границы элемента. Третье значение - Устанавливает толщину нижней границы элемента. Четвёртое значение - Устанавливает толщину левой границы элемента.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
  <head>
    <title>Толщина границы</title>
  </head>
  <body>
    <div style="border-style: solid;
border-width: 3px 1px 10px 4px">
      Толщина границ данного блока:
      <ul>
        <li>Сверху 3 пикселя
        <li>Справа 1 пиксель
        <li>Снизу 10 пикселей
        <li>Слева 4 пикселя
      </ul>
      </div>
    <br><br>
    <div style="border-style: double;
border-width: thick">В этом блоке
границы со всех сторон одинаково
толстые</div>
  </body>
</html>
```

Цвет границы.

Цвет рамки или её сторон по отдельности определяется свойством **border-color.**

Цвет бордюра может иметь следующие значения:

. **#ff0000** - шестнадцатеричное значение цвета RGB.
. **red** - именное значение цвета.
. **RGB(255,0,0)** - значение цвета RGB.
. **transparent** - прозрачная граница.

Ну и так же как и в случаях с толщиной и стилем, цвет бордюра тоже может иметь от одного до четырёх цветовых значений при каждом "раскладе" окрашивая нужные стороны бордюра как показано в таблице ниже.

Число значений	Результат
1	Пример: div {border-style: solid; border-width: 3px;

	border-color: #008000;} • Первое значение - Устанавливает единый цвет бордюра со всех сторон.
2	Пример: div {border-style: solid; border-width: 3px; **border-color: #008000 #0000ff;**} • Первое значение - Устанавливает цвет верхней и нижней границы элемента. • Второе значение - Устанавливает цвет левой и правой границы элемента.
3	Пример: div {border-style: solid; border-width: 3px; **border-color: #008000 #0000ff #ff0000;**} • Первое значение -

	Устанавливает цвет верхней границы элемента. • Второе значение - Устанавливает цвет левой и правой границы элемента. • Третье значение - Устанавливает цвет нижней границы элемента.
4	Пример: div {border-style: solid; border-width: 3px; **border-color: #008000 #0000ff #ff0000 #ffff00;**} • Первое значение - Устанавливает цвет верхней границы элемента. • Второе значение - Устанавливает цвет правой границы элемента. • Третье значение - Устанавливает цвет нижней границы элемента. • Четвёртое значение - Устанавливает цвет левой

	границы элемента.

Что то уж больно много почти идентичных таблиц получилось.. ну не судите строго у каждого читателя свой уровень подготовки, да и учебник этот с маркой: -"CSS для начинающих" так что лучше перестраховаться, рассмотрев каждый случай отдельно, не желе чем быть непонятым..

Так вот перестраховываясь, привожу пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Цвет границы</title>
  </head>
  <body>
    <div style="border-style: solid;
border-width: 10px; border-color:
#ff0000 #ffff00 #00ff00 #0000ff;">
```

```
        <p style="border-style: double;
border-width: 5px; border-color:
#008000;">Зелёный</p>
        <p style="border-style: double;
border-width: 5px; border-color:
red;">Красный</p>
        <p style="border-style: double;
border-width: 5px; border-color:
rgb(0,0,255);">Синий</p>
    </div>
  </body>
</html>
```

Границы справа слева сверху и снизу отдельно.

Для того, что бы определить стиль, цвет и ширину бордюра для одной из сторон элемента, пользуйтесь свойствами **border-bottom, border-left, border-right, border-top** и их дочерними "коллегами по цеху" список которых приведён ниже:

border-bottom - Определяет стиль, цвет и ширину нижней границы элемента.

. **border-bottom-color** - Устанавливает цвет нижней границы элемента.

. **border-bottom-style** - Определяет стиль нижней границы элемента.

. **border-bottom-width** - Определяет ширину нижней границы элемента.

border-left - Определяет стиль, цвет и ширину левой границы элемента.

. **border-left-color** - Устанавливает цвет левой границы элемента.

. **border-left-style** - Определяет стиль левой границы элемента.

. **border-left-width** - Определяет ширину левой границы элемента.

border-right - Определяет стиль, цвет и ширину правой границы элемента.

. **border-right-color** - Устанавливает цвет правой границы элемента.

. **border-right-style** - Определяет стиль правой границы элемента.

. **border-right-width** - Определяет ширину правой границы элемента.

border-top - Определяет стиль, цвет и ширину верхней границы элемента.

. **border-top-color** - Устанавливает цвет верхней границы элемента.
. **border-top-style** - Определяет стиль верхней границы элемента.
. **border-top-width** - Определяет ширину верхней границы элемента.

Не буду описывать каждый из них, думаю и так понятно, что дело обстоит, так же как и с их сородичами, свойствами **border-style, border-width и border-color**, кроме того факта, что в данном случае свойства указываются для одной из сторон границы элемента.

Приведу пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Граница слева</title>
```

```
    <style type="text/css">
      div{
        border-left: 10px solid
#008000;
        background: #c6f2de;
      }
    </style>
  </head>
  <body>
    <div>
      <p>В этом примере у контейнера
div мы обозначили только его левую
границу с помощью свойства border-
left написав:</p>
      div{<br>
      border-left: 10px solid
#008000;<br>
      background: #c6f2de;<br>
      }
      <p>Такого же результата можно
было бы добиться прописывая
свойства стиля, толщины и цвета для
левой границы элемента
отдельно.</p>
      <p>Это выглядело бы вот
так:</p>
```

```
        div{<br>
        border-left-color: #008000;<br>
        border-left-style: solid;<br>
        border-left-width: 10px;<br>
        background: #c6f2de;<br>
        }
        <p>Кстати Вам этот блок ничего не напоминает? :)</p>
    </div>
  </body>
</html>
```

Border

Свойство **border** - базовый атрибут одновременно определяет стиль, цвет и толщину границы элемента.

Так как атрибут **border** является базовым, значения родственных свойств указываются в любом порядке через пробел.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
```

```
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>border</title>
    <style type="text/css">
      div{
        border: RGB(25, 125, 25) 6px
ridge;
      }
    </style>
  </head>
  <body>
    <div>
      <h3>А знаете ли Вы что:</h3>
      <p>Слон - символ
положительного характера -
используется в Азии как царское
верховное животное и высоко ценится
за ум и хитрость.</p>
... ... ...
    </div>
  </body>
</html>
```

Однако если Вы хотите присвоить разные свойства различным сторонам

границы элемента или только одной из них, пользуйтесь свойствами **border-bottom, border-left, border-right, border-top.**

Границы таблицы.

Свойство CSS **border-collapse** определяет стиль отображения границ таблицы.

По умолчанию каждая ячейка таблицы имеет собственную рамку (ну если конечно использован атрибут HTML **border** или одноимённое свойство CSS), так вот в местах соприкосновения ячеек образуется двойная линия, **border-collapse** заставляет браузер анализировать таковые места и поступать с ними согласно присвоенному значению данному свойству.

Внешний вид границ таблицы может принимать следующий вид:

. **separate** - ячейки таблицы отделены друг от друга (по умолчанию).

. **collapse** - ячейки таблицы не имеют промежутков между собой.

. **inherit** - свойства наследуются у родителя элемента. (работает далеко не во всех браузерах.)

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Стиль таблицы</title>
  </head>
  <body>
    <table cellpadding="5" border="5">
      <caption>Таблица с бордюром по
умолчанию</caption>
      <tr>

<td>ячейка</td><td>ячейка</td><td>яч
ейка</td><td>ячейка</td>
```

```
        </tr>
        <tr>

<td>ячейка</td><td>ячейка</td><td>яч
ейка</td><td>ячейка</td>
        </tr>
    </table>
    <hr>
    <table cellpadding="5" border="5"
style="border-collapse: collapse">
        <caption>А эта таблица
использует свойство CSS border-
collapse с значением
collapse</caption>
        <tr>

<td>ячейка</td><td>ячейка</td><td>яч
ейка</td><td>ячейка</td>
        </tr>
        <tr>

<td>ячейка</td><td>ячейка</td><td>яч
ейка</td><td>ячейка</td>
        </tr>
    </table>
```

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

Свойство **border-collapse** применяется только к тегу **<table>** и элементам, у которых атрибут **display** имет значение **table** или **inline-table**.О свойстве **display** расскажу в отдельной главе.

Полезные советы:

. Довольно часто слышу вопрос: - "Как сделать вертикальную линию, по аналогии с тегом **<hr>** - горизонтальная разделительная линия?".

Один из способов решения данной задачи является использование блока "пустышки" нужных размеров с левым или правым бордюром (свойства: **border-left** или **border-right**) необходимого стиля, который собственно и будет выступать в роли вертикальной линии.

Глава 6

Классы и идентификаторы.

Данная глава является своего рода продолжением главы первой **о способах внедрения CSS в HTML документ.** Отложил эту главу "на потом", нарушив тем самым стандартную последовательность обучения CSS, с той лишь целью, чтобы раньше времени не забивать Вам голову информацией, которая в начале была для Вас малопонятной.

Теперь, когда Вы уже познакомились с некоторыми свойствами CSS, поняли принцип построения и внедрения CSS в HTML, настал час и необходимость поговорить о CSS классах и идентификаторах, так как дальнейшее обучение, хоть и представляется возможным, но не раскроет в полной мере все возможности CSS.

Классы CSS.

Начнём с классов..

Как присвоить элементу или группе идентичных элементов индивидуальный стиль, отличный от основного, уже указанного в стилевом описании документа? Не знаю задавались Вы этим вопросом или нет, но рано или поздно на него необходимо найти ответ.

Итак.. предположим в файле CSS к элементу **<p>** у нас применён следующий стиль:

```
p {color: #0000ff; font-size:14px}
```

И все вроде бы хорошо.. все параграфы синенькие и размер у них 14px, но нам надо сделать так чтобы некоторые из этих параграфов были розовые! И как быть??

На помощь приходят классы.

Для того чтобы выделить некоторые из параграфов розовым цветом, необходимо присвоить элементу определённое имя и вывести его тем

самым в класс, в некую нестандартную, для страницы или сайта в целом, категорию.

Ну что давайте попробуем? Делается это так:

p.rose {color: #ff00ff; font: italic 16px Arial}

Поясню **p** - это элемент HTML (селектор) в данном случае наш параграф, **.rose** - это индивидуальное имя класса которое мы сами выдумали, оно может быть любым необязательно **rose**-розовый, точка между селектором и именем класса есть дань уважения к синтаксису принятому в CSS - теперь браузер поймет, что данный элемент p выведен в класс **rose**.

Ну что ж имя мы присвоили теперь нам необходимо в документе HTML указать теги (в нашем случае теги **<p>**) которым необходим индивидуальный стиль. Делается это с помощью атрибута **class**.

Вот так:

** **Этот параграф использует имя класса **rose** и тем самым выделяется из основной массы** **

Ну и пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Внедрение класса</title>
    <style type="text/css">
      body {background-color: #c5ffa0}
      p {color: #0000ff; font-size:14px}
      p.rose {color: #ff00ff; font: italic
16px Arial}
    </style>
  </head>
  <body>
    <p>На этом сайте Вы найдёте
любую информацию о слонах.</p>
    <p>У нас Вы можете по выгодным
```

```
ценам приобрести лучших
слонов!!</p>
    <p>Только у нас Вы можете взять
любых слонов на прокат!!</p>
    <p class="rose">Специальное
предложение для девушек! Розовые
слоны!! только у нас!!!</p>
  </body>
</html>
```

В данном примере класс **"rose"** может быть присвоен только параграфу - элементу **p**. Для того чтобы данное стилевое описание могло распространятся на все элементы, в файле CSS (или между тегами **<style></style>** в заголовке документа) элемент явно не указывается и синтаксис приобретает следующий вид:

```
.rose {color: #ff00ff}
```

Теперь указав в любом элементе class="rose" он примет стиль данного класса.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Классы</title>
    <style type="text/css">
      body {background-color: #c5ffa0}
      h1 {color: #0000ff; font-size:22px}
      p {color: #008000; font: italic 16px
Arial}
      span {color: #0080ff; font-
size:12px}
      .rose {color: #ff00ff}
    </style>
  </head>
  <body>
    <h1>Это заголовак с основным
стилем CSS</h1>
    <h1 class="rose">А этот заголовок
использует class="rose"</h1>
    <hr>
    <a href="#" title="Обыкновенная
ссылка">Это ссылка по
умолчанию</a><br>
```

```
    <a href="#" title="Розовая ссылка" class="rose">А эта ссылка использует class="rose"</a><br>
    <hr>
    <span>Этот строковый блок использует основной стиль</span><br>
    <span class="rose">А этот класс rose</span>
    <hr>
    <p>Параграф без указания класса</p>
    <p class="rose">Параграф с указанием класса</p>
  </body>
</html>
```

Обратите внимание на тот факт, что недостающие описания стиля выведенного в отдельный класс, элементы черпают из основного стилевого описания или же берут из свойств элемента "по умолчанию".

Например, заголовок **<h1 class="rose">** был синим, а стал розовым, но при этом сохранил свой

размер 22 пикселя, так как в нашем классе rose никаких разговоров о размере шрифта не шло.. мы лишь "договорились" с браузером, что элементы из группы **rose** будут розовыми.

Идентификаторы

Идентификаторы они же id селекторы, весьма схожи с классами, с тем лишь отличием, что идентификатор может иметь одно единственное уникальное имя во всем документе. Идентификаторы, как правило, применяются в том случае, если возникает необходимость управлять стилем элемента динамически с помощью скрипта, обращаясь к его индивидуальному имени.

В файле CSS имя указывается со знаком решётки в его начале.

Вот так, например:

#block {color: #ff00ff; font: italic 16px Arial}

А к нужному элементу добавляется атрибут id="block" например

<p **id="block"**>Параграф с идентификатором</p>

Вот пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Идентификатор</title>
    <style type="text/css">
      body {background-color: #c5ffa0}
      p {color: #0000ff; font-size:14px}
      #rose {color: #ff00ff; font: italic
16px Arial}
    </style>
  </head>
  <body>
    <p>Это обыкновенный
параграф</p>
```

```
    <p id="rose">А этот параграф уникальный</p>
  </body>
</html>
```

Ну так в чем же отличие между классом и идентификатором?

Покажу на примере:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Идентификаторы и скрипты</title>
    <script>
      function show_hide(id){
      var item = document.getElementById(id);
      if (item.style.display == 'none') {item.style.display = 'block';}
      else item.style.display = 'none';
      }
    </script>
```

```
  </head>
  <body>
    <div id="block"
style="display:none">
      <h2 style="color: #ff00ff">А вот и
я!!</h2>
      <img src="rosemammoth.gif">
    </div>
    <a
href="javascript:show_hide('block')"
title="Развернуть/Свернуть"
style="color: #ff00ff">Нажми на
меня!!</a>
    <hr>
    <div id="block1"
style="display:none">
      <h2 style="color: #0000ff">А здесь
я!!</h2>
      <img src="mammoth.jpg">
    </div>
    <a
href="javascript:show_hide('block1')"
title="Развернуть/Свернуть"
style="color: #0000ff">И на меня
нажми!!</a>
```

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

Курсивом, в данном примере, выделен скрипт, который может динамически обрабатывать блоки ** ** с уникальными именами **"block"** и **"block1"** (скрывать и показывать его по нажатии на ссылку), и хотя пока Вам, думаю, мало, что понятно из выше написанного, но цель данного примера показать, как скрипт может обращаться к блоку через атрибут **id**. А вот с помощью классов мы бы не достигли такого результата.

Ну, думаю, объяснил кое как..

Полезные советы:

- При построении CSS будьте логичны, соблюдайте "значимость" элементов и их порядок, так же как они вложены друг в друга в HTML коде.

Например:

body {сначала опишите стиль страницы в целом}
div {потом её отдельных частей - блоков}
a {затем ссылок}
h1.-.h6 {далее заголовков}
p {и в конце параграфов}

Для чего это нужно?

Ну во-первых, просто для удобного чтения и "навигации" по CSS описанию. Когда Вам потребуется найти какой ни будь элемент Вы уже изначально будете представлять где он приблизительно находится в начале, середине, или конце..

Во-вторых, что более важно, загрузка страницы происходит не моментально и не всегда приятно наблюдать как содержание данной страницы при загрузке "прыгает" и всячески "шевелится" так как сначала прописываются "малозначимые" стили элементов, например шрифт параграфов, а в конце "значительные"

например размеры блоков, с помощью которых свёрстан весь сайт. К тому же загрузка, по каким либо причинам, вообще может пройти не до конца..

Что увидит (сначала увидит) пользователь при "неправильном" построении CSS? Красивый шрифт, беспорядочно разбросанный в окне браузера? Или нормальное построение, но без красивого шрифта? - Это уже решать Вам!

- При использовании классов и идентификаторов придумывайте им осмысленные информативные имена. Варианты тапа: **.aaa .123 #abc #cba** приведут к путанице!, я уж не говорю о том, что возможно в Вашем коде будет разбираться посторонний человек. Придумайте свою "систему" названий и не нарушайте её, так Вы сэкономите собственное время и затраченные усилия.

Глава 7

Размеры элемента.

Блоки и строки.

Прежде чем говорить о работе с размерами элементов их расположении, полях и отступах, расскажу немного о самих элементах, точнее об их типах.

Все элементы (теги) можно разделить на две категории: Блочные и строковые.

Чем они отличаются?

Думаю, сами названия уже говорят о различиях. Блочные или боксовые элементы - это контейнеры, прямоугольные области на мониторе компьютера которым без труда и вело-изобретательства можно присвоить определенное положение, размер, вложить в них другие блоки, определить расположение относительно друг друга. А строковые элементы располагаются

в одну сроку, выравниваются по её базовой линии и как правило служат для форматирования и редакции текста.

К блочным (**block**) элементам относятся:
** , <dl>, <form>, <h1>- <h6>, <hr>, <noscript>, <ol>, <p>, <pre>, <table>, **…

К строчным (**inline**) элементам относятся:
**<a>,
, <cite>, <code>, , <img>, <input>, <label>, <select>, , , <sub>, <sup>, <textarea>**…

Основные отличия от элементов строкового и блочного типа заключаются в том что:

Боксовые элементы могут содержать в себе другие элементы, как блочного, так и строкового типа. Строчные же элементы могут включать в себя только строчные элементы.

На примере:

```
<div>
  <b>Это <i>правильная строка</i>
в блоке</b>
</div>

<i><b>
  <div>А это неправильная толи
строка толи блок. Так лучше не
делать!!</div>
</b></i>
```

Боксовые элементы, по умолчанию, располагаются относительно друг друга вертикально, т.е. в начале и конце блока происходит "перенос строки", а строковые элементы располагаются горизонтально, перенос строки происходит только в том случае если это необходимо.

По умолчанию блочные элементы имеют ширину 100% и тем самым оставляют за собой все пространство по горизонтали, у строковых же

элементов ширина определяется содержанием.

Думаю, на примере будет понятнее.

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Блочные и строковые
элементы</title>
  </head>
  <body>
    <p style="background-color:
#00ffff;">Параграф</p>
    <p style="background-color:
#00ff00;">Параграф</p>
    <p style="background-color:
#ffff00;">Параграф</p>
    <hr>
    <span style="background-color:
#00ffff;">Строка</span>
    <span style="background-color:
#00ff00;">Строка</span>
    <span style="background-color:
```

```
#ffff00;">Строка</span>
      <hr>
      <div style="background-color:
#00ffff;">Блок</div>
      <div style="background-color:
#00ff00;">Блок</div>
      <div style="background-color:
#ffff00;">Блок</div>
   </body>
</html>
```

Ширина и высота блочных элементов.

Свойства CSS **width** и **height** - устанавливают ширину и высоту блочных элементов. Ширина и высота элемента может быть задана следующими способами:

. **auto** - Размеры элемента определяется его содержанием. (по умолчанию)

. **%** - Размеры элемента задаётся в процентах от высоты/ширины элемента родителя.

- **px** - Размеры элемента задаётся в пикселях или любых других единицах измерения принятых в CSS.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Размеры блока</title>
  </head>
  <body>
    <div style="background-color: #00ffff;
height: 100px; width: 250px">Блок
1</div>
    <div style="background-color:
#00ff00; height: 150px; width:
50%">Блок 2</div>
    <div style="background-color: #ffff00;
height: auto; width: auto">Блок
3</div>
  </body>
</html>
```

Если содержание элемента превышает его указанный размер, то в некоторых браузерах элемент автоматически присвоит значение auto, а в некоторых содержание "выползет" за пределы элемента.

Управление содержанием элемента.

Что делать с содержанием элемента, если оно превышает его размер?

Если элементу присвоены точные значения высоты и ширины (**height, width**) а его содержание, например длинный текст, не вмещается в указанных пределах, то по умолчанию такой элемент растягивается до нужных размеров, что не всегда на руку веб-мастеру. На помощь приходит атрибут **overflow**, который указывает браузеру, что делать с элементом в таких случаях.

Свойство **overflow** может иметь следующие значения:

. **visible** - Элемент растягивается до необходимых размеров. (по умолчанию)

. **hidden** - Содержание элемента "обрезается" видна лишь та его часть что помещается в элементе.

. **scroll** - Добавляются полосы прокрутки (всегда! даже если содержание помещается в пределах элемента).

. **auto** - Полосы прокрутки добавляются при необходимости.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Содержание элемента</title>
  </head>
  <body>
    <div style="overflow: auto; width:
250px; height: 150px; border: solid 2px
#dddddd">
```

```
        <h3 align="center">Диктант</h3>
На дощатой террасе, близ конопляника, небезызвестная вдова Агриппина Саввечна, потчевала, отставного коллежского асессора Аполлона Карповича, моллюсками и другими яствами, под аккомпанемент виолончели.
      </div>
   </body>
</html>
```

Минимальные и максимальные размеры элемента.

Так как размеры элемента по умолчанию регулируются исключительно вложенным в него содержанием, текстом, графикой.. и т.д. то элементы становятся, вернее, остаются "резиновыми" и это хорошо! Но хорошо не во всех случаях.. иногда требуется ограничить эту "резиновость" конкретными размерами. Например, указать ячейке таблицы, что ты мол, независимо от твоего содержания, можешь быть по

высоте не менее 50 пикселей, но и не более 200 пикселей.. определить, так сказать, диапазон её высоты или ширины.

Ниже перечисленные свойства CSS позволяют определить минимальные и максимальные размеры того или иного элемента

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

Эти свойства CSS, в свою очередь, могут выражаться в пикселях, процентах от размеров родительского элемента и **none** - без ограничений.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
```

```
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Минимальные и
максимальные размеры
элемента</title>
  </head>
  <body>
    <div style="min-height: 50px; min-
width: 100px; max-height: 250px;
max-width: 300px; border: solid 2px
#dddddd">
      Браузер Internet Explorer
игнорирует свойства css min-width,
max-width, min-height и max-height.
    </div>
  </body>
</html>
```

Полезные советы:

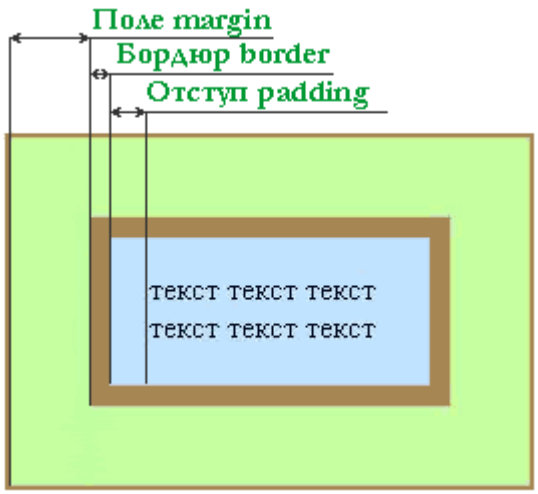
Как Вы уже поняли не все браузеры одинаково интерпретируют как HTML теги, так и свойства CSS. Особым "невежеством" по отношению как к пользователям так и к веб дизайнерам

отличился Internet Explorer 6 и его более ранние версии.. Впрочем, и в остальных браузерах далеко не всё проходит гладко. Отсюда необходимо сделать вывод, что прежде чем выкладывать ту или иную страницу в Интернет её необходимо просмотреть (протестировать) в различных браузерах и их версий. А совет заключается собственно в том, что хорошо бы было их иметь под рукой - в установленном виде на Вашем компьютере. Соберите свою "коллекцию" браузеров.

Глава 8

Поля и отступы.

В этой главе мы поговорим о полях (свойство **margin**) и отступах (свойство **padding**) элемента. Между собой эти два свойства весьма схожи, однако все же это два абсолютно разных свойства.



Давайте разберемся в этих понятиях..

Поле (**margin**) - Это расстояние от внешней границы элемента до границы окна браузера или же элемента родителя.. ну границы того блока в который вложен наш элемент.

Отступом (**padding**) - Называют расстояние от внутренней границы элемента до его содержания текста, картинок таблиц..

На рисунке наглядно показаны эти расстояния, а так же место занимаемое бордюром (**border**) про него тоже не следует забывать.

Расстояния **margin** и **padding** указываются:

- **px** - В пикселях или любых других допустимых CSS единицах измерения.
- **%** - В процентах.
- **auto** - Размер полей и отступов автоматически рассчитывается браузером.

Давайте попробуем сделать тоже самое, что показано на рисунке с помощью блоков ** ** и свойств CSS.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Поля и отступы</title>
```

```
    <style type="text/css">
      div.block1 {
        width: 260px;
        background-color: #c5ffa0;
        border: 2px solid #a68754
      }
      div.block2 {
        background-color: #c0e4ff;
        border: 8px solid #a68754;
        margin: 40px;
        padding: 20px
      }
    </style>
  </head>
  <body>
    <div class="block1">
      <div class="block2">
        <samp>Осмысливая мысли в
смысле смысла, есть смысл
помыслить о немыслимом!</samp>
      </div>
    </div>
  </body>
</html>
```

Попробуйте поменять значения свойств **margin** и **padding** и Вы поймете, что к чему.. а если и так понятно идем дальше..

Возможные значения margin и padding.

В примере выше присудив свойствам **margin** и **padding** по одному значению, мы определили поля и отступы элемента со всех четырех его сторон. Для того чтобы указать разные размеры полей и отступов для каждой из сторон элемента, необходимо указывать два, три или четыре аргумента через пробел. Причем в зависимости от количества значений результат будет разным.

Вот примеры:

margin: 5px; - *одно значение*. Одно значение - Поля одинаковые со всех сторон.

margin: 5px 40px; - *два значения.*
Первое значение - Устанавливает поля от верхней и нижней границ.
Второе значение - Устанавливает поля от левой и правой границ элемента.

margin: 5px 40px 20px; *- три значения.*
Первое значение - Устанавливает поле от верхней границы элемента.
Второе значение - Устанавливает поля от левой и правой границ элемента.
Третье значение - Устанавливает поле от нижней границы элемента.

margin: 5px 40px 20px 1px; - *четыре значения.*
Первое значение - Устанавливает поле от верхней границы элемента.
Второе значение - Устанавливает поле от правой границы элемента.
Третье значение - Устанавливает поле от нижней границы элемента.
Четвёртое значение -

Устанавливает поле от левой границы элемента.

С **padding** дело происходит по тому же принципу, как и с **margin**, отступы от внутренних границ элемента до содержания приобретают размеры в зависимости от количества аргументов и их значений.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Поля и отступы</title>
    <style type="text/css">
      div.block1 {
        width: 260px;
        background-color: #c5ffa0;
        border: 2px solid #a68754
      }
      div.block2 {
        background-color: #c0e4ff;
```

```
            border: 2px solid #a68754;
            margin: 20px 5px 20px 40px;
            padding: 5px 25px
        }
    </style>
  </head>
  <body>
    <div class="block1">
      <div class="block2">
        <samp>Осмысливая мысли в смысле смысла, есть смысл помыслить о немыслимом!</samp>
      </div>
      <div class="block2">
        <samp>Мало кто знает, как много надо знать, для того чтобы знать, как мало мы знаем!</samp>
      </div>
    </div>
  </body>
</html>
```

Дочерние свойства margin и padding.

Свойства CSS **margin** и **padding** являются базовыми и имеют ряд дочерних свойств.

У **margin** зто:

- **margin-top** - Поле от верхней границы элемента родителя,
- **margin-left** - Поле от левой границы элемента родителя,
- **margin-right** - Поле от правой границы элемента родителя,
- **margin-bottom** - Поле от нижней границы элемента родителя.

И у **padding** соответственно:

- **padding-top** - Отступ от верхней границы элемента до его содержания,
- **padding-left** - Отступ от левой границы элемента до его содержания,
- **padding-right** - Отступ от правой границы элемента до его содержания,
- **padding-bottom** - Отступ от нижней границы элемента до его содержания.

Ну думаю понятно.. если возникает необходимость определить отступ или

поле от одной единственной стороны элемента лучше воспользоваться одним из вышеперечисленных свойств.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Поля и отступы</title>
  </head>
  <body>
    <div style="width: 250px;
background-color: #c5ffa0; border: 2px
solid #a68754">
      <div style="background-color:
#c0e4ff; border: 2px solid #a68754;
margin-left:40px; padding-right:
50px">
        <samp>Осмысливая мысли в
смысле смысла, есть смысл
помыслить о немыслимом!</samp>
      </div>
```

```
        <br>
        <div style="background-color:
#c0e4ff; border: 2px solid #a68754;
margin-right:40px; padding-left:
50px">
            <samp>Мало кто знает, как
много надо знать, для того чтобы
знать, как мало мы знаем!</samp>
        </div>
      </div>
   </body>
</html>
```

Полезные советы:

Поля и отступы являются одними из основных составляющих дизайна сайта, по этому поводу следует придерживаться нескольких рекомендаций.

. Избегайте коротких, и уж тем более, нулевых отступов от содержимого к границам элемента, особенно это касается текстовых блоков. Текст с маленькими полями и отступами читается трудно и "вязко".

. Соблюдайте правила пропорциональности и соизмеримости полей и отступов как для отдельно взятого элемента, так и для всей страницы (композиции элементов) целиком.

. Один из способов акцентирования внимания посетителя страницы на нужном Вам месте - это отступы размером больше обычного, такой контраст заставит зрителя смотреть и проявить внимание к "особым" местам в тексте..

Глава 9

Курсоры.

Свойство CSS **cursor** позволяет установить нестандартный вид курсора для того или иного элемента - блока, текста, рисунка.. то есть когда пользователь наведет курсор на такой элемент он, курсор то бишь, поменяет свой вид.

Курсоры можно выбирать как стандартные, так и подгружать свои собственные пользовательские курсоры - файлы в формате **cur, ani** или **svg**.

Теперь по порядку..

Стандартные курсоры.

Вы наверняка обратили внимание на то, что по умолчанию, на странице, курсор приобретает вид в зависимости от специфики элемента, на который он нацелен, например если навести курсор на ссылку он отображается в виде указателя обычно "руки", если на

некий текст, то курсор становится вида "работа с текстом", а если просто елозить по пустому месту на странице, то курсор имеет обыкновенный вид "стрелка".

Так вот, для того чтобы изменить внешний вид курсора свойству **cursor** необходимо указать одно из возможных значений:

- **auto** - курсор назначается автоматически в зависимости от специфики элемента.(по умолчанию)
- **crosshair** - перекрестие
- **default** - встроенный(основной)
- **e-resize** - стрелка на "восток"
- **hand** - указатель аналогия pointer
- **help** - помощь
- **move** - перемещение
- **n-resize** - переместить на "север"
- **ne-resize** - переместить на "северо-восток"
- **nw-resize** - переместить на "северо-запад"
- **pointer** - указатель
- **progress** - продолжение операции

- **s-resize** - переместить на "юг"
- **se-resize** - переместить на "юго-восток"
- **sw-resize** - переместить на "юго-запад"
- **text** - текст
- **w-resize** - переместить на "запад"
- **wait** - ожидание

Кроме того некоторые браузеры поддерживают дополнительные формы курсоров:

- ***all-scroll*** - *переместить во все стороны*
- ***col-resize*** - *переместить по горизонтали*
- ***no-drop*** - *указатель "нет доступа"*
- ***not-allowed*** - *нет доступа*
- ***row-resize*** - *переместить по вертикали*
- ***vertical-text*** - *вертикальный текст*

Следует понимать, что данные значения являются, не каким либо конкретным изображением курсора, а определяют тип курсора. Внешний вид

курсора зависит от настроек операционной системы пользователя.

Проведите курсором над каждым значением свойства **cursor** в списке выше, чтобы увидеть какую форму принимает курсор в Вашем браузере.

Провели? А теперь если есть желание можете зайти в "Пуск" > "Панель управления" > "Мышь" > закладка "Указатели" > из списка "Схема" выберите любую другую схему.. Теперь можете снова провести по списку выше, чтобы увидеть, как теперь выглядят курсоры.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Стандартные курсоры</title>
  </head>
  <body>
```

```
    <div style="cursor: wait; padding: 30px; background-color: #c0e4ff; border: 2px solid #008000; height: 300px; width: 300px">
        Буду думу думать..
        <p style="width: 150px; text-align: center; cursor: help; background-color: #c5ffa0; border: 2px solid #008000">Кнопка "помощь"</p>
    </div>
  </body>
</html>
```

Пользовательские курсоры.

Для того чтобы курсор приобрёл нестандартный вид его необходимо подгрузить присвоив свойству cursor значение: **url("путь к курсору")**.

Например:

```
div { cursor : url("my.cur"); }
```

Так же можно указать несколько пользовательских курсоров, через запятую, в этом случае браузер будет пытаться отобразить первый из

перечисленных, если у него это не получится, возьмется за второй.. и т. д. А не получится может из-за того, что браузер не поддерживает определённый формат файла-курсора. Как уже говорилось выше можно использовать файлы в формате cur, ani или svg - такие курсоры поддерживаются, начиная с IE6, Firefox 1.5. Однако Firefox 1.5 не поддерживает формат ani, а IE6 в свою очередь, не понимает формат svg.

Поэтому список курсоров составляют из файлов разных форматов.. например, так:

div { cursor : url("my.cur"), url("my.svg"); }

И уж что б совсем обезопасить себя в конце списка рекомендуется ставить один из курсоров из стандартного набора.

Вот так:

div { cursor : url("my.cur"), url("my.svg"), help; }

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <style type="text/css">
      div {
        cursor : url("cursors/my.ani"),
crosshair;
        width: 200px;
        padding: 10px;
        text-align: center;
        background-color: #c5ffa0;
border: 2px solid #008000
      }
    </style>
    <title>Пользовательские
курсоры</title>
  </head>
  <body>
      <div>
```

```
        Если нацелив курсор на этот
блок вместо "навороченного" курсора
Вы видите обыкновенное перекрестие
значит Ваш браузер не поддерживает
формат ani
    </div>
  </body>
</html>
```

Ну и в конце..

Свойство **cursor**, имеет еще одно значение - **inherit**. которое говорит о том, что свойство наследуется от элемента-родителя.

Полезные советы:

. В подавляющем большинстве случаев не стоит вообще трогать курсоры, оставьте всё как есть. Пользователя могут ввести в заблуждение нестандартные курсоры, например, когда при наведении на ссылку вместо привычной "руки" появляется что-то другое. Пользовательские курсоры уместны

лишь там, где они действительно необходимы.

- Помните что любой курсор должен быть как минимум двухцветным. Это нужно для того чтобы его было видно на фоне такого же цвета как и сам курсор.. представьте одноцветный белый курсор на белом фоне.. вот уж задачка будет какой ни будь домохозяйке!! :)

Ааа.. у меня стрелка потерялась!!.. вирусы!!.. проклятый Бил Гейтс!!.. :) и давай лупить кулаком по системному блоку.. :)

Короче паники с истериками нам не нужны..

Глава 10

Форматирование.

В этой главе речь пойдет о форматировании элементов, на самом деле речь об этом уже шла в **седьмой главе**, в которой мы говорили о размерах элемента - **width** и **height**, минимальных и максимальных размеров элемента - **max-height, max-width, min -height, min-width** и свойстве **overflow**. Все свойства из седьмой главы относятся к категории "форматирование элементов" однако знания лучше получать порциями, чтобы каши в голове не случилось, вот я и решил придержать кой какой материал на потом..

Потом настало! так что готовьтесь переваривать информацию на медленном огне..

Показ элементов.

Свойство **display** указывает браузеру, как тот или иной элемент

должен быть показан на странице, другими словами определяет параметры вывода браузером элемента на экран.

Display имеет кучу возможных значений, однако, большинство из них поддерживаются далеко не всеми браузерами, поэтому мы поговорим лишь о тех которые будут корректно работать во всех браузерах , а таковых всего три:

- **block** - Показывает элемент как блочный. Происходит перенос строк вначале и в конце элемента.
- **inline** - Показывает элемент как встроеный. Элемент не переносятся на следующую строку.
- **none** - Запрещает показ элемента. Элемент "удаляется" из общего потока, его занимаемое пространство на экране не резервируется.

Теперь "расшифрую" написанное и покажу примеры..

Начнем с **block** и **inline**

Помните, в той же **седьмой главе** мы говорили о том, что все элементы можно разделить на строчные и блочные? так вот значения **block** и **inline** свойства **display** позволяет самостоятельно указывать, какие элементы мы хотим сделать строчными, а какие блочными, что позволяет решить ряд задач при верстке сайта с помощью CSS.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <style type="text/css">
      .menu {
        display: block;
        color:#006080;
        font-size: 14px;
        font-weight: bold;
        padding: 5px;
      }
```

```
        h2 {
          padding: 5px;
          color: #800080;
          font-size:16px;
        }
    </style>
    <title>display: block;</title>
  </head>
  <body>
    <p>В этом примере показано как можно сделать вертикальное меню присвоив ссылке (элемент <a>) свойство display с значением block.</p>
    <hr>
    <h2>Меню:</h2>
    <a href="#" class="menu">Все о слонах.</a>
    <a href="#" class="menu">Купить слона.</a>
    <a href="#" class="menu">Взять слона на прокат.</a>
    <hr>
    <p>Как видите ссылки стали блочными.. то есть в начале и в конце элемента <a> происходит перенос
```

```
строки, хотя по умолчанию ссылка должна была идти в общем потоке текста.</p>
  </body>
</html>
```

А вот пример использования значения **inline**:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>display: inline;</title>
  </head>
  <body>
    <p>Вот пример того как можно зпретить параграфу перенос строки</p>
    <hr>
    <p style="display: inline; background: #c5ffa0; padding: 5px;">Это параграф.</p>
    <p style="display: inline; background: #c0e4ff; padding: 5px;">И
```

```
это параграф!</p>
      <p style="display: inline;
background: #c5ffa0; padding: 5px;">И
снава параграф!!</p>
  </body>
</html>
```

Идем дальше.. значение **none** свойства **display** запрещает элемент к показу. Браузер удаляет элемент с таким значением из общего потока.

Небольшой пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>display: none;</title>
  </head>
  <body>
    А где второй блок ???
    <hr>
    <div style="background:
#c5ffa0">Блок 1</div>
```

```
    <div style="display: none">Блок 2</div>
    <div style="background: #c0e4ff">Блок 3</div>
  </body>
</html>
```

Конечно, от такого примера толку в практике ноль, какой смысл прописывать блок, а потом умышленно скрывать его!?? Однако это свойство незаменимо на страницах где присутствует динамика.. например для создания раскрывающегося списка.

Помнится мне, я Вам уже показывал этот пример:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>display:none и скрипты</title>
    <script>
```

```
        function show_hide(id){
        var item =
document.getElementById(id);
        if (item.style.display == 'none')
{item.style.display = 'block';}
        else item.style.display = 'none';
        }
      </script>
    </head>
    <body>
      <div id="block"
style="display:none">
        <h2 style="color: #ff00ff">А вот и
я!!</h2>
        <img src="rosemammoth.gif">
      </div>
      <a
href="javascript:show_hide('block')"
title="Развернуть/Свернуть"
style="color: #ff00ff">Нажми на
меня!!</a>
      <hr>
      <div id="block1"
style="display:none">
        <h2 style="color: #0000ff">А здесь
я!!</h2>
```

```
        <img src="mammoth.jpg">
    </div>
    <a
href="javascript:show_hide('block1')"
title="Развернуть/Свернуть"
style="color: #0000ff">И на меня
нажми!!</a>
  </body>
</html>
```

Курсивом, в данном примере, выделен скрипт, который может динамически обрабатывать блоки ** ** присваивая ему свойства **display:none** или **display: block**, и хотя пока Вам, думаю, мало, что понятно из выше написанного, но цель данного примера показать для каких целей предназначено свойство **display: none**.

Ну и для общего развития.. Как уже говорилось выше **block**, **inline** и **none** это далеко не все возможные значения свойства **display**.

Вот остальные:

. **list-item** - элемент выводится как блочный и добавляется маркер списка.

. **run-in** - устанавливает элемент как блочный или строковый в зависимости от контекста.

. **compact** - устанавливает элемент как блочный или строковый в зависимости от контекста.

. **marker** значение используется совместно с псевдоэлементами **before** и **after**, генерирует блок маркера

. **table** - элемент объявляется таблицей

. **inline-table** - элемент объявляется строкой таблицы

. **table-row-group** - элемент объявляется группой строк ячеек таблицы

. **table-header-group** - элемент объявляется группой строк ячеек таблицы, располагается в начале таблицы

. **table-footer-group** - элемент объявляется группой строк ячеек

таблицы, располагается в конце таблицы

. **table-row** - элемент отображается как строка таблицы

. **table-column-group** - определяет, что элемент является группой одной или более колонок таблицы

. **table-column** - элемент объявляется колонкой ячеек таблицы

. **table-cell** - элемент объявляется ячейкой таблицы

. **table-caption** - задает заголовок таблицы

Но использовать их в деле я бы не рекомендовал.. по той простой причине, что многие браузеры не поддерживают данные значения.. поэтому заострять внимания на них не будем.

Видимость элемента.

Свойство CSS **visibility** предназначено для отображения или скрытия элемента, включая рамку вокруг него и фон. При скрытии

элемента с помощью свойства **visibility** элемент становится, не виден, однако место, которое он занимает, остается за ним.

Возможные значения свойства **visibility**:

. **inherit** - Элемент отображается так как указанно в элементе родителе.
. **visible** - Отображает элемент. (по умолчанию).
. **hidden** - Скрывает элемент. В отличии от **display: none** оставляет за элементом площадь которую он занимает. Проще говоря если **display: none** "удаляет" элемент то **hidden** делает его "прозрачным".
. **collapse** - Скрывает колонки и строки таблицы идентично **display: none**, Если свойство применяется к другим элементам, не имеющим отношения к таблицам, то результат будет таким же, как **hidden**.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>visibility</title>
  </head>
  <body>
    <p>Блок 2 скрыт, однако место,
которое он занимает,
зарезервировано за ним.</p>
    <div>
      <span style="background: #c5ffa0;
padding: 5px;">блок 1</span>
      <span style="visibility: hidden;
background: #c0e4ff; padding:
5px;">блок 2</span>
      <span style="background: #c5ffa0;
padding: 5px;">блок 3</span>
    </div>
    <hr>
    <p>Строка 2 скрыта "удалена",
если только дело происходит не в
Internet Explorer</p>
    <table border="1">
```

```
        <tr>
          <td>Строка 1</td>
        </tr>
        <tr style="visibility: collapse;">
          <td>Строка 2</td>
        </tr>
        <tr>
          <td>Строка 3</td>
        </tr>
      </table>
    </body>
</html>
```

Обратите внимание **visibility: collapse;** не поддерживается браузером Internet Explorer.

Видимая часть элемента.

Свойство **clip** определяет видимую область элемента, в которой будет показано его содержимое. Все, что не помещается в эту область, "обрезается" и становится невидимым. На данный момент единственно доступная форма области - прямоугольник.

clip работает только для абсолютно позиционированных элементов и показывает лишь ту его часть, что входит в прямоугольную область заданную четырьмя значениями от левого и верхнего краёв данного элемента.

Значения:

- **rect(**сверху справа снизу слева**)** - Обрезает элемент сверху, справа, снизу и слева в соответствии с заданными, в скобках, четырьмя значениями, которые могут выражаться в процентах от ширины/длины элемента, пикселях или любых других единицах измерения принятых в CSS.
- **auto** - Оставляет элемент или одну из сторон элемента без изменений (по умолчанию).

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
```

```
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>clip</title>
  </head>
  <body>
    <div style="position: absolute; clip: rect(15px auto auto 15px); width: 100px; height: 100px; background: #c5ffa0; border: double 5px #008000">
      Этот блок частично обрезан слева и сверху
    </div>
  </body>
</html>
```

О том, что такое **position: absolute;** - абсолютно позиционированный элемент и вообще что такое позиционирование мы поговорим в отдельной главе.

Полезные советы:

. В виду того, что свойство **visibility: collapse;** не поддерживается браузером Internet Explorer

рекомендую, при работе с таблицами использовать **display: none**, результат будет аналогичным **collapse** только работающим в IE.

. Свойства **display, visibility** и **clip** в сочетании с скриптами, а так же псевдоэлементами позволяют достичь весьма интересных результатов.

Глава 11

Поплавки.

Эх рыбалка.. хорошо бы было сейчас закинув удочку посидеть на каком ни будь тихом пруду под пенье птиц и шорох камыша.. сидишь себе наблюдаешь за поплавком, потягиваешь пивко, где то неподалёку горит костер.. запах ухи смешавшись с ароматами разнотравья кружит голову и ты окунаешься в нирвану.. в мир спокойствия и гармонии..

Ладно, будем считать, что отдохнули перед очередной порцией информации.. и хотя в этой главе речь пойдёт о поплавках диалоги будут вовсе не о рыбалке..

Итак, свойство **float** - что поделаешь, этому свойству, при переводе с английского не придумали более точного определения, нежели чем "поплавок". А элемент с таким свойством называют "всплывающим"

Теперь побольше конкретики и примеров..

Обтекание элемента

Свойство **float** определяет, по какой стороне будет выравниваться - всплывать элемент, при этом остальные элементы будут обтекать его с других сторон.

Выравнивание элемента происходит по краям родительского элемента или же по краям окна браузера.

Значения свойства **float**:

- **left** - Выравнивает элемент по левому краю, остальные элементы обтекают его справа.
- **right** - Выравнивает элемент по правому краю, остальные элементы обтекают его слева.
- **none** - Выравнивание элемента не задается. (по умолчанию)

Небольшой пример: заставим текст обтекать фотографию по правому краю при этом сам блок с

фотографией "всплывёт" с левой стороны.

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>float</title>
  </head>
  <body>
    <div style="border: solid 1px black;
width: 220px">
      <div style="float: left;">
        <img src="graphics/fon.jpg">
      </div>
      Текст в этом блоке обтекает
рисунок помещённый в контейнер div с
правой стороны, так как сам блок с
рисунком, с помощью свойства float с
значением left, "всплывает" слева.
    </div>
  </body>
</html>
```

А вот в примере ниже показано как могут располагаться элементы относительно друг друга, если свойство **float** присвоено нескольким элементам

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>float</title>
  </head>
  <body>
    <div style="float: right; background:
#c5ffa0; border: solid 1px black; padding:
10px; width: 420px">
      <span style="float: left;
background: #c0e4ff; border: solid 1px
black; padding: 5px; width: 150px">блок
выровнен по левому краю</span>
      <span style="float: right;
background: #c0e4ff; border: solid 1px
black; padding: 5px; width: 150px">блок
```

```
выровнен по правому краю</span>
содержание блока div обтекает
выровненные элементы span слева и
справа
    </div>
  </body>
</html>
```

Запрет обтекания

Свойство **clear** запрещает обтекание элемента с левой и/или правой стороны. Если для элемента установлено обтекание с помощью свойства **float** то **clear** отменяет обтекание данного элемента с указанных сторон.

Значения:

- **both** - Запрещает обтекание элемента с левой и правой стороны.
- **left** - Запрещает обтекание элемента с левой стороны.
- **right** - Запрещает обтекание элемента с правой стороны.
- **none** - Запрет на обтекание элемента не задается.(по умолчанию)

Вот пример где **clear** отменяет действие **float**

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>clear</title>
  </head>
  <body>
    <div style="background:#cc0;
width:150px; height: 50px;
float:left;">Этот блок всплывает
слева.</div>
    <div style="background:#c0c;
width:150px; height: 100px; float:left;">И
этот блок всплывает слева.</div>
    <div style="background:#0c0;
width:300px; height: 80px; clear:left;">А
этот блок отменяет обтекание с левой
стороны.</div>
  </body>
</html>
```

Блочная вёрстка.

В довершение главы акцентирую Ваше внимание на еще одном способе верстки сайта, ранее в учебнике HTML я приводил примеры верстки страницы при помощи **таблиц** (**http://webremeslo.ru/html/glava4.html**) и **фреймов** (**http://webremeslo.ru/html/glava7.html**), теперь имея накопленный багаж знаний, настало время познакомить Вас с блочной верстkой сайта, используя блоки div и свойства CSS.

Разберём классический макет верстки сайта из трёх колонок, а так же "шапки" и "подвала".. в котором правая и левая колонки имеют фиксированную ширину, а центральная колонка "резиновая" т.е. занимает собой всё оставшееся место.

Взгляните на пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
```

```
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Блочная вёрстка</title>
  </head>
  <body style="background: #cc0;
margin:0;">
    <div style="clear:both; background:
#0c0; padding: 5px;">Логотип</div>
    <div style="float: left; background:
#c0c; padding: 5px; width:
170px">Меню</div>
    <div style="float: right; background:
#c0c; padding: 5px; width:
170px">Реклама</div>
    <div style="margin:0 180px;
background: #0cc; padding:
5px;">Основное
содержание<br><br><br><br><br> И
ещё куча текста..</div>
    <div style="clear:both; background:
#0c0; padding: 5px;">Подвал</div>
  </body>
</html>
```

Что было понятно "расшифрую" каждую строчку из примера выше:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Блочная вёрстка</title>
  </head>
```

- Думаю понятно...:) стандартное начало любой страницы..

```
<body style="background: #cc0;
margin:0;">
```

*- Используем **margin:0** для того чтобы обнулить поля в окне браузера.*

```
<div style="clear:both; background:
#0c0; padding: 5px;">Логотип</div>
```

*- Создаём контейнер с будущим логотипом и запрещаем его обтекание с обеих сторон, используя **clear:both**, теперь что бы не случилось, последующие блоки будут идти снизу, а шапка сайта как ей и положено будет располагаться сверху.*

Меню

*- Левая колонка с "Меню" выровнена по левому краю свойством **float: left** и имеет фиксированный размер в **170** пикселей.*

Реклама

*- Правая колонка с рекламой выровнена по правому краю свойством **float: right;** и тоже имеет фиксированный размер **170** пикселей.*

Основное содержание

*- Центральная колонка никак не выравнивается, но занимает своё место по центру, так как имеет широкие поля слева и справа **margin:0 180px**, тем самым не накладываясь на правую и левую колонки. Почему поля 180 пикселей, а не 170 в соответствии с ширенной колонок "Меню" и "Реклама"? Отвечаю: потому что кроме ширины **width: 170px** эти колонки имеют ещё и отступы со всех сторон **padding: 5px** складываем 170+5+5=**180** - такая вот арифметика..*

Подвал

- Ну и "подвал" блок, в котором, как правило, располагаются контактные данные и авторские права, так же

как логотип запрещает на всякий случай обтекание слева и справа ***clear:both*** *и тем самым устремляется вниз страницы.*

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

- Это нужно "расшифровывать"? :)

В данном примере стилевое описание блоков div происходит с помощью атрибута **style** - на самом деле это глупо.. привел такой пример, что бы просто понятнее было разобраться в коде, а вообще при блочной верстке обычно назначают каждому блоку свой идентификатор **id** (**см. Глава 6 Классы и идентификаторы.**)и всё это дело с остальными потрохами выносят в отдельный css файл..

Полезные советы:

. Верстка блоками **div** стремительно набирает популярность, и позволяет сверстать сайт практически любой сложности, однако не всегда стоит полностью отказываться от табличного способа верстки. Зачастую верным решением является комбинированный способ верстки, где грубым "каркасом" сайта является таблица, наполненная блоками **div** которые выполняют более тонкие задачи верстки сложного по структуре сайта.

Глава 12

Позиционирование.

Для начала разберемся с самим понятием позиционирование. Позиционированием называют определение конкретного месторасположения на странице того или иного элемента (бокса). Позиционирование бывает абсолютным, относительным, фиксированным и статическим.

Потороплюсь с примером, ниже будем разбираться, что в нем написано.

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Позиционирование.</title>
  </head>
  <body>
    <div style="float: left; background:
```

```
#c00; border-style: double; padding: 5px; height: 1500px; width: 300px">
        <div style="position: relative; left: 10px; top: 50px; background: #c0c; border-style: double; padding: 5px; width: 200px">Блок позиционирован относительно крёв элемента родителя.</div>
    </div>
    <div style="position: absolute; left:200px; top: 350px; background: #cc0; border-style: double; padding: 5px; width: 200px">Блок абсолютно позицонорован и выведен из общего потока, его положение задаётся от краёв окна браузера. Как видите данный блок может накладываться на другие элементы страницы.</div>
    <div style="position: fixed; left: 150px; top: 150px; background: #0cc; border-style: double; padding: 5px; width: 200px">А это фиксированный блок, тоже выведен из общего потока однако при прокручивании страницы он не меняет своего положения. Ранние версии браузера Internet
```

```
Explorer игнорируют данное
свойство.</div>
  </body>
</html>
```

Итак, для того, что бы позиционировать какой либо элемент к нему применяют свойство **position** и одно из его возможных значений:

. **absolute** - Абсолютное позиционирование элемента.
. **relative** - Относительное позиционирование элемента.
. **fixed** - Фиксированное позиционирование элемента.
. **static** - Статическое позиционирование элемента. (Элемент отображаются как обычно.)
. **inherit** - Наследует значение элемента родителя.

Теперь давайте углубляться..

Абсолютное позиционирование.

Абсолютно позиционированный элемент (**position: absolute**)

выводится из общего потока и несмотря на другие элементы и их взаимное расположение, занимает указанное место на странице от края/краёв окна браузера. При таком способе позиционирования один элемент может накладываться поверх другого.

Для того чтобы позиционировать элемент от края/краёв окна браузера, нам понадобятся следующие свойства CSS:

. **bottom** - Расстояние от нижнего края окна браузера.
. **left** - Расстояние от левого края окна браузера.
. **right** - Расстояние от правого края окна браузера.
. **top** - Расстояние от верхнего края окна браузера.

Данные расстояния могут быть заданы в пикселях, процентах или любых других принятых единицах

измерения CSS, значение по умолчанию - **auto**.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Абсолютное
позиционирование.</title>
  </head>
  <body>
    <div style="background: #0f0; border:
#000000 2px solid; padding: 5px; margin:
10px; width: 300px; height:
200px;"><h1>Блок №1</h1></div>
    <div style="background: #00f; border:
#000000 2px solid; padding: 5px; margin:
10px; width: 500px; height:
100px;"><h1>Блок №2</h1></div>
    <div style="position: absolute;
left:200px; top: 100px; background:
#f00; border: #000 2px solid; padding:
5px; width: 200px; height:
```

```
200px;"><h1>Блок №3</h1> Данный блок абсолютно позиционирован! <br><br> Блоки 1 и 2 никак не влияют на его месторасположение.</div>
  </body>
</html>
```

Как видите, в примере третий блок вышел из общего потока элементов и живёт по своим собственным правилам, остальная разметка страницы никак не влияет на месторасположение данного блока.

Относительное позиционирование.

Относительное позиционирование (**position: relative**) определяет место элемента относительно краёв элемента родителя и не выводится из общего потока.

Так же как и в случае с абсолютным позиционированием расстояния от края/краёв родительского элемента задаётся с помощью свойств: **bottom, left, right, top.**

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Относительное
позиционирование.</title>
    <style type="text/css">
      h1 {
        color: #008000;
        font-size:20px
      }
      div.blok1 {
        background: #c0e4ff;
        border: #000000 2px solid;
        padding: 5px;
        width: 500px;
        height: 400px;
      }
      div.blok2 {
        position: relative;
        left: 150px;
        background: #ffa0c5;
```

```
            border: #000 2px solid;
            padding: 5px; width: 250px;
            height: 200px;
        }
    </style>
  </head>
  <body>
    <div class="blok1">
      <h1>Элемент родитель - блок №1</h1>
      <div class="blok2">
        <h1>Блок №2</h1>
        Данный блок позиционирован относительно левого края элемента родителя.
      </div>
    </div>
  </body>
</html>
```

В случае если элемент родитель явно не задан, то отчет ведётся от края/краёв окна браузера.

Фиксированное позиционирование.

Фиксированное позиционирование **(position: fixed)** похоже на абсолютное, элемент выводится из общего потока, его координаты рассчитываются от краёв окна браузера, но при прокрутке страницы элемент не меняет своего положения.

Пример:

```
Файл style.css
h1 {
   color: #800;
   text-align: center;
   font-size:30px
}
h2 {
   color: #088;
   text-align: center;
   font-size:18px
}
div.blok1 {
   background-image: url(fon.gif);
   border: #000 2px solid;
   padding: 5px;
   width: 800px;
   height: 4000px
```

```
}
div.blok2 {
   position: fixed;
   left: 250px;
   top: 300px;
   border: #080 6px double;
   padding: 5px;
   width: 300px
}
```

Файл index.html

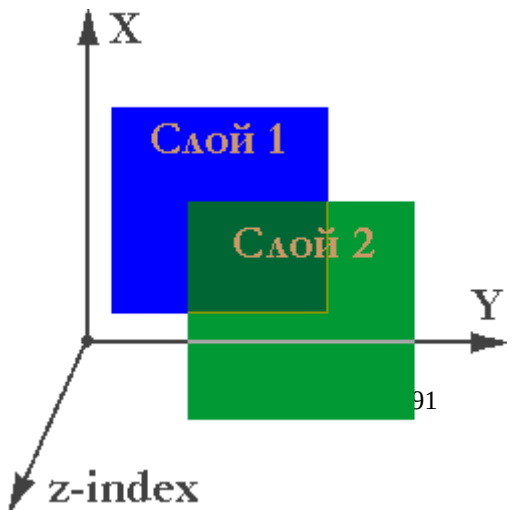
```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
      <title>Фиксированное
позиционирование.</title>
      <link rel="stylesheet" href="style.css"
type="text/css">
   </head>
   <body>
      <div class="blok1">
         <h1>Какая то страничка</h1>
```

```
    </div>
    <div class="blok2">
      <h2>Фиксированный блок с навязчивой рекламой</h2>
      <h1>Купи слона!!</h1>
      Фиксированный блок, выведен из общего потока элементов, при прокручивании страницы он не меняет своего положения. Ранние версии браузера Internet Explorer игнорируют данное свойство.
    </div>
  </body>
</html>
```

P.S. Ранние версии браузера Internet Explorer игнорируют данное свойство и элемент выводится на странице так как будто его вовсе не позиционировали.

z-index

Как уже говорилось, выше позиционированные элементы могут



накладываться один поверх другого, имитируя тем самым некую трёхмерность страницы, где каждый последующий наложенный друг на друга элемент является слоем.

Свойство **z-index** позволяет веб-мастеру управлять позициями этих слоёв в глубину экрана (по оси Z.) , другими словами позволяет указывать браузеру какие элементы следует показывать на переднем плане, а какие на заднем.

Значения свойства **z-index**:

- **auto** - Элементы накладываются друг на друга в том порядке, каком они указаны в коде HTML. (по умолчанию).
- **целое число** - Чем выше данное значение, тем более высокую позицию занимает элемент по отношению к тем элементам, значение которых ниже.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
```

```
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>z-index</title>
  </head>
  <body>
    <div align="center" style="position:
absolute; z-index:5; width: 350px;
height: 100px; top: 120px; left: 0px; color:
#0000ff; font-size:100px">z-index</div>
    <div style="position: absolute; z-
index:3; width: 150px; height: 150px;
top: 0px; left: 100px; background-color:
#ff00ff"> </div>
    <div style="position: absolute; z-
index:4; width: 150px; height: 150px;
top: 100px; left: 0px; background-color:
#ff0000"> </div>
    <div style="position: absolute; z-
index:2; width: 150px; height: 150px;
top: 100px; left: 200px; background-
color: #ffff00"> </div>
    <div style="position: absolute; z-
index:1; width: 150px; height: 150px;
top: 200px; left: 100px; background-
```

```
color: #00ff00"> </div>
  </body>
</html>
```

Числовое значение **z-index** может быть и отрицательным, однако не все браузеры правильно интерпретируют отрицательные значения.

Еще следует отметить, что при равном значении **z-index**, на переднем плане находится тот элемент, который в коде HTML идет ниже остальных.

Это же правило действует при **z-index** равным **auto** или же в том случае если бы данное свойство и вовсе не применялось к тем или иным элементам.

Ну и какой смысл, спросите Вы, вообще использовать **z-index** если можно просто в нужном порядке расположить элементы в HTML коде?

Покажу на примере:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
```

```
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>z-index</title>
    <style type="text/css">
      body {background-color: #c5ffa0}
      a {
        position: absolute;
        z-index: auto;
        top: 100px;
        border: #000000 1px solid;
      }
      a:hover {
        position: absolute;
        z-index:1;
        top: 80px;
        border: #800000 1px solid;
      }
    </style>
  </head>
  <body>
    <h2>Проведите курсором по
картам</h2>
    <a href="#" style="left: 10px"><img
src="karta1.gif"></a>
```

```
    <a href="#" style="left: 30px"><img src="karta2.gif"></a>
    <a href="#" style="left: 50px"><img src="karta3.gif"></a>
    <a href="#" style="left: 70px"><img src="karta4.gif"></a>
    <a href="#" style="left: 90px"><img src="karta5.gif"></a>
    <a href="#" style="left: 110px"><img src="karta6.gif"></a>
    <p style="position: absolute; left: 10px; top: 250px;">В злосчастном IE6, в отличии от других браузеров, псевдокласс <b>:hover</b> работает только для ссылок, поэтому в качестве контейнера для рисунков/карт использую никуда не ведущую ссылку. (<b>a href="#"</b>)</p>
    <p style="position: absolute; left: 10px; top: 300px;">Впрочем, IE можно "надурить" используя скрипты, просто не хочу лишний раз выходить за рамки обучения CSS.</p>
  </body>
</html>
```

Как видите свойство **z-index** незаменимо там, где присутствует некая динамика.

Полезные советы:

Попугаю немного:

- Списки, созданные с помощью тега **select**, в Internet Exolorer отображаются поверх других элементов, несмотря на то, какое указано значение **z-index**.
- В браузере Netscape 4.x поля форм всегда отображаются поверх всех других элементов, независимо от значения **z-index**..
- Браузер Netscape 4.x относительно **(position: relative)** не позиционирует поля форм, списки, изображения, таблицы и их ячейки, а абсолютное позиционирование **(position: absolute)** не работает со списками и элементами форм.
- Ранние версии Internet Explorer, до Internet Explorer 5.5, игнорируют свойство **z-index** примененное к

фреймам **(frame)** и плавающим фреймам **(iframe)**.

- В IE 4 абсолютное позиционирование **(position: absolute)** примененное к ссылкам и спискам не работает, ссылки так вообще теряют свою функциональность.. а относительное позиционирование **(position: relative)** не дружит с ячейками таблицы (**td** и **th**).
- Про фиксированное позиционирование **(position: fixed)** и ранние версии Internet Explorer уже говорил.. не работает фиксация блоков в ранних IE и все тут..

Такие дела.. ну хочу заметить, что поклонников браузера IE (домохозяек, для которых Интернет и встроенный в комплектацию Windows Internet Explorer это одно и тоже) остаётся с каждым днем всё меньше и меньше тем более его старых версий, так что можно и рискнуть.. если по-другому никак..

Глава 13

Стиль списка.

В девятой главе **учебника HTML** (**http://webremeslo.ru/html/glava9.html**) мы с Вами уже знакомились с таким немаловажным элементом как список и действительно хорошее средство для структуризации данных. Однако списки организованные одними средствами HTML весьма убоги в плане дизайна и не радуют глаз человека.

В этой главе мы немного поколдуем над списками с помощью свойств CSS. Так мы будем говорить о стеле списка, то для обучения Вам понадобится базовая информация о тегах: **   <dl> <dt> <dd>** - изложенная в учебнике HTML глава 9 **"Списки"** (**http://webremeslo.ru/html/glava9.html**) - рекомендую освежить в голове информацию о данных элементах, прежде чем приступать к работе.

Ну а если в голове и так свежо тогда начнем!

Вид маркера в списке.

Если вы помните, то в чистом HTML вид маркера в списке определял атрибут **type** и одно из его возможных значений, в CSS данную задачу берёт на себя свойство: **list-style-type** которое, в свою очередь, тоже имеет свои стандартные значения определяющие вид маркера как всего списка сразу, так и его отдельного "пункта".

Значения **list-style-type:**

. **disc** - Диск. (по умолчанию для **<ul>**)
. **circle** - Полый круг.
. **square** - Квадрат.
. **decimal** - Арабские цифры. (по умолчанию для **<ol>**)
. **lower-roman** - Строчные римские цифры.
. **lower-alpha** - Строчные буквы.

- **upper-roman** - Заглавные римские цифры.
- **upper-alpha** - Заглавные буквы.
- **none** - Маркер отсутствует.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Вид маркера в списке</title>
  </head>
  <body>
    <ul style="list-style-type: square">
      <li>Пункт 1.
      <li>Пункт 2.
      <li style="list-style-type:
circle">Пункт 3 (особенный).
    </ul>
    <ul style="list-style-type: upper-
roman">
      <li>Пункт 1.
      <li>Пункт 2.
      <li>Пункт 3.
```

```
    </ul>
  </body>
</html>
```

Пользовательский маркер рисунок.

Наиболее интересным CSS инструментом для работы со стилем списка является, на мой взгляд, возможность вместо стандартных "скучных" маркеров описанных выше использовать свои собственные нестандартные изображения - небольшие рисунки, вписывающиеся в общий дизайн Вашего сайта.

Эту задачу выполняет свойство **list-style-image** которое определяет в качестве маркера списка некое графическое изображение с указанием пути к нему.

Значений данного свойства всего два:

. **none** - Отменяет графическое изображение маркера.

. **url** - Путь к файлу с рисунком маркера.

Путь к рисунку после **url** указывается в круглых скобках.

Вот так:

list-style-image: url(graphics/marker.gif)

- Такая запись будет говорить о том, что рядом с документом есть папка graphics в которой лежит файлик-рисунок: "  " - под названьем marker.gif

Теперь попрубуем сделать так, что бы каждый пункт нашего списока был промаркерован этим рисунком. Смотрим пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Нестандартный маркер-
```

```
рисунок</title>
  </head>
  <body>
    <ul style="list-style-image:
url(graphics/marker.gif)">
      <li>Первый любимый пункт.
      <li>Второй любимый пункт.
      <li>И не менее любимый третий
пункт.
    </ul>
  </body>
</html>
```

Стиль обтекания маркера списком.

Свойство **list-style-position** указывает браузеру на то, как следует отображать текст в списке относительно его маркеров. По умолчанию маркеры находятся в стороне от текста списка, но можно сделать так, что они будут обтекаться текстом.

Возможных значений свойства **list-style-position** всего два:

. **outside** - Маркер находится в стороне от списка.(по умолчанию)
. **inside** - Маркер обтекается текстом.

Пример для наглядности:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Обтекание маркера
текстом</title>
    <style type="text/css">
      body {
        margin: 0px;
        background: #e8e8e8
      }
      div {
        width: 300px;
        height: 200px;
        float:left;
        margin: 10px;
        padding: 10px;
        border: RGB(25, 125, 25) 2px
ridge;
```

```
        background: #fff
      }
      h3 {
        text-align: center
      }
    </style>
  </head>
  <body>
    <div>
      <h3>Здесь маркер обтекается текстом:</h3>
      <ul style="list-style-position:inside">
        <li>Пункт, в котором говорится о том, что хорошо бы было сделать, что-то там, где это что-то еще не сделано.
        <li>Пункт, в котором говорится о том, что неплохо бы было сделать, нечто там, где это нечто еще не сделано.
      </ul>
    </div>
    <div>
      <h3>А здесь нет:</h3>
      <ul style="list-style-
```

```
position:outside">
            <li>Пункт, в котором говорится
о том, что хорошо бы было сделать,
что-то там, где это что-то еще не
сделано.
            <li>Пункт, в котором говорится
о том, что неплохо бы было сделать,
нечто там, где это нечто еще не
сделано.
        </ul>
      </div>
   </body>
</html>
```

list-style

Базовое свойство **list-style** используется, когда стилю списка необходимо одновременно присвоить несколько значений. Может иметь от одного до трёх значений из свойств применяемых к стилю списка, в любой последовательности через пробел.

Все три свойства и их возможные значения мы рассмотрели выше,

поэтому повторятся не буду, а просто перечислю их:

. **list-style-type** - Вид маркера в списке
. **list-style-image** - Нестандартный маркер рисунок
. **list-style-position** - Стиль обтекания маркера списком

Если в голове остались какие то пробелы можете вернуться и перечитать.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Стиль списка</title>
  </head>
  <body>
    <div style="width: 250px;">
      <ul style="list-style:
url(graphics/marker.gif) inside">
```

```
        <li>- Этот список использует в
качестве маркера рисунок.
        <li>- Текст этого списка
обтекает маркер.
      </ul>
    </div>
  </body>
</html>
```

Полезные советы:

- Списки наряду с заголовками, выделенным текстом и картинками, являются элементами, которые приковывают взгляд посетителя при беглом просмотре страницы. Умелое и ненавязчивое размещение таковых элементов залог того, что посетитель окунётся в чтение Вашего документа более глубоко.
- Так как списки являются великолепным инструментом для структуризации данных, а свойства CSS позволяют создать практически любой их дизайн, используйте списки, помимо их основного назначения, в качестве "меню" - навигации по сайту,

где каждый пункт является ссылкой на ту или иную страницу Вашего сайта.

Глава 14

Полоса прокрутки.

Коротенькая и совсем несложная глава про цвет полосы прокрутки.

Для того чтобы изменить дизайн полосы прокрутки, раскрасить её на свой лад, необходимо воспользоваться свойствами из семейства **scrollbar**, присвоив их нужному элементу.

В качестве элементов могут выступать любые блоки, в которых гипотетически может появляться полоса прокрутки ** , <textarea>, <iframe>** и т.д.. в том числе и сама страница - тег **<body>**.

В таблице ниже перечислены все свойства CSS из семейства **scrollbar**, а также показан результат применения этих свойств.

Свойство полосы прокрутки:

scrollbar-3dlight-color: #ff0000; - Цвет

верхней и левой тени ползунка и кнопок.

scrollbar-arrow-color: #ff0000; - Цвет стрелок на кнопках полосы прокрутки.

scrollbar-base-color: #ff0000; - Базовый цвет полосы прокрутки.

scrollbar-darkshadow-color: #ff0000; - Цвет нижней и правой тени.(dark shadow - темная тень)

scrollbar-face-color: #ff0000; - Цвет лицевой части полосы прокрутки.

scrollbar-highlight-color: #ff0000; - Цвет подсветки полосы прокрутки.

scrollbar-shadow-color: #ff0000; - Цвет тени полосы прокрутки (shadow-тень).

scrollbar-track-color: #ff0000; - Цвет дорожки для полосы прокрутки.

Ну вот Вам ещё и рисунок для наглядности:

Возможные значения свойств из семейства **scrollbar**:

. **#ff0000** - Шестнадцатеричное значение цвета RGB.

. **red** - Именное значение цвета.

. **RGB(255, 0, 0)** - Значение цвета RGB.

. **inherit** - применяется значение родительского элемента.(по умолчанию)

Пример:

```
Файл primer3.css
body{
   background-color: #fff8dc;
   scrollbar-3dlight-color: #ffebcd;
   scrollbar-arrow-color: #ffffff;
   scrollbar-base-color: #ffebcd;
   scrollbar-darkshadow-color:
#f5f5dc;
   scrollbar-face-color: #b8860b;
   scrollbar-highlight-color: #f5f5dc;
   scrollbar-shadow-color: #f5f5dc;
   scrollbar-track-color: #f5f5dc;
}
```

```
div{
   padding: 20px;
   overflow: auto;
   width: 200px;
   height: 100px;
   border: solid 2px #deb887;
   background-color: #ffffff;
}
```

Файл index.html

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN">
<html>
   <head>
      <title>Полоса прокрутки</title>
      <link rel="stylesheet"
href="primer3.css" type="text/css">
   </head>
   <body>
      <div>
         <h3 align="center">Диктант</h3>
         На террасе, близ конопляника,
небезызвестная вдова Агриппина
Саввична, потчевала, отставного
коллежского асессора Аполлона
```

```
Карповича, моллюсками, под
аккомпанемент виолончели.
      </div>

<br><br><br><br><br><br><br><br><br
><br><br><br><br><br><br>

<br><br><br><br><br><br><br><br><br
><br><br><br><br><br><br>

<br><br><br><br><br><br><br><br><br
><br><br><br><br><br><br>

<br><br><br><br><br><br><br><br><br
><br><br><br><br><br><br>
  </body>
</html>
```

Кроссбраузерность свойства scrollbar.

Хочу отметить, что свойства CSS из семейства **scrollbar** являются расширением спецификации CSS2, введенным компаний Microsoft, и реализованным в браузерах Internet Explorer начиная с версии IE 5.5,

соответственно другие браузеры, на тот момент, полностью игнорировали данное свойство. Но прошло время и многие браузеры стали подстраиваться под это дополнение к спецификации от Microsoft.

Однако, до сих пор всё не так уж гладко! Так что во многих браузерах стиль полосы прокрутки или игнорируется или работает не совсем корректно.. например многие игнорируют данное свойство для тега **<body>** или не понимают некоторые отдельные свойства скроллбара.

Ниже приведу несколько советов, которые возможно помогут решить данную проблему в тех или иных браузерах. Употребляю слово "возможно" потому что очень трудно отследить в динамике за всеми изменениями разных браузеров. Мир не стоит на месте каждый день, там или здесь, что-то меняется! Жаль вот только что кроссбраузерность всегда остаётся головной болью!

Но что то я заговорился.. Итак, ряд "таблеток":

1. По возможности используйте вот такой заголовок **<!DOCTYPE>**:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

2. Указывайте свойства скроллбара не только для тега **<body>** но и для тега **<html>**

Вот так:

```
html,body{
scrollbar-3dlight-color: #ffebcd;
scrollbar-arrow-color: #ffffff;
}
```

3. Попробуйте, если это более менее рационально для Вашего сайта, внедрять стили полосы прокрутки используя атрибут **style**, а не одноименный тег в "голове" документа или внешний css файл

Вот так:

** блок **

Возможно, данные рекомендации сработают в некоторых браузерах, в любом случае в IE (начиная с версии 5.5) всё будет работать корректно, а раз хоть где-то работает это уже хорошо!

Полезные советы:

- Так как **scrollbar** является одним из важных элементов интерфейса, постарайтесь не вести пользователя в заблуждение, используя совсем уж нестандартный дизайн полосы прокрутки.
- Если Вам прямо таки совсем необходим нестандартный скроллбар, не только нестандартный цвет, он и нестандартная форма, да и еще и работающий во всех браузерах используйте скрипты, но не забывайте про первый совет!!

. В браузере внешний вид полосы прокрутки черпается из настроек Windows и например у меня, она имеет вот такой вид:

Но как только Вы примените к полосе прокрутки любое свойство из семейства **scrollbar**, связь с настройками Windows потеряется, точнее станет "по умолчанию" и полоса приобретёт вот такой вид:

Глава 15

Псевдоклассы.

Псевдоклассы - это особые свойства, которые позволяют менять стиль элемента в зависимости от действий пользователя, а так же положения этого элемента (тега) в общем потоке документа, что позволяет разбавить дизайн страницы некой динамикой и логикой. Классическим примером применения псевдоклассов является ссылка, которая меняет свой цвет при наведении на неё курсором.

Вот список всех псевдоклассов:

. **hover** - Стиль элемента на который наведён курсор мыши.
. **active** - Стиль для ссылки которая становится активной, но переход по ней еще не совершен.
. **visited** - Стиль для недавно посещённой ссылки.
. **link** - Стиль для нечасто посещаемой ссылки.

. **focus** - Стиль элемента находящегося в фокусе.
. **first-child** - Стиль первого дочернего элемента.
. **lang** - Определяет язык, который используется в фрагменте документа.

О каждом псевдоклассе мы отдельно поговорим ниже, а сейчас пару слов о синтаксисе.

Для того чтобы применить тот или иной псевдокласс к элементу и определить его стиль нужно следовать следующим правилам синтаксиса:

a:hover { color:#ff0000}

где:

a - элемент (селектор), а проще тег к которому мы решили применить псевдокласс в нашем случае это ссылка.
:hover - после двоеточия собственно нужный нам псевдокласс.

{color:#ff0000} - ну и блок объявления стилей в фигурных скобках.

А вся эта запись вместе будет говорить о том, что если навести курсором на такую ссылку - то она покраснеет.

Так собственно мы подошли к первому, самому популярному, псевдоклассу **hover**.

Псевдокласс hover.

Как Вы уже, наверное, догадались, псевдокласс **hover** активизируется в том случае, если на элемент наведен курсор.

Ну а раз уж догадались просто покажу пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Псевдоклассы</title>
```

```
    <style type="text/css">
      .menu {
        display: block;
        width: 180px;
        background-color:#fff8dc;
        color:#008;
        font-size: 16px;
        text-decoration: none;
        padding: 3px;
      }
      .menu:hover {
        display: block;
        width: 180px;
        background-color:#b8860b;
        color:#fff;
        padding: 3px;
        font-size: 16px;
        text-decoration: none;
      }
      tr:hover {
        background-color:#b8860b;
      }
    </style>
  </head>
  <body>
    <p>Чем Вам не блок навигации по
```

```
сайту?</p>
      <a href="#"
class="menu">Главная</a>
      <a href="#" class="menu">Карта
сайта</a>
      <a href="#" class="menu">Купить
слона</a>
      <a href="#" class="menu">Продать
слона</a>
      <a href="#" class="menu">Взять
слона на прокат</a>
      <hr>
      <p>А вот ещё один
распространенный трюк.. строки в
таблице подсвечиваются при
наведении на них курсором.</p>
      <table border="1" bordercolor="#ccc"
width="600" cellpadding="1"
cellspacing="0">
         <tr>

<td>Иванов</td><td>+</td><td> <
/td><td> </td><td>+</td><td>&nbs
p;</td> <td> </td>
         </tr>
         <tr>
```

```
<td>Петров</td><td> </td><td>+<
/td><td> </td><td> </td><td
>+</td> <td> </td>
        </tr>
        <tr>

<td>Сидоров</td><td> </td><td>
 </td><td>+</td><td> </td>
<td> </td><td>+</td>
        </tr>
      </table>
   </body>
</html>
```

Несколько слов к примеру выше..

Как Вы наверное заметили в качестве селектора псевдокласса может выступать не только какой либо элемент - тег, но и класс или идентификатор. Так в примере к классу **.menu** применён псевдокласс **hover** и синтаксис приобретает следующий вид:

```
.menu:hover { color:#ff0000;}
```

Не запутались в терминологии?

Простыми словами мы сказали браузеру что мол подсвечивай красным только те ссылки которые находится в навигационном блоке (выведены в класс**.menu**), а остальное оставь как есть!

Псевдокласс **hover** может быть применён к любому элементу, выводимому на экран, так в нашем примере, для того чтобы организовать подсветку строк таблицы мы применили его к тегу **<tr>**. Однако следует отметить, что браузер Internet Explorer 6 и его более ранние версии поддерживает псевдокласс **hover** только для ссылок - тег **<a>**, так что, к примеру, строки таблицы, при наведении на них курсора, в браузерах IE6 и ниже подсвечиваться не будут!

Псевдоклассы и ссылки.

Рассмотрим сразу три псевдокласса созданных для работы со ссылками.

. **active** - Стиль активной ссылки.
. **visited** - Стиль для недавно посещённой ссылки.
. **link** - Стиль для нечасто посещаемой ссылки.

Сначала покажу пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Псевдоклассы и
ссылки</title>
    <style type="text/css">
      a:link {color:#0000ff}
      a:active {color:#ff0000}
      a:visited {color:#008000}
    </style>
  </head>
  <body>
    <a href="#1">Ссылка на первый
якорь</a>
    <a href="#2">Ссылка на второй
якорь</a>
```

```
    <a href="#3">Ссылка на третий якорь</a>
    <hr>
    <a name="1"><h4>Первый якорь</h4></a>
    <p>Псевдокласс link, в этом примере, указывает, что все не посещенные ссылки должны отображаться синим цветом.</p>
    <a name="2"><h4>Второй якорь</h4></a>
    <p>Попробуйте нажить и удерживать на одну из не посещенных ссылок, чтобы увидеть для работу псевдокласса active.</p>
    <a name="3"><h4>Третий якорь</h4></a>
    <p>visited - псевдокласс который, в этом примере, говорит о том, что все посещенные ссылки должны отображаться зеленым цветом.</p>
  </body>
</html>
```

Теперь расскажу более подробно.

Псевдокласс **active** присваивает ссылке определённый стиль в тот момент, когда эта ссылка активна, то есть в тот момент, когда пользователь нажал на ссылку, но еще не отпустил кнопку мыши. Короче **active** - это стиль ссылки в момент клика по ней.

Браузеры некоторое время помнят, на какие ссылки нажимал пользователь в последнее время, так вот, псевдокласс **visited** указывает стиль ссылки, которая недавно посещалась пользователем.

Псевдокласс **link** описывает стиль ссылки, которая ранее не посещалась пользователем. Надо отметить, что никакой ощутимой разницы между записью **a {...}** и **a:link {...}** нет, так что применение данного псевдокласса в этом случае ровным счетом ничего не меняет.

Все три вышеперечисленных псевдокласса предназначены для работы с ссылками, однако

псевдокласс **active** может быть применён к любому элементу - работать будет везде, кроме браузера Internet Explorer 6 и ниже.

Псевдокласс focus.

Данный псевдокласс определяет стиль элемента, если тот находится в фокусе. Теперь собственно о том, что такое фокус?.. как в случае с оптикой и иллюзией, слово фокус в CSS обозначат, что некий объект, а точнее элемент, находится в центре внимания наблюдателя - пользователя. Такими элементами могут быть теги **<a> <input> <select>** и **<textarea>**.

Посмотрите на пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Псевдокласс focus</title>
```

```
    <style type="text/css">
input:focus {color: red}
    </style>
  </head>
  <body>
    <form>
      <input type="text" value="Введите текст в эту форму" size="30">
    </form>
    <p>Вели? а теперь щелкните по пустому месту на экране чтобы форма потеряла фокус.</p>
  </body>
</html>
```

В примере текст в текстовом поле **<input>** изначально чёрный, но как только элемент получает фокус - то есть тогда когда пользователь кликнет по данному полю и начнет набирать текст, он окрасится красным.

Вот и весь фокус-покус..

Псевдокласс first-child.

Думаю, в ходе обучения CSS Вы уже поняли, что все элементы можно

определить как родительские или дочерние и что элемент родитель может содержать в себе несколько дочерних элементов, ну например:

** **- *блок родитель*
  ** ** *- первый дочерний элемент*
  ** ** *- второй дочерний элемент*
** **

Так вот псевдокласс **first-child** определят стиль первого дочернего элемента находящегося в родительском контейнере.

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Псевдокласс first-child</title>
    <style type="text/css">
      div {
        margin: 20px;
        padding: 30px;
```

```
        background-color: #c0e4ff;
        border: 2px solid #008000
      }
      p {
        color: #555;
        background-color: #dcdcdc;
        border: 2px solid #555
      }
        p:first-child {
        color: #f00;
        background-color: #c5ffa0;
        border: 2px solid #008000
      }
    </style>
  </head>
  <body>
    <div>
      <p>первый дочерний параграф - он выделен псевдоклассом first-child</p>
      <p>второй дочерний параграф</p>
      <p>третий дочерний параграф</p>
    </div>
    <div>
```

```
        <p>и здесь первый дочерний
параграф выделен хотя блок родитель
уже другой</p>
        <p>второй дочерний
параграф</p>
    </div>
  </body>
</html>
```

Обратите внимание на тот факт, что если бы в блоке родителе перед дочерними параграфами стоял любой другой тег, заголовок **<h1>** к примеру, то псевдокласс **first-child** уже не действовал бы к первому параграфу.. так как хоть параграф то он и первый, но элемент в блоке родителе уже второй.

Применяют данный псевдокласс в тех случаях , если требуется задать разный стиль для первого и последующих элементов, например сделать "буквицу"- одну единственную в начале документа или обозначить первый абзац текста на всех

страницах сайта , выделить первый пункт в списках...и т.д.

Язык текста.

Псевдокласс **lang** определяет язык текста того или иного элемента или документа в целом.

Если Вы помните из курса HTML, язык документа определяют атрибуты: **charset** - **кодировка документа** и **content** - **язык документа** (**http://webremeslo.ru/html/glava10.html#charset**) для тега **<meta>**.

```
<meta http-equiv="Content-Type" Content="text/html; Charset=Windows-1251">
<meta http-equiv="Content-Language" Content="ru">
```

Так вот для того чтобы определить язык отдельно взятого текстового блока используют псевдокласс **lang()** - в круглых скобках которого собственно и указывается язык.

Язык может быть:

. **ru** - Русский
. **en** - Английский
. **de** - Немецкий
. **fr** - Французский
. **it** - Итальянский

Всё вместе пишется так:

span:lang(en) {font-style: italic}

- здесь мы указали, что текст взятый в контейнер **** английский и что он должен отображаться курсивом.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Псевдокласс lang</title>
    <meta http-equiv="Content-Type"
Content="text/html; Charset=Windows-
1251">
    <meta http-equiv="Content-
Language" Content="ru">
    <style type="text/css">
```

```
        p:lang(ru) {color: #00f;}
        p:lang(en) {color: #f00;}
      </style>
    </head>
    <body>
      <p lang="ru">Русский текст
выделен синим</p>
      <p lang="en">English text is chosen
red</p>
    </body>
</html>
```

Обратите внимание, что в теге, в нашем случая параграфе, мы указываем с помощью атрибута **lang** используемый язык и его стиль: ** текст ** прописанный в блоке CSS.

Так же псевдокласс **lang** позволяет определять вид кавычек для цитат (тег **<q>**) с помощью значения **quotes** - кавычки. В России привычно использовать двойные кавычки, в других странах дела обстоят иначе.

Пишется так:

q:lang(en) {quotes: "\201C" "\201D"}

В фигурных скобках после значения **quotes** указывается юникод символа/ов или просто смвол/ы для открывающей и закрывающей кавычки.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Псевдокласс lang.
Цитаты.</title>
    <meta http-equiv="Content-Type"
Content="text/html; Charset=Windows-
1251">
    <meta http-equiv="Content-
Language" Content="ru">
    <style type="text/css">
      div {
        color:#000;
        font-size: 24px;
      }
```

```
        q:lang(en) {quotes: "\201C" "\201D"}
        q:lang(de) {quotes: "\201E" "\201C"}
        q:lang(fr) {quotes: "\00AB" "\00BB"}
        q:lang(ru) {quotes: "ёклмн.." "..ёпрст"}
    </style>
  </head>
  <body>
    <div>
        <p>Обратите внимание на вид кавычек для цитат:<p>
        <q>Цитата по умолчанию</q><br>
        <q lang="fr">Французская цитата</q><br>
        <q lang="de">Немецкая цитата</q><br>
        <q lang="en">Английская цитата</q><br>
        <q lang="ru">Русская цитата</q><br>
    </div>
```

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

Полезные советы:

. Псевдоклассы нельзя внедрять в HTML документ с помощью атрибута **style**. Можно только с помощью тега **<style>** в голове документа или внешнего CSS файла.

. Если селектор, какого либо псевдокласса, явно не указывать, а написать вот так, например:

:hover {color: #ff0}

- то это будет значить, что действие данного псевдокласса будет распространятся на все элементы документа.

. Браузер IE6 и ниже игнорирует практически все псевдоклассы.

Глава 16

Псевдоэлементы.

Псевдоэлементы - это особый вид свойств CSS, которые позволяют работать не над самим элементом, а над его отдельной частью.

Вот перечень всех псевдоэлеметов:

- **:first-letter** - Стиль первой буквы текстового блока
- **:first-line** - Стиль первой строки текстового блока
- **:after** - Добавляет содержимое после элемента.
- **:before** - Добавляет содержимое до элемента.
- **::selection** - Стиль выделенного пользователем текста.

Так же как и в случае с псевдоклассами, псевдоэлементы используются согласно следующего синтаксиса:

```
p:first-letter { color:#ff0000}
```

где:

p - селектор, к которому применяется псевдоэлемент.
:first-letter - после двоеточия собственно псевдоэлемент.
{color:#ff0000} - блок объявления стилей в фигурных скобках.

В данном случае мы указали, что первая буква всех параграфов будет красного цвета.

Ну что ж давайте пробежимся по перечисленным псевдоэлементам.

Стиль первой буквы.

Псевдоэлемент **first-letter** задаёт стиль первой буквы в каком либо текстовом блоке, проще говоря, без особых усилий позволяет сделать "буквицу". Вы заметили, что на этом сайте первые буквы в абзацах отличны цветом и размером от других букв? - это и есть пример работы псевдоэлемента **first-letter**.

Ну а если Вам этого примера мало можете взглянуть на ещё один:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Псевдоэлемент first-
letter.</title>
    <style type="text/css">
      p {
        font: 16px Arial;
       }
       p:first-letter {
        font: bold 24px Verdana;
        color:#ff0000
       }
    </style>
  </head>
  <body>
    <p>Мало кто знает, как много надо
знать для того, что бы знать, как мало
мы знаем.<p>
    <p>Осмысливая мысли, в смысле
смысла, есть смысл, помыслить о
```

```
немыслимом..<p>
  </body>
</html>
```

В примере выше мы изменили размер, шрифт, цвет и жирность первой буквы для всех параграфов на странице.

Стиль первой строки.

Псевдоэлемент **first-line** определяет стиль первой строки в текстовом блоке.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Псевдоэлемент first-
line.</title>
    <style type="text/css">
      body {background-color: #fffacd}
      p {color: #005}
      p:first-line {
```

```
            font: 16px Arial;
            color: #f00
        }
    </style>
  </head>
  <body>
    <p>Если бы при приеме на
работу… … …</p>
    <p>Однажды молодой сисадмин …
… …</p>
  </body>
</html>
```

Честно признаться, я не знаю в каких ситуациях действительно целесообразно применять псевдоэлемент **first-line**, хотя не исключаю, что такие ситуации бывают. Обусловлено это в первую очередь тем, что в зависимости от расширения экрана, размера шрифта, интервала между словами и символам и т.д. первая строка в текстовом блоке будет разной длины, что не позволяет веб мастеру полностью

контролировать стиль данного текстового блока.

Контент.

Псевдоэлементы **after** и **before** предназначены для "врезки" в страницу сайта контента который изначально неуказан в HTML документе. Вставляется содержание перед (**:before**) или после (**:after**) какого либо элемента с помощью свойства **content**, которое собственно и определяет содержимое для вставки.

Всё вместе пишется так:

p:after {content: "Конец, а кто слушал молодец!!"; }

Теперь после каждого параграфа будет добавляться надпись: "Конец, а кто слушал молодец!!"

Значением свойства **content** может быть:

. **"текст"** - собственно любой текст или символы.

. **"\0410"** - юникод.

. **url(путь)**- адрес какого либо объекта.

. **open-quote** - открывающая кавычка.

. **close-quote** - закрывающая кавычка.

. **no-open-quote** - отменяет открывающую кавычку.

. **no-close-quote** - отменяет закрывающую кавычку.

. **inherit** - наследует значение элемента родителя.

. **none** - ничего не добавляется.

. **normal** - для псевдоэлементов **before** и **after** тоже самое, что и **none**.

. **counter** - показывает значение счетчика, заданного свойством **counter-reset**.

. **attr(атрибут тега)** - показывает текст, который является значением атрибута того или иного тега указанного в скобках.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Псевдоэлементы after и
before</title>
    <style type="text/css">
      *:before {color: #00f; font-variant:
small-caps;}
      *:after {color: #f00; font-variant:
small-caps;}
      body:before {content: "начало
документа";}
      body:after {content: "конец
документа";}
      p:before {content: "анекдот:";}
      p:after {content: ":-)"; font-size: x-
large;}
      ul {list-style-type: none;}
      li:before {content: "№ "; color:
#0f0;}

h4:before{content:url(graphics/marker
```

```
.gif);}
        q:before {content: open-quote;
font-size: 30px;}
        q:after {content: close-quote;
font-size: 30px;}
        img:after {content:attr(src);}
      </style>
    </head>
    <body>
      <hr>
      <h4>Заголовок с сердечком.</h4>
      <h4>Параграфы с добавлением
слова "анекдот:" вначале и смайлика в
конце:</h4>
      <p>Если бы при приеме на
работу...</p>
      <p>Однажды молодой
сисадмин...</p>
      <h4>После рисунка добавляется
значение его атрибута: "src" - путь к
рисунку:</h4>
      <img src="graphics/fon.jpg">
      <h4>Цитата в кавычках:</h4>
      <q>Лёд тронулся, господа
присяжные заседатели!</q>
      <h4>Список с нестандартными
```

```
маркерами:</h4>
      <ul>
        <li> Первый
        <li> Второй
        <li> Третий
      </ul>
      <hr>
   </body>
</html>
```

Чувствую перехитрил я с примером, поэтому несколько комментариев к нему:

Звездочка перед псевдоэлементом ***:псевдоэлемент{свойство}** говорит о том, что указанные правила стиля распространяются на все элементы. Так в нашем примере текст в начале везде синий, а в конце везде красный, если он конечно не указывается дополнительно, как например в случае с зелёными "маркерами" в списке из примера.

В качестве добавляемого контента может вступать какой либо объект, в

примере мы добавили рисунок ко всем заголовкам, однако если браузер не сможет обработать тот или иной файл, то ничего не добавится.

Спецсимволы HTML (например, спецсимвол: **¶**), будут отображаться простым текстом (**¶** а не **¶**) так что если необходимо добавить какую либо хитрую закорючку используйте юникод.

Псевдоэлементы **after** и **before**, как и свойство **content** не поддерживаются браузерами Internet Explorer 7 и ниже.

Псевдоэлемент ::selection.

Псевдоэлемент **::selection** (я не опечатался, пишется именно с двумя двоеточиями) указывает на стиль выделенного пользователем текста.

Данный псевдоэлемент появился на свет только в спецификации CSS3 и к сожалению поддерживается не всеми

браузерами так IE его полностью игнорирует, а браузер Firefox использует свой аналогичный псевдоэлемент **::-moz-selection** который официально не входит в спецификацию CSS.

Пример:

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Псевдоэлемент
::selection</title>
    <style type="text/css">
      p::selection {
        color: #f00;
        background: #0f0;
      }
    </style>
  </head>
  <body>
    <p>Попробуете выделить данный
текст, как будто Вы собираетесь его
скопировать. Если Ваш браузер
```

```
(например Opera 9.6 и выше)
поддерживает псевдоэлемент
::selection, Вы увидите, что
выделенный текст станет красным, а
его фон зелёным.</p>
  </body>
</html>
```

К данному псевдоэлементу можно применить только следующие CSS свойства: **color**, **background** и **background-color**.

Полезные советы:

. В спецификации CSS3 в отличии от CSS2 и CSS2.1 все псевдоэлементы принято писать с двумя двоеточиями **::first-letter, ::first-line, ::after, ::before,** и новый **::selection** - таким вот способом разработчики решили отделить пседоэлементы от псевдоклассов. Однако такой синтаксис напрочь игнорирует Internet Explorer до 9 версии включительно! Так что пока псевдоэлементы лучше писать по старинке с одним

двоеточием. Однако следует понимать, что например **:first-letter** и **::first-letter** это формально два разных псевдоэлемента.

. В правилах стиля для псевдоэлементов допустимо использовать только свойства, относящиеся к шрифту, тексту и его фону.