Игорь Квентор

Блочная верстка веб-сайта

www.websovet.com

Содержание

Содержание 2
От автора 3
Введение 4
Что такое блок? 5
Тэги 7
Тэг div 8
Атрибуты тэга 8
Атрибуты id и class 9
Тэг span 10
Создаем свой первый сайт | Инструментарий 10
Создаем свой первый сайт | Начало 11
CSS или Каскадный Лист Стилей 14
Создание файла стилей CSS 15
О фоновых картинках 20
Шапка сайта 21
Навигация или меню сайта 24
Контент 26
Нумерованный список 29
Боковая колонка или сайдбар 31
Подвал сайта 33
Полезное 35

От автора

Здравствуйте! Меня зовут Игорь Квентор. Я автор блога Websovet.Com, который посвящен вебмастерингу в общем и блочной верстке веб-сайтов в частности.

Книга "Блочная верстка веб сайтов" предельно доступно и просто, в нескольких уроках рассказывает о таких сложных для новичка вещах, как HTML и CSS.

Не секрет, что для овладения навыками веб строительства, необходимо изучить язык разметки HTML. При одном только взгляде на мудреный код у новичка разбегаются глаза, и в желудке становится нехорошо. И это понятно. Новое и неизвестное доселе всегда выглядит страшным и жутко сложным.

Чего бы там не рекламировали создатели визуальных редакторов веб-сайтов типа Dreamweaver, все равно рано или поздно пытливый ум начинающего вебмастера засунет нюх во всю скрытую от посторонних глаз кухню страничного кода и захочет разобраться что тут и как. Как, впрочем, и любой фанат «визуальщины» нет-нет, да и признает, что хорошо бы все-таки изучить HTML, ибо без него как без рук.

И тогда человек идет в лавку и покупает книгу на 800 страниц в 2 кг весом, а затем, собравшись с духом, да на волне первоначального энтузиазма, погружается в мудрое чтиво. Но на сколько его хватит? Правильно, на введение и половину первой главы. Потому что дальше на адепта нападет неистребимая зевота и скука. Еще бы! Хочется прямо сразу начать действовать, а тут… Хорошо, если сразу разъясняется код, виды тэгов и общие принципы построения веб-страниц. А если пятьдесят страниц только описания истории появления HTML? Да от этого у кого хочешь отпадет всякое желание заниматься дадьше.

Поэтому в моей книге вы не найдете глубинных смыслов и жутких теоретических джунглей, а сразу начнете верстать Свой Первый Сайт. Причем не абы как, а с соблюдением всех современных норм и правил. И на простом, в общем-то, примере вы легко постигнете азы веб строительства, а код, который казался страшно непонятным и запутанным, предстанет перед вами во всей красе как вполне обычная и понятная азбука.

Книга распространяется совершенно бесплатно. Вы можете поделиться ей со своими друзьями и знакомыми, кто также интересуется блочной версткой веб-сайтов. Вы также можете отправить им ссылку на страницу, где эту книгу можно скачать, либо поделиться ссылкой в ваших профилях в социальных сетях.

Enjoy!

Введение

Что есть такое **блочная верстка** и с чем ее едят? Верстка сайта (не путать с версткой газет и журналов) — это запись в виде кода всего содержимого сайта. Код виден только браузерам и не виден посетителям сайта. Самый простой способ увидеть код сайта — нажать сочетание клавиш Ctrl+U при просмотре в браузере. В новом окне откроется малопонятная непосвященному тарабарщина. Это и есть код, который нам предстоит изучить. Или иначе — верстка.

Ранее сайты верстали при помощи таблиц. Каждый элемент страницы, будь то заголовок, текст или картинка, располагался в собственной ячейке. Ячейки эти кучно роились в других, более крупных ячейках, а те в свою очередь лежали в главной ячейке, сиречь самой странице сайта.

Табличная верстка сейчас уже морально устарела, хотя очень многие вебмастера продолжают ее использовать. Большим минусом ее является тяжеловесный код. Ведь каждую маломальскую ячейку нужно как-то обозначить, задать ей координаты и размеры. Кроме того, каждая ячейка, как своеобразная коробочка, имеет стенки, у которых есть толщина и цвет. И эти данные тоже нужно закодировать.

В результате помимо полезного наполнения страницы, мы получаем кучу сопутствующего содержимого в виде служебной информации. При существующих ныне скоростях в интернете, оно вроде и не страшно. Вряд ли вы сумеете на глаз заметить разницу в скорости загрузки страницы, сверстанной таблицами от страницы сверстанной блоками. Более того, табличные данные рекомендуется верстать таки таблицами, а не лепить гору обтекаемых блоков.

Блочная верстка позволяет обойтись без жесткой конструкции табличной сетки. Принцип тут простой: каждый элемент страницы, будь то функциональная часть (шапка, например), кусок текста (абзац), список, картинка и пр. называется блоком. Блоки, как кирпичики, располагаются на странице в режиме "естественного потока". Это как будто вы пишете письмо, и как только строка кончается, то переходите на следующую. Монитор — это тоже своеобразная "страница". Содержимое сайта на мониторе так же "раскладывается" слева направо и сверху вниз по порядку, заполняя по-умолчанию все свободное пространство.

Я очень люблю приводить в качестве наглядного примера пустую картонную коробку. Представьте себе, что у вас есть коробка из-под обуви. И несколько предметов прямоугольной формы, которые вам нужно плотно уложить в эту коробку в определенном порядке. Каждый предмет — это блок или кирпич, если угодно. Чтобы не перепутать, каждый из них помечен своим именем. Некоторые уникальные, единственные в своем роде. Другие похожи как две капли воды. Некоторые занимают достаточно много места, встают плотно от одной до другой стенки коробки. Другие мелкие, их можно уложить в ряд, по нескольку штук сразу.

Блоки — это любое полезное содержимое страницы сайта. Картинки, абзацы текста, видео-ролики, баннеры и пр. Они в свою очередь входят в состав более крупных блоков: шапка сайта, центральная колонка, боковая колонка(и), подвал. Наша задача — как можно более аккуратно разместить все эти блоки на странице (в коробке), чтобы ничего не торчало, не терялось и не наслаивалось друг на друга

У блоков есть одна интересная особенность: каждый из них считает, что недостойно находиться на одной горизонтали (читай — строке) с другим блоком. Например, ширина коробки 1 метр, ширина двух одинаковых блоков по 50 см. Казалось бы, они оба должны уместиться в один ряд. Но по умолчанию они все равно встанут один под другим, причем второй будет внизу.

Это логично, если исходить из того, что верстка изначально придумана для текста. Ведь текст — это абзацы. Они и должны быть один под другим, да еще и с заметным отступом. Иначе читать будет трудно.

Это умолчание можно откорректировать и расположить несколько блоков на одной линии. Но об этом позже.

Тут у сметливого читателя может возникнуть вопрос: а чем, собственно, тогда отличается блочная верстка от табличной? Ведь все очень похоже: мы распихиваем мелкие блоки (мелкие ячейки) в более крупные блоки (крупные ячейки), а их в одной большой коробке (главная ячейка таблицы).

Отвечаю: по-сути ничем! Но за одним важным исключением: нам не нужно для каждой ячейки прописывать ее ячейковые атрибуты, без которых никак, а также нам без разницы, какая толщина стенок у этой ячейки и уж тем более ее цвет. Ибо у блоков стенок просто нет (рамка есть, но это другое)! Вот и все отличие. Но сколько места, времени и сил сразу экономится!

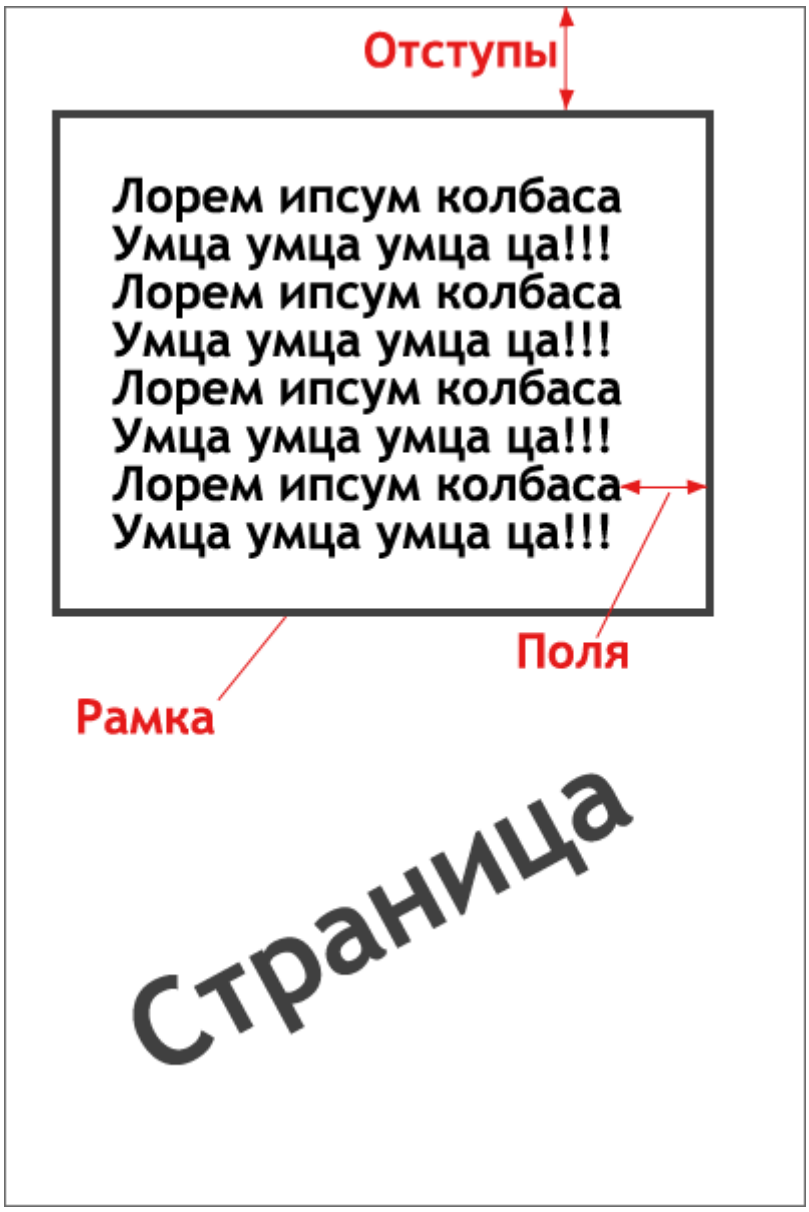
В блочной верстке принцип потока — основополагающий. Это как длинный паровоз из блоков, который словно змея зигзагообразно плотно укладывается в каркасе (коробке) страницы. Каркас сам по себе держит всю конструкцию. Блоки плотно прилегают друг к другу. А те, что не уместились в одном ряду, автоматически заползают ниже, в последующие ряды. Поэтому табличная сетка как скелет здесь попросту не нужна.

Ну и для сравнения: когда я сверстал свою самую первую веб-страницу, используя таблицы, а потом переделал в блочный вариант, то вес кода уменьшился ровно в 4 раза.

Что такое блок?

Блок — это обычная прямоугольная область, обладающая рядом свойств, таких как: рамка, поля и отступы. Содержимым блока, как я уже и говорил, может быть что угодно: кусок текста, картинка, видео-ролик, список, форма для заполнения, меню навигации и т.п.

Для примера рассмотрим абзац текста, размещенный на странице:



Рамка (border) — это контур, для которого можно задать такие характеристики как толщина, цвет и тип (пунктирная, сплошная, точечная). В отличие от таблиц вещь необязательная.

Поля (padding) — отделяют содержимое блока от его рамки, чтобы текст, например, не лепился тесно к стенкам блока.

Отступы (margin) — это пустое пространство между различными блоками, либо между блоком и стенками страницы сайта. Они позволяют расположить блоки на заданном расстоянии относительно друг друга.

Здесь нужно дать пояснение в плане русскоязычных названий. Дело в том, что слово margin (а не padding) в переводе с английского означает поле. И многие ревнители езыга пытаются указать мне на ошибку: мол margin — это поля, а padding — набивка. Не спорю. Если тупо и дословно переводить, то так оно и есть. Но при таком раскладе теряется весь смысл и понимание сути вещей. Отступ — гораздо понятнее означает некоторое расстояние от одного забора до другого. А поле звучит понятнее какой-то там набивки. Это свободное пространство от грядки с картошкой до забора. Так что не надо песен, как говорится!

Тэги

Тэг — это ключевое понятие и особая конструкция языка HTML. Не путать с метками (тэгами), которые проставляются в блогах и жежешечках как ключевые слова! Можно сказать что тэг — это буква алфавита HTML. Служит он в первую очередь для конкретизации объекта, а во вторую — его начала и конца.

В самом простом случае тэг — это как деталь детского конструктора, которая имеет своё строгое предназначение: планка — значит планка, колесо — значит колесо и ничто иное. К примеру, тэг абзаца:

```
<p>Лорем ипсум колбаса. Умца умца умца ца!</p>
```

всегда обозначается буквой `p` и никак иначе. Любой браузер, когда видит такую конструкцию, сразу понимает, что это абзац и показывает его в соответствии с предписанными правилами, о которых поговорим чуть позже.

Тэги всегда заключены в угловые скобки. Различают *открывающий* и *закрывающий* тэги. В примере выше мы видим как открывающий, так и закрывающий тэг. У последнего перед буквой добавлен слэш — косая черта, наклоненная вправо /. Данный значок расположен на клавиатуре во втором ряду снизу, в английской раскладке (там, где русская точка). Не путайте его со значком, расположенным во втором ряду сверху в русской раскладке. Это совершенно разные знаки, хотя и похожи.

Форма `<p></p>` характерна для большинства тэгов. Она задает характер и границы объекта. Смотрим далее:

```
<p></p><p></p>
```

Это уже два абзаца, записанные подряд. Вы видите, что они расположены в строгом порядке: сначала первый тег "открывает" абзац, следом тэг со слэшем "закрывает" абзац. Далее процесс повторяется. Нельзя нарушать этот порядок. Следующая запись является ошибочной:

```
<p><p></p>
```

То есть, тэги не должны пересекаться или теряться! Они могут либо идти последовательно, поочередно открываясь и закрываясь, либо быть вложенными один в другой. Но об этом позже.

Не все тэги имеют закрывающую пару. Например, тэг изображения `img` его не имеет вовсе. Но чтобы соответствовать современным стандартам и требованиям спецификации XHTML, ему все-таки добавляют перед закрывающей угловой скобкой пробел со слэшем. Выглядит это примерно так:

```
<img />
```

Тэг div

Ох уж этот див! Притча во языцех и зубная боль верстальщиков старой (табличной) школы. В отличии от строгих привязок стандартных HTML-тэгов к своему содержимому (`p` — к абзацам, `a` — к ссылкам, `img` — к изображениям), тэг `div` является по-сути нейтральным. То есть ему всё равно, что содержать, хоть всё разом.

Обычно тэг `div` используют для задания больших функциональных областей на странице, таких как: шапка (header), блок навигации (меню), блок(и) основного содержимого, футер (footer) или подвал по-нашему и т.п. Но может он применяться и к более мелким деталям сайта. И он также является парным: `<div></div>`

То есть, див — это такая коробочка, поменьше чем "главная коробка" и побольше отдельно взятого элемента вроде абзаца текста или картинки.

Если рассматривать див с позиции все той же табличной верстки, то это будет что-то вроде крупной ячейки, в которой размещаются более мелкие. Понятно, что к такой ячейке можно применять ряд правил, которому будут подчиняться все вложенные в нее элементы.

Более подробно со всем этим хозяйством разберемся далее по ходу сборки нашего первого сайта

Атрибуты тэга

Атрибут — это описательная характеристика тэга. Что он может делать и каким образом. Например, опять же возьмём тэг изображения `img` и покажем его код целиком, как он обычно выглядит в коде страницы:

```
<img src="images/risunok.jpg" width="200px" height="50px" alt="Рисунок меня, где я кормлю собаку, а она кусает меня зачем-то" />
```

В данном случае **src, width, height, alt** являются атрибутами тэга. В кавычках даны значения атрибутов.

Расшифровать такую запись несложно. Тэг указывает, что в данном месте страницы нужно прицепить изображение `img` с именем `risunok.jpg` из папки `images`, шириной 200 пикселей и высотой 50 пикселей.

Обратите внимание на запись адреса рисунка. Он пишется через слэш: `images/risunok.jpg` Это типичный способ адресации в HTML. О нем мы поговорим чуть позже, когда станем верстать страницу сайта.

В коде добавлен альтернативный текст, обозначенный атрибутом `alt`. Этот текст всплывает на страничке при наведении мышки на рисунок. Вещь весьма необходимая и полезная. Не все пользователи в сети обладают хорошим зрением. Кто-то пользуется программой распознавания и чтения текста. А кто-то просто выключает показ картинок в браузере. Вот для них и существуют альтернативные подписи к рисункам. Если же их нет смысла подписывать (например, декоративные элементы дизайна), то у атрибута alt в кавычках не пишется ничего — `alt=""`. Но сам атрибут должен присутствовать.

Атрибуты id и class

Чтобы как-то различать однотипные тэги им были придуманы специальные атрибуты. Наиболее часто используют два вида: идентификаторы (id) и классы (class).

id — атрибут, позволяющий придать тегу уникальный набор свойств, то есть такой, который на странице сайта используется только один раз. Например, header или footer. Ясное дело, что такие элементы на странице уникальны. Не может быть у сайта 2 шапки или 2 подвала. Верно? Иначе это будет бардак, а не сайт. То есть, уникальный идентификатор id позволяет, во-первых, не заблудиться в коде, а во-вторых, задавать элементам страницы уникальные же свойства.

class — атрибут, который позволяет один и тот же набор свойств задать нескольким элементам на странице сайта. Например, если в тексте статьи у нас использованы 3 картинки и двум из них нужно добавить цветную рамку, а третью оставить без изменений. Так как изображений несколько, то атрибут id уже нельзя использовать.

Данные атрибуты могут применяться не только к большим функциональным блокам, обозначенным тэгами div, но и непосредственно к тэгам элементов: абзацам, картинкам, ссылкам и т.д.

Здесь нужно прояснить одну важную, но не совсем очевидную вещь. Можно, конечно же, все элементы страницы сайта обозначить тэгами div, сложив любую мало-мальскую деталь в отдельную коробочку. И многие неумелые верстальщики так и поступают. Их код просто пестрит "дивами". Или вот такими конструкциями:

```
<div class="text"><p>Лорем ипсум колбаса. Умца умца умца ца!</p></div>
```

Что называется — кашу маслом не испортишь. Однако здесь это явно излишне. Смотрите:

```
<p class="text">Лорем ипсум колбаса. Умца умца умца ца!</p>
```

Класс под именем `text` мы добавили непосредственно тэгу абзаца. Зачем еще окружать его тэгами div? Совершенно излишне.

То есть, нужно отчетливо понимать, для чего используются дивы: для задания крупных функциональных областей, а не для каждой мелочи.

Идем далее.

Имя `"text"` называется также **значением атрибута**. Именно к значению (или имени, если угодно) применяются правила отображения данного элемента. Правила эти записываются в листе стилей CSS, до которого мы еще доберемся. А пока запомните простую вещь:

Имя (значение) атрибута выбирается произвольно!

Что это значит? Вы думаете, слова header, footer или тот же text — это строго установленные имена элементов? Ничего подобного! Имя — вещь сугубо личная и может придумываться верстальщиком прямо на ходу. Например, нет никакой разницы в именах для шапки сайта: header или megaheader. Улавливаете? Вы сами придумываете имя для атрибута, какое вам удобнее.

Тэг span

Тэг **span**, так же как и **div**, является нейтральным. Он может применяться к любому элементу или группе элементов на странице сайта. Главное отличие его от дива — характер размещения на странице. Если div — это чисто блоковый тэг, который по-умолчанию не терпит соседства с другим блоком, то span — это *строковый* тэг. То есть, на одной строке может размещаться подряд несколько "спанов", тогда как "дивы" стремятся залезть один под другой. О чем мы ранее уже говорили.

Поясню на примере. Вот есть у нас абзац текста, который набран обычным черным шрифтом. В этом тексте нам нужно одно слово выделить красным цветом, другое зеленым. И мы пишем вот такую конструкцию:

```
<p class="text">Лорем ипсум <span class="redcolor">колбаса</span>.
<span class="greencolor">Умца умца умца ца!</span></p>
```

То есть, у нас имеется абзац текста, для которого прописан один общий класс с именем `"text"`, а внутри размещены два нейтральных тэга `span` с именами `redcolor` и `greencolor`. В листе стилей CSS мы для каждого имени пропишем соответствующий набор правил. Пока их касаться не будем. Заметьте только, что оба "спана" идут один за другим и они как бы вложены внутрь тэга `p` с классом `"text"`. Они строковые. Располагаются в одну линию.

Чтобы стало понятнее просто запомните: по-умолчанию div-ы на странице сайта всегда располагаются автоматически один под другим, тогда как span-ы вытянуты в строку и переносятся на следующую только когда закончится место справа.

Создаем свой первый сайт | Инструментарий

Окей! Отдохнем чуток от теории и соберем небольшую кучку необходимых для верстки сайта инструментов. Нам понадобится один правильный текстовый редактор для набора кода и несколько браузеров для проверки сайта. Плюс Фотошоп для подготовки картинок.

Заходим вот на этот симпатичный сайт www.pspad.com и скачиваем замечательный редактор текста и программного кода по имени **PSPad**. Замечателен он, прежде всего тем, что придумали его программисты для программистов же. Ну и верстальщиков тоже не забыли. А потому в отличие, скажем, от банального Блокнота, он не хулиганит и не портит код всякими невидимыми вставками. Кроме того, достаточно удобен и понятен, ибо по-русски все.

Еще нам, конечно же, понадобятся браузеры. Причем разные. А все потому, что каждый браузер считает себя круче других и иногда показывает сайт исключительно по-своему. И наш красивый во всех отношениях сайт у одного браузера может вылезти чуть вбок, у другого раздаться вширь, а у третьего вообще нарисуется шапка вместо валенок и наоборот.

Поэтому нам понадобятся как минимум четыре самых распространенных на данный момент браузера: Interner Explorer, Opera, Google Chrome и Mozilla FireFox.

Создаем свой первый сайт | Начало

А создадим мы вот такой симпатишный сайт о Летающих Парасенгах.

И начнем сразу же с верстки главной страницы сайта. В текстовом редакторе PSPad создадим новый документ: Файл – Новый – HTML. Только прежде убедитесь, что во вкладке Формат у вас включен пункт ANSI. Иначе вместо русского текста получим кракозябры.

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

```
  <meta http-equiv="content-type" content="text/html; charset=windows-
1250">
  <meta name="generator" content="PSPad editor, www.pspad.com">
  <title></title>
  </head>
  <body>

  </body>
</html>
```

Как видите, редактор автоматически создал самую простую структуру готовой HTML странички. Разберем, что тут есть.

DOCTYPE

Для начала определим тип нашего документа. Любая грамотно свёрстанная страница должна в самом начале содержать так называемый *доктайп* (DOCTYPE). Нужен он не для форсу бандитского, а для всевозможных устройств вывода информации, в том числе и браузеров. По-умолчанию редактор выдал не самый строгий на сегодняшний день доктайп `HTML 4.01 Transitional`

Обратите внимание: доктайп заключен в угловые скобки, как какой-нибудь тэг. По-сути он им и является, однако не имеет закрывающего тэга со слэшем. Подробно на видах доктайпов останавливаться не будем. Кому интересно — вэлкам изучать W3C

HTML

Сразу за доктайпом появляется самый первый и самый важный тэг `<html>`. А в самом низу его "закрывашка" `</html>`. Таким образом, мы обозначили одновременно и вид документа (html страница), и его начало и конец.

HEAD

Следом за `<html>` идет тэг "головы" страницы `<head>`. Не путайте его с шапкой сайта, так называемым хедером! Это суть разные вещи. `<head></head>` содержат в себе исключительно служебную информацию, которую простой посетитель сайта не видит. Она нужна в первую очередь для самого браузера, а во вторую для всевозможных поисковиков типа гугла или яндекса.

META

Служебная информация записана в специальных мета-тэгах `meta`, которые также не отображаются на странице в браузере. Их может быть достаточно много. Мета-тэги показывают, в какой кодировке создана страница сайта, какие у нее есть ключевые слова, описательные текст, название (тайтл) и пр.

В нашем случае в голове страницы размещены два метатэга. Первый показывает кодировку, которую мы сразу же исправим с 1250 на 1251, ибо это более правильно. Второй — это приколка самого редактора, который говорит, что это он наваял страницу, а вовсе не мы. Нам этот метатэг не нужен. Удаляем смело.

TITLE

Ну и, наконец, тайтл **<title></title>** надо обозначить по-человечески. Ибо именно он будет красоваться в самой верхушке браузера на синей полосе и, если повезет, то когда-нибудь появится в результатах поиска гугла или яндекса :)

Кроме того, мы добавим в голову еще парочку полезных метатэгов, а именно ключевые слова и описательный текст. И в самом конце головы прицепим ссылку на файл стилей.

Смотрим, что получилось:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
 <head>
  <meta http-equiv="content-type" content="text/html; charset=windows-
1251">
  <meta name="description" content="Сайт о поросенках-летунах и
счастливой летучей жизни." />
<meta name="keywords" content="летать, свин-летун, пиггсы, полёты,
лёччики, пилоты." />
<title>Сайт Летающих Парасенгов | Главная</title>
<link rel="stylesheet" href="style.css" type="text/css" />
  </head>
  <body>

  </body>
</html>
```

Первый мета-тэг показывает на кодировку сайта. В рунете желательно пользовать все-таки виндовз-1251, чтобы случайно у Васи Пяткина в его IE версии 3.0 не повылезали вместо красивого и грамотного текста всякие кракозябры.

Второй мета-тэг — это краткое описание сайта. Именно эта строка первой покажется в результатах поиска яндекса или рамблера, если они ещё нас найдут.

В третьем мета-тэге ключевые слова для поисковиков. Объяснять не буду зачем это нужно, думаю и так ясно.

Про тайтл уже было сказано.

И наконец последняя строка — это не что иное как ссылка на наш лист стилей CSS, который мы создадим чуть позже. Про него скажу лишь вот что: по умолчанию наш лист стилей могут использовать любые устройства вывода инфы будь то экран монитора, принтер, либо телетайп какой-нить шпиёнский. Но если мы хотим указать для какого устройства конкретно предназначена страница, то после атрибута **type** со значением **text/css** должны будем указать дополнительный атрибут **media** со значениями:

print — для принтера
handheld — для “наладонника”
screen — для монитора only и т.д.

Это понятно — если кто-то захочет распечатать нашу страничку, то ему вовсе ни к чему на отпечатке навигация, кнопки-картинки, рекламные баннеры и супер-красивые заголовки. Только полезная инфа. И вот для такого случая пишется отдельный лист стилей попроще, специально для печати.

А теперь впишем закрывающий тэг **</head>**. Все, со служебной инфой покончено.

Обратите внимание — тэги как бы вложены друг в друга:

<head><title></title></head>

Это правило вложения должно выполняться всегда! Никаких перестановок типа:

<head><title></head></title> — ЭТО НЕПРАВИЛЬНО!

BODY

Раз уж у страницы есть голова, то по всей вероятности должно быть и тело. Ручки-ножки не в счет. За тело отвечает тэг **<body>**. Он идет сразу же следом за **<head></head>** и является именно той частью веб страницы, которая видна посетителю сайта. То есть, здесь размещается полезное содержимое — контент.

Между **<body>** и **</body>** может размещаться сколько угодно инфы. Тело — оно и есть тело. Пока на этом остановимся и сохраним нашу страничку с именем index.html. Почему именно index? А потому, что любой сервер станет сразу же искать у себя в заначке страницу с этим именем, когда пользователь наберет в своем браузере адрес вида http://www.moykrutoysite.ru/. Устроены они так. Индексная страница для них всегда является главной, то есть — стартовой.

Теперь сохраним файл нашей страницы с именем index.html в какую-нибудь папку. Скажем, pigfly. Это будет наш проект. На хостинге, куда мы отправим готовый сайт, такая папка будет называться *корневой*. В ней мы также поместим файл стилей и еще одну папку с картинками.

CSS или Каскадный Лист Стилей

Итак, **CSS** или иначе *Каскадный Лист Стилей*. Почему каскадный объясню позднее. Пока лишь в общих чертах обозначим, что это за зверь такой.

Стиль — это внешний вид того или иного элемента страницы. Скажем, по-умолчанию все ссылки браузеры отображают синим цветом, а посещенные — фиолетовым. Это не есть красиво. Чтобы задать свои собственные цвета, нам необходимо где-то это записать. Чтобы браузер прочел и выполнил нашу такую хотелку. Хотелки наши как раз и прописываются в файле (или иначе — листе) стилей.

Сами стили могут быть в трех различных ипостасях: местные, встроенные или отдельным файлом. Правильнее всего — последний вариант. И это логично: зачем лепить на странице сайта рядом с полезным содержимым и кучу описаний, как оно (содержимое) должно выглядеть? И пусть описание не видно посетителю (браузеры его не отображают как текст). Но ведь каша получается. Да и страница тяжелеет весьма и весьма. Поэтому и было решено вынести стили в отдельный файл, ссылку на который мы прописали ранее в голове **<head></head>**.

Смысл тут не только в облегчении страницы. Основная идея — разделение полезного содержимого и его оформления. Это особенно оправдано при наличии большого

количества страниц на сайте. Представьте себе: захотелось вам поменять цвет шрифта на всем сайте. Это ж сколько страниц пришлось бы перелопатить, если бы стиль был прописан в каждой из них. А так вы исправили значение лишь в одном файле стилей, и это правило автоматически будет применено ко всем страницам сайта.

Тут стоит сразу же упомянуть о двух других вариантах: *местном стиле* и *встроенном стиле*.

Местный стиль — это когда непосредственно у какого-либо элемента страницы прописывается стиль. Выглядит это таким образом:

```
<p style="color: red;">Лорем ипсум колбаса. Умца умца умца ца!</p>
```

Весь этот абзац будет набран красным цветом. Местный стиль удобно применять, когда нужно изменить вид всего одной-двух мелких деталей, чтобы из-за них не добавлять лишних правил в отдельный файл стилей. И это тоже правильно: зачем загромождать файл стилей, если мы этот дополнительный стиль больше никогда не будем использовать в дальнейшем.

Встроенный стиль — это практически тот же файл стилей, но записанный не отдельным файлом css, а внедренный непосредственно в код страницы сайта в пределах тэгов `<head></head>`. Выглядит это таким образом:

```
<style type="text/css">
…тут всякие правила…
</style>
```

Зачем делать такую штуку, если можно вынести стили в отдельный файл? Иногда как раз этого не стоит делать. Например, если у вас мини-сайт или сайт-заставка, который состоит из пары-тройки страниц, а то и вовсе из одной главной. Согласитесь, что нет смысла выносить стили в отдельный файл в таком случае. Тем более, если и самих-то стилей всего ничего.

Создание файла стилей CSS

Открываем новый документ в текстовом редакторе PSPad, но на этот раз выбираем другой тип документа: Файл – Новый – Cascading Style Sheet. Там уже будет прописана одна строка:

```
/* CSS Document */
```

Это так называемый *комментарий*. Нужен он больше для самого верстальщика, нежели для сайта. Комментариями часто помечают в файле стилей разделы документа. Например, часть правил для шапки сайта, часть для контента и т.д. То есть, для удобства чтения кода, чтобы не заблудиться. Комментарием может быть любой текст. Браузеры не читают и нигде не отображают комментарии.

Чтобы добавить комментарий, достаточно поместить его между двумя слэшами со звездочками /* */.

Ну а теперь приступим непосредственно к записи кода в листе (файле) стилей.

В листе стилей CSS части кода называют *правилами*. Каждое правило состоит из *селектора* (читай - атрибута) и *объявления*. Объявление, в свою очередь, состоит из *свойства* и *значения*.

Чтобы стало понятнее, рассмотрим пример:

```
p {color: #000;}
```

Данная запись означает, что все абзацы будут набраны чёрным шрифтом. Здесь `p` — это атрибут, а то, что находится в фигурных скобках и есть *объявление правила* для этого атрибута. Слово `color` является *свойством объявления*, а решётка с тремя нулями — *значением объявления*.

В данном случае значение объявления записано в виде шестнадцатиричного числа, обозначающего цвет. Всем, кто пользуется Фотошопом это должно быть известно. Возникает вопрос: почему всего три нуля, ведь в данном обозначении должно быть шесть знаков? Все просто. Когда пары знаков одинаковы 00 00 00, FF FF FF, то допускается писать сокращённо — 000, FFF и т.п. Все браузеры это понимают правильно.

Правило можно писать как угодно — хоть в строку, как у нас, хоть в столбик — это роли не играет:

```
p {
color: #000;
}
```

Важно только не забывать две вещи:

1. После каждого свойства необходимо ставить двоеточие.
2. После каждого значения — точку с запятой.

Окей! С этим разобрались. Теперь давайте сразу запишем ряд правил, а после разберем, кто есть ху:

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


body {
padding: 2% 0 0;
background: #fff;
color: #333;
font-family: "Comic Sans MS", Verdana, Arial, Helvetica, sans-serif;
}

#container {
width: 760px;
margin: 0 auto;
border : 1px solid #999;
}
```

В первом правиле звёздочка означает не что иное, как всю страницу разом. Это так называемый *общий сброс*. Он нужен для того, чтобы некоторым из браузеров не пришло в голову добавлять рамки, отступы или поля где ни попадя. Особенно этим грешит IE (Internet Explorer). Это не обязательный набор правил. Но правильные верстальщики пишут его везде и не парятся.

Сама звёздочка — это не тэг и нигде потом в коде страницы не указывается. Браузеры прекрасно понимают её значение и применяют данные с ней правила ко всей странице. Значения указываются либо в процентах, либо в пикселах. Кстати, если стоит ноль, то единицу измерения не нужно указывать. Ноль — он и в Африке ноль.

Следующий набор правил — для тела страницы body. Первое из них задает поля: сверху 2%, с боков по нулям, снизу тоже ноль. Это значит, что наша страничка не будет лепиться к верхушке окна браузера, а отступит от него на 2% размера окна. Тут значения идут подряд без запятых и только после последнего ставится точка с запятой.

Вы можете спросить: почему всего три значения, когда у прямоугольника, коим по-сути и является страница сайта, ровно 4 стороны? Не больше и не меньше. Все просто. Сейчас расскажу о принципе, по которому в верстке задаются любые значения для сторон прямоугольника.

Напомню: у любого прямоугольника есть 4 стороны. Кто не верит — бегом читать учебник геометрии. И значения для них задаются по часовой стрелке, начиная сверху, затем правое, низ и, наконец, левое. Таким образом, если бы нам нужно было задать поля так, чтобы сверху и снизу было по 10 пикселей, а по бокам по 20, то мы бы написали так:

```
padding: 10px 20px 10px 20px;
```

Уловили? По часовой стрелке, начиная сверху: 10, 20, 10 и снова 20. Это просто. Идем далее.

Фишка в том, что существует так называемая *сокращенная запись* правил. Сокращения могут быть различными. Одно из них — это сокращенная запись значений, если они одинаковы для нескольких сторон. В нашем случае у верха и низа поля по 10 пикселей, а по бокам по 20. Поэтому запись правила можно сократить до:

```
padding: 10px 20px;
```

И это будет ровно тоже самое!

Другой вариант: сверху поле в 10 пикселей, по бокам по 20, а снизу всего 5. Как быть? А вот так:

```
padding: 10px 20px 5px;
```

Ну и, наконец, если со всех сторон одинаковые поля, то и значение в правиле дается всего лишь одно:

```
padding: 10px;
```

Надеюсь, идея понятна.

Следующее правило задает общий фон всему сайту:

```
background: #fff;
```

Как видите, цвет записан в три знака. На самом деле шестнадцатиричный код белого цвета #ffffff. Но, как я уже и говорил, если пары знаков одинаковы, то можно сократить запись до трех значков. Цвет общего фона лучше прописывать всегда и не надеяться на цвет по-умолчанию, который у всех браузеров и так белый. Вдруг какой умник в своем браузере изменит цвета по-умолчанию на что-то другое?

Следующее правило задает цвет шрифта для всего сайта:

```
color: #333;
```

В нашем случае это темно-серый #333. Это не значит, что теперь все буквы на сайте станут темно-серыми. Ссылки так и останутся синими по-умолчанию, а посещенные, соответственно, фиолетовыми. Данная запись лишь устанавливает общий цвет шрифта, который в конкретных ситуациях (например, для заголовков) можно будет в дальнейшем изменить. Но об этом чуть позже.

Следующее правило устанавливает семейство шрифтов, которые мы используем:

```
font-family: "Comic Sans MS", Verdana, Arial, Helvetica, sans-serif;
```

Зачем такая длинная запись, если нам нужен конкретно шрифт Comic Sans MS для основного шрифта? Дело в том, что иногда на компьютере пользователя нет такого шрифта. Но вполне вероятно, что у него есть следующие из перечисленных. Вот поэтому мы и предлагаем браузеру пользователя в порядке предпочтения выбрать какой-либо из них. Последнее слово sans-serif означает общее семейство шрифтов без засечек. Соответственно, serif — это с засечками.

Тут есть одно важное замечание: если имя шрифта состоит более чем из одного слова, то его нужно взять целиком в кавычки. Например: "Times New Roman".

Идем далее. Следующий набор правил очень любопытный. Во-первых, появилось незнакомое слово container с решёткой (#) впереди. Данная решётка и означает уникальность атрибута. То есть тэг div с данным атрибутом будет использован только один раз на странице. Помните, мы говорили про id? Вот это оно и есть.

```
#container {
width: 760px;
border : #999 solid 1px;
margin: 0 auto;
}
```

Что это за контейнер такой и для чего он нужен? Дело в том, что любой браузер по-умолчанию будет лепить страницу сайта в верхний левый угол. И это правильно! Порядок нужен во всем. Но что если наш сайт довольно узкий по ширине, а монитор дюймов на 20? Весь сайт сиротливо будет прижиматься к левой стороне монитора, а справа останется куча пустого пространства. Не слишком красиво, верно?

Вот поэтому нам нужно наш неширокий сайт разместить по центру экрана. Но как это сделать? Одним из способов как раз и является дополнительный контейнер. По-сути, это

пустой блочный тэг div с заданной шириной и отступами. Это как коробка, о которой мы говорили ранее. Именно в нее мы и будем укладывать наш сайт.

Как видите, мы в наборе правил для контейнера указали ширину в 760 пикселей, рамку светло-серого цвета #999 толщиной в 1 пиксель и какой-то хитрый отступ 0 auto. Что это значит? А это как раз и означает, что сверху и снизу отступ будет равен нулю, а по бокам этот блок будет равноудален от правой и левой стороны окна браузера, причем автоматически. То есть, при любом размере экрана монитора (в разумных пределах) наш сайт всегда будет находиться строго по центру.

Возникает резонный вопрос: а почему бы у тэга body не указать такую же ширину страницы и автовыравнивание? А ничего не выйдет! Браузеры не дураки (не все) и тэг body для них равносилен всему экрану монитора. Вы же экран не станете двигать? То-то же!

Предвосхищая вопрос о так называемой «резиновой» верстке, когда страница сама подстраивается под любой размер экрана (во всяком случае, это подразумевается, но работает не всегда корректно), то скажу прямо и откровенно: я против такого подхода.

Во-первых, это блажь чистой воды. Угодить всем просто невозможно. Кто-то уже приобрел монитор в 24 дюйма, а кто-то до сих пор щурится в 15. И ежу понятно, что на большом мониторе все элементы страницы расползутся как тараканы, а на маленьком сгрудятся в непонятную и перемешанную кучку. Оно вам надо?

Во-вторых, в дизайне, как и в любом другом виде творчества, центральное положение занимает гармония, а не количество контента на 1 см². В случае «резиновой» верстки мы получаем нарушение пропорций при изменении размеров монитора, и говорить о какой-то там гармонии уже не приходится.

В-третьих, следует учитывать психологию восприятия контента. Комфртно читается текст, когда в одной строке умещается максимум 10 – 12 слов. Если их будет больше (а при растянутом на большом экране сайте так и будет), то читать текст становится очень трудно, ибо глаз постоянно теряется на длинной строке.

Поэтому я против резиновой верстки.

Продолжаем разговор за CSS. Далее в листе стилей запишем несколько правил для шапки (header) сайта:

```
#header {
background: url(header.jpg) no-repeat;
width: 760px;
height: 158px;
}
```

Снова видим уникальный элемент с решеткой. Иначе такая запись называется также *селектор*. Напомню еще раз, что для таких элементов можно писать любые имена. Например:

```
#shapka
```

Суть от этого не изменится.

Здесь мы указали, что вся наша шапка залита фоном-картинкой с размерами 760x158 пикселей. Такая у меня вышла при разрезании фотошопного макета. Фон для какого-либо блока записывается в файле стилей в виде ссылки на картинку:

```
background: url(images/header.jpg) no-repeat;
```

url(images/header.jpg) — это ссылка на картинку. Здесь все просто. Пишем слово url и сразу за ним в круглых скобках адрес, где картинка находится. В нашем случае она лежит в папке images, которая в свою очередь находится в корневой папке сайта вместе с файлом стилей и непосредственно самой страницей index.html.

После указания адреса есть любопытное слово no-repeat. Означает оно ничто иное, как запрет на повтор картинки. Дело в том, что по-умолчанию браузер стремится залить фоновой картинкой все пространство блока, к которому относится. Нам это не нужно. Попутно замечу, что если бы нам потребовалось размножить фон только по горизонтали (как мы и сделаем потом с фоном для менюшки), то вместо no-repeat мы должны будем записать repeat-x, а если только по вертикали, то соответственно repeat-y.

Важное замечание! Между закрывающей круглой скобкой адреса и этим словом обязательно должен быть пробел. Иначе браузер IE не увидит картинки.

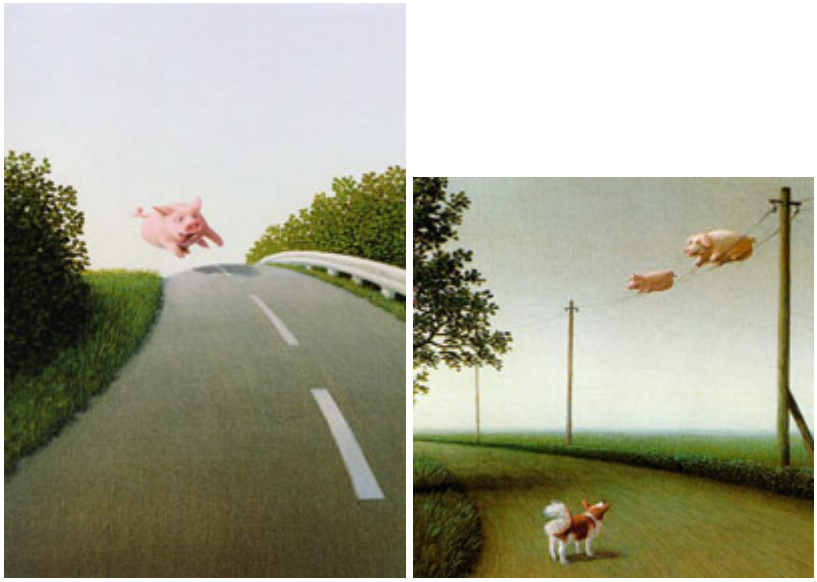
Кроме адреса фоновой картинки мы в этом наборе правил также указали ширину и высоту блока шапки. Это тоже обязательный элемент. Если этого не сделать, то браузер будет автоматически сжимать данный блок до размеров того полезного содержимого, которое мы положим в шапку. А нам нужно, чтобы была видна вся фоновая картинка шапки. Она у нас сделана как раз четко в размер сайта по ширине.

А теперь сохраним наш лист стилей в ту же самую папку, где уже лежит файл index.html. Сохраняем с именем style.css

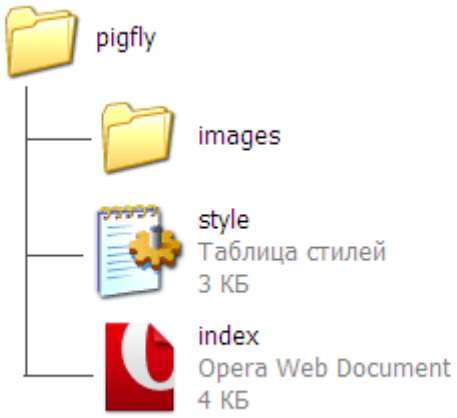
О фоновых картинках

Много про Фотошоп рассказывать не буду. Это тема для отдельной книги. Из нарисованного макета я вырезал целиком шапку, оба рисунка с поросятами, рисунок вензеля, волнистой зелёной линии, фон полосы навигации (не весь, а только отрезанный кусочек 8x35 пикселей) и зеленой галки для списка новостей.

Итого у нас вышло 7 изображений.



Их поместим в папке images, которую разместим в папке проекта pigfly вместе с файлом главной страницы index.html и файлом стилей style.css. Напоминаю об этом, чтобы вы не запутались. На всякий случай вот такая структура папок и файлов у нашего проекта:



Все просто.

Шапка сайта

Окей! Давайте уже прицепим к нашему сайту шапку и посмотрим, наконец, что у нас получилось. Открываем снова в текстовом редакторе PSPad нашу главную страницу — файл index.html. Между тегами <body></body> добавляем следующий код:

```
<div id="container">
<div id="header">
</div>
```

```
</div>
```

Сохраняемся. Открываем нашу страничку любимым браузером и любуемся на шапку. Что мы сделали? Мы добавили в тело страницы нашу "коробочку" — контейнер, а уже в него положили шапку. Опять же, наблюдаем последовательность открытия и закрытия тэгов. Вначале идет тэг div с селектором контейнера следом тэг шапки , далее тэг шапки закрывается , и затем так же закрывается тэг контейнера . То есть вложенность тэгов налицо.

div id — это и есть тэг с индивидуальным атрибутом, и id дает это явно понять. Атрибут же стоит после знака равенства и должен быть обязательно заключен в кавычки. Читается такая запись следующим образом: блочный тэг div с индивидуальным (id) селектором под названием "container".

С этим разобрались. Теперь давайте поместим в шапку что-нибудь полезное. На ум приходит сразу же заголовок сайта. Все заголовки имеют свои строго закрепленные тэги: h1, h2, h3 и т.д. Соответственно, их называют заголовком первого уровня, второго, третьего и т.д. Совершенно логично предположить, что заголовок первого уровня — самый важный на сайте, а стало быть, должен появляться лишь один раз. Обычно им обозначают название сайта.

Давайте добавим в пределах блока шапки следующий кусок кода:

```
<a href="http://www.piglfy.ru/"><h1>Летающие Парасенги</h1></a>
<p class="description">Сайт о поросенках-летунах и счастливой летучей жизни</p>
```

Мы взяли в тэги заголовка первого уровня <h1></h1> название нашего сайта, а сам заголовок сделали ссылкой с адресом нашего домена.

Кроме того, мы добавили описание сайта, видимое посетителям. Текст взят тот же самый, что и в метатэгах описания. Помните? Вот. Только теперь мы его поместили в тэги абзаца `p` с классом "description". Что это за класс такой?

Если помните, в главе про атрибуты id и class мы уже говорили о том, что класс — это атрибут, который может быть использован на одной странице сайта несколько раз. Как и у индивидуального атрибута id у него может быть произвольное имя. В нашем случае мы обозвали его "description".

Теперь давайте в файле стилей запишем для элементов шапки еще несколько правил:

```
h1 {
font-size: 250%;
font-weight: bold;
padding: 30px 10px 0;
}

#header a {
color: #ffc;
text-decoration: none;
}

#header a:hover {
color: #EBB0AC;
```

```
text-decoration: none;
}

.description {
font-size: 100%;
color: #fff;
padding: 0 10px;
}
```

Как видите, тэгу заголовка h1 мы задали размер шрифта в процентах — 250%. Можно было указать также пиксельное или относительное значение — px и em соответственно. Но об этом чуть позже. Также мы указали толщину шрифта — bold. И добавили поля (padding), чтобы заголовок не лепился к краям шапки.

Но ведь у нас в заголовке есть еще и ссылка. Для нее тоже надо прописать правила, чтобы было красиво, а не сине-фиолетово по-умолчанию. Здесь появляется новый тип записи — *последовательность*. Что это значит?

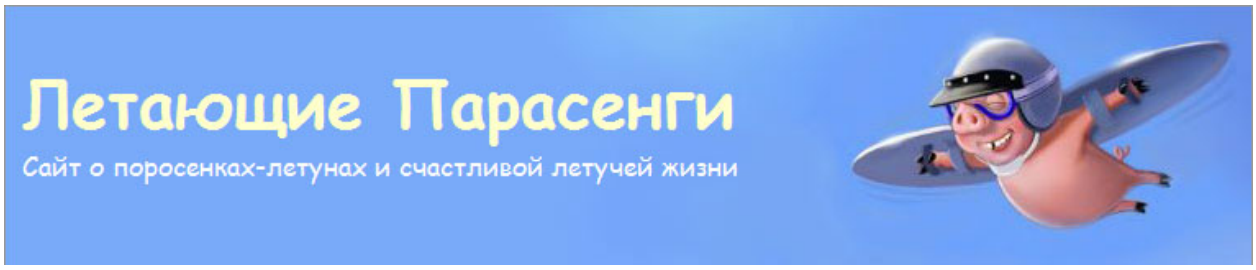
Если бы мы просто написали ряд правил для ссылок, обозначенных тэгом **a**, то оно автоматически применилось бы для всех ссылок сайта. А нам то нужно чтобы правила работали только для шапки. Поэтому мы и записали это дело таким образом: #header a. Это означает, что данные правила будут применены к блоку с селектором header и конкретно к тэгу **a**.

Для ссылки мы указали цвет в состоянии покоя #ffc (светло-желтый) и добавили, что подчеркивание не нужно (text-decoration: none;). По-умолчанию все браузеры ссылки подчеркивают.

Следующий набор правил предназначен для ссылки в активном состоянии, когда на нее наведена мышка. Для этого мы к тэгу ссылки приписали через двоеточие слово hover, а в наборе правил изменили цвет на #EBB0AC. Это что-то светло-розовое такое. Как видите, здесь уже все шесть знаков шестнадцатиричного кода цвета. Они не повторяются в парах, поэтому сокращенная запись уже не возможна. Взято просто в Фотошопе пипеткой с пуза поросенка из картинки в шапке.

И, наконец, набор правил для описания сайта. Здесь впервые появляется запись правил для класса. Вы видите, что теперь вместо решетки слева от селектора (имени) стоит просто точка. Она как раз и означает класс. В самом наборе уже знакомые вещи: размер шрифта, его цвет и поля. Последние вы можете регулировать как вам угодно, "играя" с взаимным расположением заголовка и описания.

Теперь сохраняем оба файла (страницу и стили) и смотрим, что у нас получилось в браузере. Должно быть так:



Полюбовались? Идем далее.

Навигация или меню сайта

Теперь продолжим писать код файла стилей для меню или иначе — главной навигации сайта. Следом за правилами для шапки запишем несколько правил для блока навигации:

```
#nav {
background: url(images/nav-bg.jpg) repeat-x;
color: #f00;
font-size: 120%;
font-weight: bold;
line-height: 1.8em;
text-align: center;
}

#nav ul {
list-style-type: none;
}

#nav li {
display: inline;
margin: 0 8px;
}

#nav li a {
color: #0c0;
text-decoration: none;
}

#nav li a:hover {
color: #f00;
text-decoration: none;
}
```

Уже немного сложнее, не правда ли? Разберём по косточкам.

Панель навигации будет у нас одна, сразу под шапкой, горизонтальная. Для её реализации мы воспользуемся таким элементом как маркированный список **ul**.

Для тех, кто не в танке объясняю: *маркированный список* — это список из нескольких пунктов, записанных в столбик, у которых слева вместо порядковых чисел стоят маркеры (кружки, квадратики, и пр.). Данный список в HTML обозначается тэгом **ul**. Элементы списка (а попросту говоря — строчки) обозначаются тэгом **li**. Выглядит это примерно так:

```
<ul>
<li>Утром надел трусы.</li>
<li>Не забыл про часы.</li>
<li>Снова не забыл.</li>
</ul>
```

Как видим, тэг li вложен несколько раз в тэг ul. Если добавить такой код на страницу сайта, то мы получим вот такой список:

- Утром надел трусы
- Не забыл про часы
- Снова не забыл

Нам для навигации не нужны маркеры, да и сами пункты должны быть вытянуты в строку, а не размещаться столбиком. Посмотрим, что можно тут сделать.

Блок навигации мы обозвали #nav. Вначале укажем общие настройки для него: бэкграунд — это картинка с именем nav-bg.jpg размером 8x35 пикселей, которая лежит в папке images. Ее, как вы помните, мы вырезали из фотошопного макета сайта. Почему такая мелкая? Дело в том, что она у нас повторяется по всей длине менюшки. Зачем вырезать такую длинную картинку, когда фон можно просто размножить средствами самого css? Вот как раз именно для этого мы указали в данном правиле слово repeat-x, что означает “повторить по оси x”, то есть по горизонтали (об этом уже говорилось ранее).

Далее мы указали цвет шрифта ярко-красного цвета #f00. Для чего это нужно, если меню по обыкновению делается ссылками на соответствующие страницы? Дело в том, что ссылаться на самое себя не принято в сообществе Правильных Верстальщиков. Это тавтология, если хотите. Поэтому текущий раздел (страница) сайта не должен быть ссылкой. Но цвет то ему надо какой-то придумать. Вот оно и…

Остальное в этом наборе правил несложно: размер шрифта, толщина, высота по вертикали и выравнивание текста по центру (text-align: center;). Тут у нас появилась новая единица измерения em, которая равна высоте прописной буквы выбранного шрифта. Это, как я уже упоминал, *относительный размер* шрифта. Относится он к настройкам браузера по-умолчанию. Каждый браузер имеет такие настройки. Так вот, значение 1.8em показывает, что шрифт будет увеличен на 1.8 относительно размера шрифта по-умолчанию. Это значение можно наоборот уменьшить. Для этого достаточно написать так 0.8 или просто .8 (точка и цифра). Число, разумеется, можно выбирать любое.

Следующее правило указывает, что у нашего списка не должно быть никаких маркеров. Вот оно, решение! Задаётся оно значением none для правила list-style-type. Это правило имеет несколько вариаций. Об этом расскажу позднее.

Теперь нужно избавиться от столбика. Для этого мы добавим строчкам списка (тэгам **li**) расположение по горизонтали, то есть в линию — display: inline; И еще добавляем отступы: сверху и снизу по нулям, с боков по 8 пикселей. Чтобы пункты меню не лепились тесно друг к дружке.

А теперь укажем, каким образом наше меню будет реагировать на наведение мышки. Пропишем ещё пару правил для ссылок. Вы видите, что здесь опять же присутствует некая последовательность: #nav li a. Такая запись означает, что данный набор правил будет применен к ссылкам **a**, которые находятся в тэгах **li** списка с именем **#nav**.

Что ж, с этой частью кода файла стилей разобрались. Давайте теперь добавим на страницу сайта наше меню. Следом за шапкой добавим блок навигации:

```
<div id="nav">
<ul>
<li>Главная</li>
```

```
<li><a href="#">О нас</a></li>
<li><a href="#">О летучести</a></li>
<li><a href="#">О везучести</a></li>
<li><a href="#">Свинки-герои</a></li>
<li><a href="#">Подружиццо</a></li>
</ul>
</div>
```

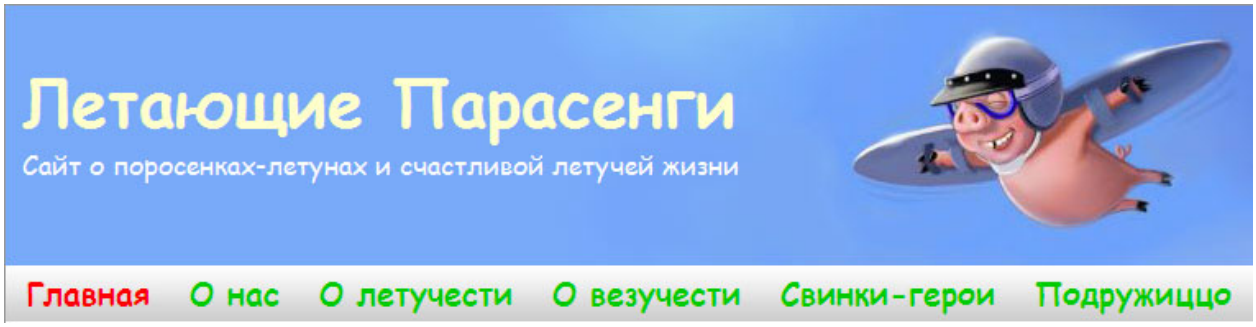
Как видим — всё просто: наши разделы оформлены как пункты списка, и каждый пункт, кроме первого, является ссылкой. В данном случае вместо адреса пока несуществующих страниц мы просто вставили решётку (#), которая всегда возвращает нас на текущую страницу.

Важный момент: первый пункт списка — не ссылка. Потому что сейчас мы верстаем главную страницу сайта. Зачем еще раз ссылаться на самое себя? И точно так же мы не станем ссылаться на других страницах сайта на самое себя. Соответственно, для каждой страницы сайта мы несколько видоизменим список меню, убирая ссылку там, где она не нужна.

Теперь настала пора сказать пару слов о *каскадности*. Если вы еще не забыли, то CSS переводится как *каскадный лист стилей*. Что это означает?

В правилах для блока навигации мы вначале указали настройки для всего блока #nav, затем для маркированного списка, обозначенного тэгом ul, далее показали правила для строк li и т.д. Каждое последующее правило получает "в наследство" характеристики предыдущего: от nav к ul, от ul к li. Все вместе они являются вложенными в блок контейнера и получают также и от него часть правил (в частности, центрирование посередине экрана и заданную ширину в 760 пикселей). Вот именно это и является своеобразным *каскадом*. Или иначе — *наследованием* правил.

А теперь сохраняем нашу страничку и файл стилей. И идём скорее смотреть в браузере, что у нас получилось:



Контент

Ну а теперь пора уже, наконец, наполнить нашу страницу чем-нибудь полезным, то есть **Контентом**. На макете видно, что полезная площадь страницы разделена на две функциональные области:

1. Основной текст (с картинками и пр.)
2. Блок новостей (справа).

Обычно, такую вёрстку называют *двухколоночной*. Как сделать так, чтобы блок текста оставался слева, а блок новостей справа я расскажу чуть позднее. А пока давайте запишем на страничке сайта сразу же после кода меню следующее:

```
<div id="text">
<img class="img1" src="images/pig1.jpg" alt="Летящий свин" />
<p>Летать всегда! Летать везде! Летать много, очень-очень много и
всегда с улыбкой на морде лица – вот наше кредо!</p>
<p>Все пиггасы рано или поздно приходят к осмыслению никчемной жизни в
грязной луже и подаются в лёччики-пилоты. Причём для летания вовсе не
нужна никакая посторонняя техника. Только сильное и несокрушимое
желание, а также упорство, спортивная злость и немного вредности. Оно
того стоит! Уж поверьте.</p>
<p>Всего лишь после недели тренировок на брезентовом батуте и трёх
зачотных прыжков с крыши сарая, адепт получает звание летуна-прыгуна.
При этом заработанные синяки, ссадины и шишки также засчитываются в
+</p>
<img class="venzel" src="images/venzel.gif" alt="" />
<img class="img2" src="images/pig2.jpg" alt="Пиггсы на проводе" />
</div>
```

Добавив приведённый выше код и сохранившись, откроем страницу в браузере и посмотрим, что получилось. Пока что фигня получилась. Текст появился, но всё наперекосяк. Это мы поправим в листе стилей, а пока давайте немного разберём, что же это мы намудрили.

Тэг `div` с атрибутом `"text"` — это и есть область полезного содержимого странички, иначе называемое *контентом*. Как и в любом тексте тут мы видим абзацы, обрамленные тэгами `p`, а также несколько изображений (тэги `img`).

С тэгами абзацев всё понятно: открылся один, за ним кусок текста, закрылся; открылся следующий, за ним опять кусок текста, закрылся, и т.д.
А вот тэги изображений у нас идут под разными классами. Потом мы в файле стилей зададим им различные правила, в том числе и так называемое *обтекание* — float.

Еще раз для тех кто в танке с заваренной башней:
Атрибут МОЖНО и НУЖНО прицеплять к существующему уже тэгу, будь то тэг абзаца, рисунка, списка и т.п. Не надо перенасыщать страницу div-ами. Div используется для основных, крупных функциональных блоков страницы, а не для каждой мелочи.

Теперь давайте красиво оформим основное содержимое страницы. Запишем далее в листе стилей CSS:

```
#text {
width: 545px;
font-size: .8em;
margin: 10px 0;
float: left;
}

#text p {
text-align: justify;
text-indent: 1.5em;
margin: 0;
padding: 0 15px;
```

```
}

#text a {
color: #396;
}

#text a:hover {
color: #f36;
border-bottom: #f36 dotted 1px;
}
```

В первом правиле мы указали, что ширина у области текста будет равна 545 пикселям. Размер шрифта 0.8em. Как я и говорил, в данном правиле ноль можно не писать, .8em — обозначает тоже самое. С отступами margin все понятно — верх и низ по 10 пикселей, по бокам 0. А вот последняя строчка как раз и задаёт местоположение нашего блока текста ни где попало, а с левой стороны.

Слово float переводится как "обтекание". Но тут есть одна фишка. Читаем: float: left — обтекание слева. Но ведь это сам этот блок с текстом находится слева! А колонка новостей "обтекает" его справа. В этом есть некая путаница. Чтобы не париться, просто запомните: left — сам объект слева, а течёт всё правее. И наоборот, right — объект справа, а течёт всё левее.

Для чего это нужно? Добавив приведённые выше правила в свой лист стилей и сохранившись, посмотрите, что получилось — блок контента выровнялся по левому краю странички, оставив справа пустое место. В это пустое место мы потом и вставим блок новостей, присвоив ему в листе стилей значение right для атрибута float.

Тем самым мы видоизменили основной постулат блочной верстки, когда каждый блочный элемент размещается на новой строке, то есть один под другим. Если бы не эти "обтекания", то нам бы не удалось разместить два (или более) блока рядом.

Идем далее. В следующем наборе правил мы для абзацев нашего текста задали выравнивание по всей выделенной площади. Слово justify как раз это и означает. Если этого не указать, то по умолчанию весь текст выровняется по левому краю. В англоязычных странах это всегда было нормой, и норма эта исходила из размеров английских слов и букв. Но в кириллице такое выравнивание смотрится неаккуратно — весь правый край текста становится как бы рваный. Поэтому мы выровняли его по обоим краям. Это не выравнивание по центру! Это скорее равномерное распределение слов в строке.

Правило text-indent означает не что иное как обычную "красную строку". Размер её также указан в относительных единицах em. Можно писать и отрицательные значения. Тогда получится как бы обратная красная строка. Кстати, если хотите скрыть часть текста какого-то блока, то можете задать большую отрицательную величину. Скажем, -1000em. И тогда текст просто не будет виден на экране монитора, спрятавшись далеко за его пределами слева.

Ну и, наконец, ссылочки. Для неактивной задали салатовый цвет, а для активной красный, да ещё и с точечным (dotted) подчёркиванием в 1 пиксель толщиной.

Что ж, кроме текста у нас в блоке контента есть несколько картинок. Давайте запишем в листе стилей следующее:

```
.img1 {
width: 200px;
height: 287px;
margin: 0 0 0 15px;
float: right;
}

.img2 {
width: 200px;
height: 200px;
margin: 10px 10px 0 15px;
float: left;
}

.venzel {
width: 300px;
height: 23px;
margin: 10px 10px 0 15px;
float: left;
}
```

Здесь также нет ничего сложного. Каждую картинку мы обозвали своим именем: img1, img2 и venzel, указав в каждом правиле размеры картинок и отступы для них. Кстати, значения можно указывать и с минусом. Например -10px. И картинка сдвинется в противоположную сторону, хоть даже и за край экрана. Однако не все браузеры это любят. Так что проверяйте в разных, как оно получается.

Каждой картинке мы задали обтекание в соответствии с её расположением на странице. Первая картинка будет справа от текста, вторая — слева, и вензель тоже слева.

Зачем в листе стилей указывать размеры картинок, если они не фоновые? Даже если вы этого не сделаете, то картинки все равно будут показаны. Но по обыкновению браузер вначале всегда грузит текст, а потом картинки. Если ему заранее не указать их размеры, то возникнет неприятное дергание содержимого страницы по мере подгрузки картинок. Так что лучше заранее как бы зарезервировать под них соответствующее место.

Теперь сохраняемся и любуемся на то, что у нас получилось. Если всё выполнили правильно, то на страничке будет красиво выровненный текст с рисунками свинок-летунов и завитушкой-вензелем под текстом.

Нумерованный список

В блоке текста мы разместим список новых участников. Вообще на сайтах за это отвечает какой-нибудь php-скрипт. Но, так как мы делаем простую статичную страницу, то со скриптами не заморачиваемся. Сейчас нас больше интересует конкретная штуковина под названием *нумерованный список*. Задаётся такой список тэгом `ol`.

Откроем в текстовом редакторе нашу страничку и сразу после вот этого места:

```
<img class="img2" src="pig2.jpg" alt="Пиггсы на проводе" />
```

вставим следующий кусок:

```
<div id="members">
<h2>Список новых учаснегов:</h2>
<ol>
<li><a href="#">Рыжий</a></li>
<li><a href="#">Брат Корнелий</a></li>
<li><a href="#">Муха</a></li>
<li><a href="#">Пигфлай</a></li>
<li><a href="#">Нигга Боб</a></li>
<li><a href="#">Помидорка</a></li>
<li><a href="#">Косолапыч</a></li>
<l><a href="#">Тушка</a></li>
<li><a href="#">Свин Полезный</a></li>
</ol>
</div>
<img class="line" src="images/line.gif" alt="" />
```

Что мы тут видим? Появился новый селектор members. Так мы обозвали наш список. Тэг h2 — это заголовок второго уровня. Сразу после списка мы положили картинку волнистой линии.

Ну, а теперь откроем наш лист стилей и запишем ещё несколько правил.

```
#members {
width: 300px;
float: right;
}

#members h2 {
color: #f60;
font-size: 120%;
font-weight: bold;
text-align: center;
}

#members ol {
color: #999;
font-size: 120%;
margin: 10px;
}

#members li {
margin: 0 5px;
}

#members li a {
color: #0c0;
}

#members li a:hover {
color: #f00;
}

.line {
width: 304px;
height: 13px;
float: right;
```

```
}
```

Расшифруем. Для начала мы задали всему блоку со списком размер в 300 пикселей по ширине и обтекание справа. Ну, это уже понятно — список должен быть правее рисунка с собакиным и сидящими на проводах поросятами.

Далее задали правило для заголовка второго уровня. Здесь вам уже всё знакомо. Заметьте, что здесь тэг заголовка второго уровня **h2** принадлежит блоку #members.

Для самого нумерованного списка с тэгом **ol** мы на этот раз не указывали способ отображения меток (list-style-type), как в маркированном списке. Таким образом, по-умолчанию каждому пункту списка будет автоматически присвоен порядковый номер.

Все имена в списке оформлены в виде ссылок, типа на странички профиля участников. Для них мы задали только цвета, без всяких подчёркиваний. Но почему-то на страничке они всё равно подчёркиваются при наведении мышки, да ещё точечной линией! Вот тут как раз и сработал так называемый *каскад*. Список ведь лежит в зоне действия блока **text** и поэтому просто перенял от него часть правил. В том числе и для ссылок.

Последнее правило здесь для рисунка линии. Оно так же, как и предыдущие картинки, оформлено классом с именем .line и имеет обтекание right. Тем самым мы как бы подчеркнули список участников. Ну, просто декорация такая.

Сохранились. А теперь смотрим, что получилось.

Боковая колонка или сайдбар

Боковая колонка или иначе *сайдбар* (sidebar) — чаще всего является местом для вывода навигации второго уровня (рубрики, последние новости, ссылки на другие ресурсы, кнопки-счетчики и т.п.). В нашем примере мы в боковую колонку выведем последние новости. Понятное дело, что это будет лишь статичный текст. Обычно новости выводятся при помощи специальных скриптов. Наша задача: просто разместить вторую колонку рядом с блоком контента. Как я уже и говорил: этот тип верстки называется *двухколоночным*.

Поехали!

Открываем нашу страничку index.html и следом за рисунком волнистой линии вставляем вот этот кусок кода:

```
</div>
<div id="news">
<h3>Самые распоследние новости:</h3>
<ul>
<li>Всю прошедшую неделю лил жуткий дождь, и полёты временно
приостановились. Самые безбашенные пиггасы, однако, всё равно
кучковались стаями на проводах местной радиолинии и дружно создавали
помехи. Малаццы!</li>
<li>Пиггас Хмурый Пятак снова хмурый. Обещал всех урыть. Злой
сильно.</li>
```



```
<li>У нашего друга Боббса завтра ДР! Поздравления и подарочки просил
вручать возле новой будки и непременно на виду у соседского пса
Мухомора, чтобы тому завидно стало. Пляски намечаются до самого утра.
При наличии на небе луны — будет весело.</li>
</ul>
</div>
<div class="clearfloat"></div>
```

Что мы здесь видим? Во-первых, закрывающий тэг , который показывает, что область блока "text" с контентом закончилась. Далее идет div боковой колонки с именем "news". В нем мы снова применим маркированный список новостей, прилепив сверху заголовок 3-го уровня `h3`.

Сложного здесь ничего нет. После закрытия этого "дива" мы добавили еще один любопытный блок div с классом clearfloat. Уже по одному только названию можно понять, что это очистка от всяких обтеканий. А вот зачем она, поясним подробнее:

Вообще наша вёрстка является *плавающей*. Вон сколько у нас уже обтекаемых элементов. Вполне возможен случай, когда нужно прекратить все эти обтекания. Ну, например, если какую-то картинку обтекает текст, но его мало. Тогда следующий за текстом элемент (другой текст, картинка и пр.) будет стремиться также заполнить оставшееся пространство. Но нам это не нужно, ибо ломает всю картину. Вот как раз для таких случаев придуман пустой блок, которому в листе стилей задано специальное правило. Об этом чуть позже.

А теперь в листе стилей допишем следующий код:

```
#news {
background: #ffc;
width: 185px;
color: #665;
margin: 10px 5px;
float: right;
}

#news h3 {
color: #f60;
font-size: 120%;
font-weight: bold;
text-align: center;
}

#news ul {
list-style: url(marker.jpg) inside;
}

#news li {
font-size: 75%;
padding: 5px 10px;
}

.clearfloat {
clear: both;
}
```

Здесь мы для начала поменяли фон для блока новостей, чтобы визуально отделить колонку от основного содержимого. Затем задали ширину блока, цвет для шрифта и обтекание справа.

С заголовком 3 уровня **h3** получилось хитро. Если вернуться чуток назад, то можно заметить, что у заголовка 2-го уровня **h2** также задан размер шрифта в 120%. Но по какому-то странному стечению обстоятельств размеры у h3 существенно больше, нежели у h2. Как такое возможно? Притом, что в обоих случаях в файле стилей был задан один и тот же размер в 120%.

Вообще-то по-умолчанию браузеры сами определяют размеры для заголовков в зависимости от их ранга. То есть по логике вещей, заголовок 3-го уровня должен быть мельче 2-го. А у нас наоборот! Это произошло потому, что заголовок h2 в нашем случае находится в блоке основного текста, для которого мы задали размер всего шрифта в 0.8em. Таким образом, этот заголовок уже изначально получил стартовый размер меньше, чем основной шрифт сайта. Помните о наследовании? Вот здесь он как раз и сработал.

Следующим правилом мы задали для списка стиль маркеров. По-умолчанию любой браузер отображает маркеры в виде чёрных кружков. Но можно задать и другое отображение — окружностей, квадратов или вообще без маркеров. А можно и свой собственноручно нарисованный маркер. Так мы и поступим.

Помните симпатишную зелёную галку среди заготовленных картинок? Ее то и используем в качестве маркера. Для этого в наборе правил для списка ul пропишем правило list-style и добавим в качестве значения адрес этой самой картинки — url(images/marker.jpg). Следом приписали слово inside. Что это значит?

Маркеры по-умолчанию не входят в сам блок, а выносятся за его края. Это не всегда есть хорошо. Поэтому мы принудительно указали им быть inside, то есть внутри колонки с текстом.
Ну и, наконец, несколько слов о последнем здесь правиле — очистке. Как видите, все просто: класс с именем clearfloat имеет всего лишь одно значение — clear: both; Что означает "очистка с обоих сторон". То есть, после этого блока кончается действие предыдущего правила обтекания. Само собой разумеется, что имя можно придумать любое. Просто это выглядит боле менее "говорящим". Вот и все.

Сохраняемся и смотрим, что вышло. Если сделали все правильно, то получите практически полностью собранный сайт. Остается только оформить подвал. Чем мы и займемся.

Подвал сайта

Подвал, он же *футер* — это весьма важная часть сайта, хотя и мало кто туда добирается, особенно при очень длинных страницах. Там обычно дублируются ссылки на разделы сайта, пишутся копирайты и контактная информация.

Футер не должен доминировать над шапкой, но и не должен теряться, делая страницу неуравновешенной. Мы сделали его немного контрастным по цвету, но при этом небольшой высоты.

Запишем в коде страницы сразу же после очистки оставшийся код:

```
<div id="footer">
<p>Главная | <a href="#">О нас</a> | <a href="#">О летучести</a> | <a
href="#">О везучести</a> | <a href="#">Свинки-герои</a> | <a
href="#">Подружиццо</a></p>
<p>&copy; Pigfly.ru, 2007 – 2011 | Все поросячие права защищены. Не
ешьте свинины, будете здоровы!</p>
</div>
```

Здесь у нас слово "Главная" опять же не является ссылкой (об этом мы говорили), а следом через знак вертикального разделителя | идут обычные ссылки на другие страницы сайта. То есть, обычный повтор главного меню. Только на этот раз без использования маркированного списка `ul`.

Всего в футере два абзаца. В первом ссылки на страницы, а во втором как раз и есть всякие копирайты, год, права и пр. Обратите внимание на любопытное слово `&copy;` Именно так в HTML прописывается так называемые *символьные объекты*. Браузеры умеют преобразовывать их автоматически в соответствующие значки. Указанный значок ни что иное как знак копирайта — буковка С в круге.

Вот практически и вся страница. Проверьте только, чтобы в конце кода у вас было два закрывающих подряд. Первый закрывает блок боковой колонки, а второй — блок общего контейнера.

Теперь допишем в листе стилей оставшийся код:

```
#footer {
background: #665;
color: #fff;
font-size: 70%;
padding: 5px;
}

#footer a {
color: #ff0;
}

#footer a:hover {
color: #f00;
}

#footer p {
padding: 2px;
text-align: center;
}
```

Здесь мы задали фон подвала серо-зелёного цвета, а цвет текста — белый. Ссылки у нас здесь ярко-жёлтые, а в активном состоянии — красные. Для текста мы выбрали размер мельче всех на странице — 70%. Как я и сказал, футер — важная часть, но не настолько, чтобы бросаться в глаза.

Выравнивание для всего текста в подвале мы задали по центру.

Вот, собственно, и всё! Результат вы можете посмотреть в прилагаемом к книге готовом сайте в папке pigfly. Там же можете свериться с кодом страницы и файла стилей, если вдруг что-то упустили.

Полезное

Здесь я решил собрать некоторые подсказки по синтаксису CSS. Своеобразная шпаргалка, которой пользуюсь часто сам:

1. **Свойства шрифтов**.

font-family — семейство шрифтов. Обычно пишется так:
font-family: verdana, arial, sans-serif;

То есть, на выбор предложено в порядке убывания либо шрифт verdana, либо arial, либо любой другой без засечек. Слово sans-serif как раз и означает, что шрифт без засечек. Делается такая запись по той причине, что на компьютере посетителя нашего сайта может вдруг не оказаться какого-то из указанных шрифтов. Тогда браузер выберет что-то похожее и близкое.

font-style — стиль написания шрифта. Может быть как normal, так и italic (*наклонный*).
Пример: font-style: italic;

font-weight — normal или bold. Соответственно, нормальный либо жирный.
Пример: font-weight: bold;

font-size — размер шрифта. Указывается обычно либо в процентах, либо в относительных величинах em, либо в пикселях px.
Примеры: font-size: 120%;
font-size: 1.2em;
font-size: 14px;

2. **Свойства текста**.

text-align — выравнивание текста. Значения могут быть следующие: left, right, center, justify. О последнем я уже упоминал — это равномерное распределение слов в строке.

text-indent — «красная строка». Указывается либо в % либо в пикселях.

line-height — высота строки. Весьма полезная фишка, когда надо выровнять разнокалиберный шрифт.

3. **Свойства цвета и фона**.

color — задает цвет шрифта. Задается он чаще шестнадцатиричным числом вида #000000. Как я уже и говорил ранее, при одинаковых числах в парах можно делать сокращенную запись #000.
Пример: color: #fff;
color: #f4f5f7;

Также возможна текстовая запись названий основных цветов. Пример: color: red;

background — фон. Если мы не используем какую-либо картинку в качестве фона, то задаем так же, как и цвет для шрифта:

background: #fff;

Если используем картинку, то все равно указываем фоновый цвет. Например:
background: #333 url(images/bg.gif) no-repeat;
Это нужно на тот случай, когда посетитель намеренно отключает картинки в браузере.

4. **Свойства рамки**.

border — рамка. Имеет толщину, цвет, фактуру и местоположение. Обычно пишется таким образом:
border: #333 solid 1px; — запись означает, что рамка темно-серого цвета, сплошная, толщиной в 1 пиксель. Другие значения фактуры: dotted — точечная, dashed — пунктирная, double — двойная (у этой толщина должна быть никак не меньше 3 пикселей, иначе выйдет одинарная).

Местоположение рамки также легко обозначить в правилах:

border-top — вверху
border-bottom — внизу
border-left — слева
border-right — справа

Можно задать различные цвет или толщину рамки сразу для всех 4 сторон объекта. Например запись:
border-color: #ccc #f4f5f7 #333 #000; — означает, что цвет верхней рамки светло-серый (#ccc), справа #f4f5f7, снизу #333, слева #000. Точно так же можно задать и толщину. Принцип схож с записью полей или отступов.

5. **Свойства списков**.

Задается свойство следующим правилом:
list-style-type:

У маркированного списка маркеры могут быть следующего вида:

disc — круг
circle — окружность
square — квадрат
none — отсутствует
либо, если мы хотим использовать свой рисунок маркера, то так:
list-style-image: url(images/bullet.gif);

Понятно, что картинка bullet.gif уже должна существовать в папке images вашего сайта.

Для нумерованных списков можно также задать различное отображение номеров:

lower-roman — римские цифры в нижнем регистре
upper-roman — то же, но в верхнем регистре
none — отсутствует.

6. **Свойства изображений**.

Пару слов об обтекании рисунка текстом. На самом деле нет ничего сложного. В листе стилей пишем class, для которого указываем необходимое направление обтекания и отступы для рисунка. Например, чтобы рисунок оставался на странице слева, а текст обтекал его справа, пишем следующий код:

```
.pic {
float: left;
margin: 0 10px 10px 0;
}
```

Это означает, что для любого рисунка с примененным к нему классом pic, обтекающий текст будет располагаться справа, сам рисунок будет иметь справа и снизу отступы по 10 пикселей, чтобы текст не лепился к нему вплотную.

На этом все! Следите за новостями на сайте Websovet.Com ибо не за горами новая книга о современной блочной верстке, HTML5, CSS3, jQuery и не только.

Вэлкам!