WWW.WEBSOVET.COM

Автор: Игорь Квентор

Содержание

От автора 3
Блочная верстка | Урок 1 4
  Тэг 6
  Тэг div 6
Блочная верстка | Урок 2 8
  DOCTYPE 9
  Тэг <html> 10
  Тэг <title> 11
  Тэг <body> 11
Блочная верстка | Урок 3 13
Блочная верстка | Урок 4 17
Блочная верстка | Урок 5 22
Блочная верстка | Урок 6 25
Блочная верстка | Урок 7 28
Блочная верстка | Урок 8 31
Блочная верстка | Урок 9 34
Полезняшки 38
Приложения 42

От автора

Привет! Вот я наконец-то и дозрел до оформления своего цикла статей о блочной верстке веб-сайта в настоящую электронную книгу в формате PDF. Книга совершенно бесплатна, и если кто-то вам будет предлагать ее купить, то знайте, что это гнусный барыга и шарлатан. Ибо даже на обложке белым по красному начертано «FREE», то есть — халява. ☺

Бесплатно — не означает, что книга ни о чем. Как раз наоборот, здесь предельно доступно и просто, в нескольких уроках, я рассказываю о таких сложных для новичка вещах в веб-мастеринге, как HTML и CSS. Не секрет, что для овладения навыками веб-строительства, необходимо изучать язык разметки веб-страниц — 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, а в особенности пользователи, визуальных редакторов типа «что вижу, то и получаю», все равно рано или поздно пытливый ум начинающего веб-мастера засунет нюх во всю скрытую от посторонних глаз кухню страничного кода и захочет таки разобраться что тут и как. Как, впрочем, и любой фанат «визуальщины» нет-нет, да и признает, что «хорошо бы, конечно, все-таки выучить HTML, ибо надо…».

И тогда человек идет в лавку и покупает том о семистапятидесяти страницах и двух кг весом, а затем, собравшись с духом, да на волне первоначального энтузиазма, погружается в мудрое чтиво. 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 это у кого хочешь челюсть свернет от зевоты.

Потому 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, принятых, уважаемым всеми Настоящими веб-мастерами, Консорциумом W3C. 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 азбука.

Enjoy! ☺

Блочная верстка | Урок 1

Данный цикл статей-уроков изначально был опубликован в моем блоге Вебмастеринг. Решил перенести его сюда с некоторыми поправками, дополнениями и уточнениями.

Итак, приступим:

Что есть такое блочная верстка и с чем ее едят? Ранее сайты верстали при помощи таблиц. Каждый элемент страницы — заголовок, текст, картинка — располагался в собственной ячейке, ячейки кучно роились в других, более крупных ячейках, 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 мало-мальскую ячейку нужно обозначить определенными тэгами. 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 ненужного мусора. Поэтому более не станем говорить за таблицы, а таки плавно перейдем к блочной верстке.

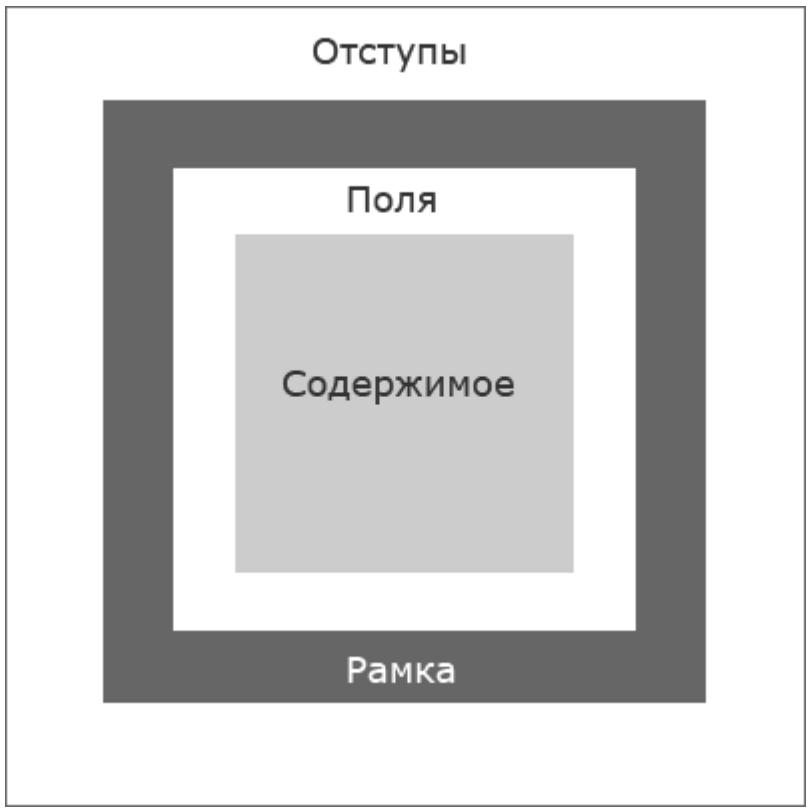
Блочная верстка позволяет обойтись без жесткой конструкции табличной сетки. Принцип тут простой: каждый элемент страницы, будь то картинка, кусок текста (абзац), табличка, список и пр. называется блоком. Блоки, как кирпичики, располагаются на странице в режиме "*естественного потока*". Это как будто 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.

У блоков есть одна интересная особенность: каждый из них считает, что недостойно находиться на одной горизонтали (читай — строке) с другим блоком. Поэтому по-умолчанию блоки всегда располагаются один под другим. Это умолчание можно подкорректировать и расположить несколько блоков на одной линии. Но об этом чуток попозже.

То есть, мы не делим пространство веб-страницы на ячейки как в табличной верстке, а словно бы складываем в коробочку все элементы. В этой самой "коробочке" они плотно укладываются один к другому. В результате мы избавляемся от скелета, а стало быть, и от массы ненужного

кода. 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. Да и разбираться в нем стало намного проще.

Что же такое этот самый блок? Блок — это обычная прямоугольная область, обладающая рядом свойств, таких как: рамка, поля и отступы. Содержимым блока может быть что угодно — кусок текста, картинка, список, форма для заполнения, меню навигации и т.п.



Рамка (border) — это контур, для которого можно задать такие характеристики как толщина, цвет и тип (пунктирная, сплошная, точечная).

Поля (padding) — отделяют содержимое блока от его рамки, чтобы текст, например, не был “впритык” к стенкам блока.

Отступы (margin) — это пустое пространство между различными блоками, позволяющее на заданном расстоянии расположить два блока относительно друг друга.

Из картинки видно, что внутреннее содержимое, например кусок текста, всегда имеет рамку. Она может быть как видима, так и невидима. В последнем случае у нее просто толщина равна нулю. Сам текст может как иметь поля (расстояние от рамки), так и не иметь их. В свою очередь, рамка тоже может иметь отступы от других таких же блоков, или прижиматься к ним плотно. Это позволяет гибко регулировать размещение блоков на странице, задавая цифровые значения отступам и полям.

Тэг

Тэг — это особая конструкция языка HTML. Можно сказать — буква алфавита. Различают открывающий и закрывающий тэги. 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:

** Текст абзаца. **

всегда обозначается буквой “p” и никак иначе.

Тэги всегда заключены в угловые скобки. Как видно из примера, наличествуют как открывающий тэг, так и закрывающий. У последнего перед именем добавлен “*слэш*” **/** — косая черта, наклоненная вправо. Для самых любознательных: данный значок расположен на клаве во втором ряду снизу справа в английской раскладке, а не во втором сверху в русской. Сверху — это совершенно другой знак, хотя и похож.

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:

****

Тэг div

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. Его используют для задания *функциональных областей* на странице, таких как: “шапка” (header), блок навигации, блок основного содержимого, “футер” (footer) или подвал по-нашему.

У данного тэга, как и у любого другого, имеются свои собственные атрибуты.

Атрибут — описательная характеристика тэга. То есть, что он может делать и каким образом. Например, опять же возьмём тэг изображения:

****

В данном случае src, width, height, alt являются атрибутами тэга. В кавычках (и это тоже обязательное требование современных стандартов) даны значения атрибутов. Расшифровать такую запись несложно. Тэг указывает, что в данном месте страницы нужно прилепить изображение **risunok.jpg** из папки **images**, шириной 200 пикселей, высотой 50 пикселей.

И еще добавлен альтернативный текст, который всплывает на страничке при наведении мышки на рисунок. Это не блажь, а тоже необходимое требование. Не все пользователи 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 маркер списка или стрелка какая), то у атрибута alt оставляют пустое место — **alt=""**.

Итак, атибуты тэга **div**. Наиболее часто используют два вида:

id — атрибут, позволяющий придать тегу уникальное значение, то есть такое, которое на странице используется **только один раз**. Например **“header”** (про кавычки не забываем), или **“footer“**. Таким образом мы сможем задать затем в листе стилей для тэга **div** с атрибутом **id** и значением **“header”** (шапка) одни настройки, а для подвала **“footer”** совсем другие, и браузер верно распознает, что вот этот абзац, обозначенный тэгом **<p>** относится к “шапке” и будет набран крупным и красным шрифтом, а вот этот к “подвалу” и будет мелким и серым. И никакой путаницы!

class — атрибут, позволяющий одно и то же значение придать **нескольким элементам**. Например всем изображениям на странице добавить рамку одинаковой толщины и цвета. Так как изображений несколько, то атрибут **id** уже нельзя использовать.

Блочная верстка | Урок 2

Для начала соберем небольшую кучку необходимейших инструментов.

Заходим вот на этот симпатичный сайт — www.pspad.com — и скачиваем замечательнейший редактор текста и программного кода PSPad. Замечателен он многим, но самое главное — им очень удобно работать. Попользуетесь и поймете все сами.

Далее, нам понадобится Фотошоп. Не Корел, не ПайнтШопПро и иже с ними, а обычный такой фотошопчег версии 7 или 8. Выше не рекомендую, ибо лицензионную вы вряд ли станете пользовать :) а "рыночный" вариант хорош до 8-го номера (по-другому обзывается CS). Все что выше, 9 и 10 — это куча глюков и ничего более. Я пользуюсь "восьмеркой" и тихо радуюсь. Важно(!): ставьте ТОЛЬКО английскую версию. По ходу езыг хоть немного вспомните. Пригодится.

Еще нам, конечно же, понадобятся средства для отображения наших неимоверных усилий в сайто-строительстве, а именно **браузеры**. Я не зря сказал это во множественном числе, ибо сие есть горькая (пока еще) правда. Горькая потому, что каждый браузер считает себя круче других и некоторые куски кода показывает исключительно по-своему. Тогда как сосед его, презрительно морщась, выруливает в другую степь. Отчего наш красивый во всех отношениях сайт у одного браузера вылезает чуть вбок, у другого раздается чуть вширь, а у третьего вообще шапка взаместо валенок и наоборот.

Поэтому нам понадобятся как минимум три самых распространенных на данный момент браузера: **Interner Explorer, Opera** и **FireFox**. Устанавливаем всех троих (IE стоит по умолчанию у всех пользователей Виндовза).

Верстать мы будем вот такой вот симпатишный сайт о Летающих Парасенгах.

Настоящий сайт [Pigfly.ru - Об удаче, Счастье и Богатстве]() выглядит уже несколько по-иному. Мы рассмотрим лишь отвлеченный пример.

DOCTYPE

Для начала определим тип нашего документа. Любая грамотно свёрстанная страница должна в самом начале содержать так называемый **DOCTYPE**. Нужен он не для форсу бандитского, а для всевозможных устройств вывода информации и браузеров в том числе. Пока что все ныне существующие браузеры прекрасно обходятся и без указания доктипа. Но уже грядут те времена, когда страница сайта, сразу начинающаяся с тэга **<html>** просто не будет прочитана, потому как стандарты становятся жесче.

Мы определим нашу страничку в соответствии с самым крутым и строгим на данный момент доктипом под названием **Strict 1.0**.

Выглядит сей крендель так:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

Прошу обратить внимание: как я уже говорил, требования здесь весьма строгие — все тэги, не имеющие закрывающей пары, должны заканчиваться пробелом со слэшем / перед закрывающей угловой скобкой. Но вот ведь сам доктип тоже выглядит как тэг! Почему же у него нет этого пробела со

слэшем? А просто! Захотелось так разработчикам сих строгих правил. Но это единственный случай, где правило не работает.

Тэг <html>

Далее мы впишем ещё одну хитрую строку:

<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">

Ну, раз тут появилось слово **lang** и **ru**, то и ежу понятно, что это указание на езыг документа. Не путать с кодировкой! Её мы укажем позднее. В этой же строке, в общем-то, и начинается уже код самой страницы с тэга

<html>

Следующий тэг **<head>**. В нём содержится всякая служебная информация: название страницы (то, что входит в титл), кодировка, ключевые слова для поисковых роботов, описание страницы и т.п. Вся эта инфа записана в служебных тэгах **meta**, которые также не отображаются на странице в браузере.

Мы запишем на нашей страничке следующую информацию:

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

Поясним:
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.

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.

Тэг <title>

Следующим парным тегом будет титл — это тот текст, который появляется на самом верху браузера при открытии странички:

<title>Пиггасы | Главная</title>

Понятное дело, что туда необходимо вписать название страницы. Не Index, не default, не просто цифра 1 какая, а настоящее, хорошее название. Это также необходимо, как и название для книги. Да и поисковые роботы сильно уважают, когда в титле набран вменяемый текст.

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> — НЕПРАВИЛЬНО!

Тэг <body>

Вот мы и добрались до тЕльца нашей странички, которое и обзывается соответствующим тэгом **<body>**. Пока давайте допишем код страницы до конца (он ещё будет впоследствии дополнен):

<body>
</body>
</html>

Сохраним документ в отдельной папке. Сохраняем как **index.html**. Почему именно index? Любой сервер, при заходе пользователя по адресу www.piggs.ru станет сразу же искать у себя в заначке страницу с этим названием. Устроены они так. Индексная страница для них всегда является главной, то есть — стартовой.

В следующем уроке займемся написанием листа стилей **CSS** для нашего сайта.

Блочная верстка | Урок 3

Итак, **CSS** или иначе *Каскадный Лист Стилей*. Почему каскадный объясню позднее. Открываем новый документ в PSPad (Блокноте, WordPad-е и т.п.) и вписываем следующую строку:

/*© PIG.RU, 2007 | piggs@pig.ru */

На самом деле содержимое строки не так важно. Она у нас закомментирована слэшами со звёздочками **/* */** .

Строка комментария не читается браузерами. Она нужна только для служебной информации. Кодеры обычно вписывают туда свои пометки, дабы самому не запутаться в коде, когда его много. В самом верхнем комменте (как у нас) пишутся ссылки на сайт кодера или дизайнера, его мыло, авторские права и т.п. Мы ограничимся только копирайтом и мылом.

Ну а теперь непосредственно код!
В листе стилей код называют **правилами**. Каждое правило состоит из **селектора** (читай - атрибута) и **объявления**. **Объявление**, в свою очередь, состоит из **свойства** и **значения**.

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. В данном случае значение записано в виде шестнадцатиричного числа, обозначающего цвет. Всем, кто пользуется Фотошопом это должно быть известно. Почему всего три нуля, ведь в данном обозначении должно быть шесть цифр (или букв)? Когда пары знаков одинаковы 00 00 00, то допускается писать сокращённо — 000. Все браузеры это понимают правильно.

Правило можно писать как угодно — хоть в строку, как у нас, хоть в столбик — это роли не играет. Важно только не забывать две вещи:
1. После каждого Свойства необходимо ставить двоеточие.
2. После каждого Значения — точку с запятой.
Если пропустить хотя бы в одном месте эти знаки, то IE и Firefox будут глючить, а Opera, возможно, и покажет всё без ошибок.

Сначала зададим общие правила для страницы:

```
* {
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;
}
```

Поясним:

1. В первом правиле звёздочка означает не что иное, как всю страницу разом. 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. В правиле мы указали последовательно:

Отступы - 0,
Поля - 0,
Рамка - 0.

Это нужно для того, чтобы ни поля, ни отступы, ни рамки не появлялись там, где нам не нужно. Если этого специально не указывать, то например тот же IE (Internet Explorer) напихает этого добра куда ни попадя, испортив нам всю кухню. Уж лучше мы в следующих правилах сами добавим что нужно и где нужно. Значения указываются либо в процентах, либо в пикселах. Если стоит ноль, то единицу измерения не нужно указывать.

2. Следующим правилом мы задали для тела страницы следующие указания: 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**.

Вообще 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 левое (для особо дотошных, кто никак не может определиться откуда начинать смотреть: мы смотрим на экран монитора в упор, право — это там где часики, а лево — где кнопка Пуск).

Так как у нас по бокам должно быть одинаковое расстояние от края экрана (в нашем случае ноль, то есть на всю ширину экрана), то и значений мы дали всего три — **2% 0 0**. Кого смущают проценты, могу написать так: 15px 0 0. Это будет почти то же самое. Средняя цифра в этой записи — ноль — означает что она одинакова как для правой, так и для левой стороны.

Ещё раз:

а) 5px 10px 15px 20px; — сверху будет поле в 5 пикселов, справа 10, снизу 15, слева 20.
б) 5px 10px 15px; — сверху 5, с боков по 10, снизу 15.
в) 5px 10px; — сверху и снизу по 5, с боков по 10.
г) 5px; — со всех сторон по 5.

Пункты б), в) и г) здесь — это просто сокращенная запись. Думаю, принцип понятен.

Остальное несложно: фон белого цвета, цвет шрифта тёмно-серый (#333 или, если угодно #333333), ниже перечислены в порядке предпочтения используемые семейства шрифтов. Тут есть одно замечание: если имя шрифта состоит более чем из одного слова, то его нужно взять целиком в кавычки. Например:

"**Times New Roman**"

3. А вот в следующем правиле уже что-то новенькое — появилось незнакомое слово **container** с решёткой (#). Данная решётка и означает уникальность атрибута. То есть тэг **div** с данным атрибутом будет использован **только один** раз на странице.

Зачем вообще нужен контейнер? А затем, чтобы поместить нашу страничку в центр экрана монитора. Для этого мы указали у контейнера отступы: сверху и снизу ноль, а с боков **auto**. Сие и означает, что при любом размере экрана наш сайт всегда будет строго по центру. Ширина страницы при этом равна 760 пикселям.

Возникает вопрос: а почему бы у тэга **body** не указать такую же ширину страницы и *авто-выравнивание*? Дело в том, что браузеры читают данный тег (body) по-своему и отдают под него ВСЮ видимую область экрана. То

есть, если бы мы даже и захотели задать авто-выравнивание для body, но при этом ширину страницы задали в 760 пикселей, то страничка все равно «лепилась» бы к левому краю экрана. А нам это не нужно. Вот мы и приспособили эдакую “коробочку” под нашу страничку.

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 22 дуйма, а кто-то до сих пор щурится в 14”. 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. Это примерно то же самое, как смотреть широкоэкранный фильм в старом телевизоре — изображение вытягивается по вертикали, делая физиономии продолговатыми и превращая актеров в эдаких Замесов (х/ф «Враг мой»).

Блочная верстка | Урок 4

Продолжаем разговор за CSS. Вообще в *каскадных листах стилей* принято некое упорядоченное расположение правил. Вначале указываются общие правила для всей страницы, а потом идут в той очерёдности, в которой появляются элементы на странице сверху вниз. Мы пока указали не все общие элементы. Нет заголовков, списков, ссылок (тэги h, ul, ol, a). Их мы добавим позже, а пока начнём уже хоть что-нибудь прицеплять весомое к нашей странице.

Из нарисованного в Фотошопе макета я вырезал целиком шапку (её мы используем не как картинку, а как фон), оба рисунка с поросями, рисунки вензеля, зелёной линии внизу, фон полосы навигации (не весь, а только отрезанный кусочек 8x35 пикселей) и зеленой галки у списка новостей.

Итого у нас вышло 7 изображений. Их поместим в той же папке, где у нас лежит сама страничка и лист стилей. Тут необходимо сразу сделать небольшое замечание: мы с вами сейчас верстаем очень простую страницу, можно сказать — весь наш сайт и есть эта одна страница. Поэтому мы не мудрствуя лукаво положили все в одну корзинку, то есть папку. Но когда вы

сами уже станете верстать свои многостраничные сайты, то распределяйте файлики по темам и по назначению. То есть, все картинки в одну отдельную папку Images, все статьи в папку Articles и т.д. Чтобы у вас всегда в папках был порядок. Иначе просто потом заплутаете.

А теперь продолжим.

Далее в листе стилей запишем правило для “шапки” (header):

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

Здесь мы указали, что вся наша шапка залита фоном-картинкой с размерами 760x158 пикселей (такая у меня вышла при разрезании макета). **url(header.jpg)** — это ссылка на картинку. Важный момент(!) — почему ссылка выглядит как простое имя с расширением для картинки? Да потому, что сама картинка лежит в той же самой папке, где и лист стилей. Это так называемая *относительная ссылка*. Если бы была *абсолютной*, то выглядела бы примерно так:

```
url(http://www.мой-сайт.ru/header.jpg)
```

Запись **no-repeat** означает, что картинка не должна повторяться. По-умолчанию браузеры любой фон множат на экране до бесконечности. Если их не урезонить вовремя. Вот этой записью мы как раз и прекратили его вольности. 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**.

А теперь сохраним наш лист стилей в ту же самую папочку, где лежит Главная страница и картинки. Сохраняем таким же образом, как и раньше, только в имени добавляем расширение .css — **style.css**

Теперь мы снова откроем нашу Главную страницу. Между тегами **<body></body>** добавляем следующий код:

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

Сохраняемся. Открываем нашу страничку браузером и любуемся на ровно посерёдке отцентрованную 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 мы открыли тэг контейнера ** ** следом открыли тэг шапки ** **, далее тэг шапки закрывается ** **, и затем так же закрывается тэг контейнера ** **. То есть вложенность тэгов налицо. Понятное дело, что у закрывающих тэгов **div** не нужно снова приписывать атрибут. Просто закрыть и все.

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.

Полюбовались? Теперь продолжим писать код для листа стилей. Открываем его и следом за правилом для шапки запишем правило для блока навигации:

```
#nav {
background: url(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;
}

#nav li a:hover {
color: #f00;
}
```

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

Панель навигации будет у нас одна — сразу под шапкой, горизонтальная (в подвале мы сделаем простое дублирование обычными ссылками). Для её реализации мы воспользуемся таким элементом как маркированный список.

Для тех, кто не в танке объясняю:

маркированный список — это список из нескольких пунктов, записанных в столбик, у которых слева взаместо 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, тэги имеют парные закрывающие тэги, и тэг **li** вложен в тэг **ul**.

Теперь вернёмся к нашему листу стилей. Блок навигации мы обозвали атрибутом **nav**. Вначале укажем общие настройки для него: *бэкграунд* — это картинка с именем **nav-bg.jpg** размером 8x35 пикселей. Это обычный такой “столбик” с градиентом от белого к серому сверху вниз. Чтобы растянуть его по всей полосе навигации, мы указали в значении слово **repeat-x**, что означает “повторить по оси x”, то есть по горизонтали (об этом уже говорилось в чуть выше).

Далее мы указали цвет шрифта ярко-красного цвета **#f00** для активного раздела. Он у нас не будет ссылкой, а останется простым текстом (делать на Главной странице ссылку на самое себя — тавтология). Остальное в этом правиле несложно: размер шрифта, толщина, высота по вертикали (тут появилась новая единица измерения **em**, которая равна высоте прописной буквы выбранного шрифта. Значение **1.8em** показывает, что шрифт увеличен на 1.8 высоты стандартной буквы), ну и выравнивание текста по центру.

Следующее правило указывает, что у нашего списка не должно быть никаких маркеров. Это задаётся значением **none**. И правда, зачем нам в меню лишние точки, кружочки или квадратики?

По-умолчанию строки списков всегда располагаются в столбик. Чтобы этого не происходило, мы далее указываем для строк списка расположение по горизонтали, то есть в линию — **display: inline;**

И еще добавляем отступы: сверху и снизу по нулям, с боков по 8 пикселей.

А теперь укажем, каким образом наши менюшки будут раегировать на наведение мышки. Понятное дело, что менюшки — это ссылки на другие разделы нашего сайта. А каким тэгом означаются ссылки? Правильно, тэгом **a**. Поэтому пропишем ещё пару правил для ссылок.

```
#nav li a {
color: #0c0;
}

#nav li a:hover {
color: #f00;
}
```

В первом мы обозначили цвет ссылки в спокойном состоянии, а во втором — в активном, то есть при наведении мыши.

Ну а теперь следом добавим вот такое правило:

```
a {
text-decoration: none;
}
```

Это общее для всех ссылок правило. Оно указывает, что все ссылки на странице по умолчанию не используют подчёркивание. Ну согласитесь, в меню навигации это не всегда выглядит красиво. А там, где нужно, мы это правило изменим.

Блочная верстка | Урок 5

Итак, продолжим верстать нашу Главную страницу. В этом уроке будет мноооого кода. Что у нас уже есть? Контейнер, в котором лежит шапка. Следом за шапкой добавим блок навигации. Находим в коде следующее место:

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

и сразу же за ним запишем следующее:

```
<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 несуществующих страниц мы просто вставили решётку (#), которая всегда возвращает нас на текущую страницу.

Теперь настала пора сказать пару слов о *каскадности стиля*. В 4-м уроке 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**… Каждое последующее правило получает “в наследство” характеристики предыдущего: от **nav** к **ul**, от **ul** к **li**. Все вместе они являются вложенными в тэг контейнера и получают также от него часть правил (в частности, центрирование посередине экрана и заданную ширину в 760 пикселей). Это и является своеобразным каскадом.

Это было лирическое отступление.

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

Ну а теперь пора уже, наконец, наполнить нашу страницу чем-нибудь полезным, то есть **Контентом** (”…как много в этом слове…”). На макете во втором уроке видно, что полезная площадь страницы разделена на две функциональные области:

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

Обычно, такую вёрстку называют *двухколоночной*. Как сделать так, чтобы текст остался слева, а новости справа я расскажу чуть позднее, когда мы займёмся листом стилей. А пока давайте запишем на страничке сразу после этого места:

```
……
<li><a href="#">Свинки-герои</a></li>
<li><a href="#">Подружиццо</a></li>
</ul>
</div>
```

Следующий код:

```
<div id="text">
<img class="img1" src="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="venzel.gif" alt="" />
<img class="img2" src="pig2.jpg" alt="Пиггсы на проводе" />
</div>
```

Впечатав 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 абзаца всё понятно: открылся один, за ним кусок текста, закрылся; открылся следующий, за ним опять кусок текста, закрылся, и т.д.

А вот у тэгов изображений появилось нечто новое — это слово **class**. Если кто помнит, то оно означает многоразовость использования данного атрибута (в противовес индивидуальному **id**). Кроме того, **class** можно использовать и вовсе без тэга **div**! Что мы и сделали. У тэгов изображения мы просто вписали это слово сразу же за самим тэгом.

Поясним на примере:
Обычно строку для вставки изображения записывают так:

****

Мы же записали это следующим образом:

****

Мы просто добавили к тэгу новый атрибут **class** с именем **img1**, для которого и укажем в листе стилей всё что пожелаем. Это очень удобная запись. Можно, конечно же, влепить сюда и обычный **div**, но это уже будет не просто тавтология, а прям графоманство какое-то.

Код - вещь красивая!

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 блоков страницы, а не для каждой мелочи.

Блочная верстка | Урок 6

Продолжаем верстать самую Правильную веб-страницу! Здесь опять будет много кода. Однако же не утрашимся! Открываем лист стилей. Мы закончили на правиле для всех ссылок страницы. Теперь нам надо красиво оформить основное содержимое страницы. Его мы обзовём атрибутом **text**.

Запишем далее в листе стилей CSS:

```
#text {
width: 545px;
font-size: .8em;
color: #333;
margin: 10px auto;
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 (в данном правиле ноль можно не писать, .8em — обозначает тоже самое). С отступами понятно — верх и низ по 10 пикселей, по бокам на автомате. 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: “обтекание — слева”. Но ведь это сам текст находится слева! А обтекает его всё остальное справа. В этом есть некая путаница. Чтобы не заплутать, просто запомните: **left** — сам объект слева, а течёт всё правее. И наоборот, **right** — объект справа, а течёт всё левее.

Для чего это нужно? Скопировав (а еще лучше — набрав ручками) приведённые выше правила в свой лист стилей и сохранившись, посмотрите, что получилось — текст 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**.

В следующем правиле 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 будто рваный. Поэтому мы выровняли его по обоим краям. Это не выравнивание по центру! Это скорее равномерное распределение слов в строке. Получилось просто хАрАшО!

Далее. Слово **indent** означает не что иное как обычную "*красную строку*". Размер её также указан в полтора размера шрифта.

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

А теперь давайте укажем правила для наших картинок. Запишем в листе стилей:

```
.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 атрибутом **img1**, **img2**, **venzel**, указав в каждом правиле размеры картинок и отступы для них. Вообще отступы и поля проще всего подбирать опытным путём. То есть вначале просто без них кидаем блоки или картинки на страницу, а потом смотрим, куда все это оно сползло и чуток корректируем, добавляя где надо и убавляя, где не надо. Кстати, значения можно указывать и с минусом. Например -10px. И картинка сдвинется в противоположную сторону, хоть даже и за край экрана.

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

И ещё одна вещь. Как видите, данные правила начинаются не с решётки # , а с точки. Это и есть признак того, что правило относится не к **id**, а к **class**.

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

В следующий раз мы добавим на страницу список новых учаснегов и блок новостей.

Блочная верстка | Урок 7

В блоке текста мы разместим список новых участников. Вообще за это отвечает какой-нибудь php-скрипт, но так как мы делаем простую статичную страницу, то оформим эту штуку обычным списком. Только на этот раз не *маркированным*, а *нумерованным*. Задаётся такой список тэгом **ol**.

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

```
……..
<img class="venzel" src="venzel.gif" alt="" />
<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>
</ol>
<ol>
<li><a href="#">Косолапыч</a></li>
<l><a href="#">Тушка</a></li>
<li><a href="#">Свин Полезный</a></li>
<li><a href="#">Сало</a></li>
<li><a href="#">Кнопка</a></li>
<li><a href="#">Васёк</a></li>
</ol>
</div>
<img class="line" src="line.gif" alt="" />
```

Что мы тут видим? Появился новый атрибут **members**. Так мы обозвали наш список. Тэг **h2** — это заголовок второго уровня. Первый уровень, как лехкко догадаться, это чаще всего название сайта или страницы.

Далее у нас идут подряд аж целых два нумерованных списка. Зачем два? На макете мы разместили новых участников в две колонки с вертикальной нумерацией в каждой. Можно было бы, конечно, занести всех в один общий список, сделать ширину одной ячейки ровно вполовину отведённого под список места, и тогда ячейки автоматом переносились бы на новую строчку. Но тогда нумерация выглядела бы так:

1 2
3 4
5 6

и так далее. Согласитесь - коряво как-то.

Поэтому мы не стали мудрить, а сделали просто два списка подряд. Но! Каждому в листе стилей задали обтекание слева. Об этом чуть позднее. Сразу после списков мы положили картинку волнистой линии.

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

```
#members {
width: 300px;
height: 190px;
float: right;
}

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

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

#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 размер. Причём на этот раз не только ширину, но и высоту. Для чего это нужно? Если, например, в нашем списке было бы чуток меньше народу, то картинка волнистой линии “поджимала” бы список снизу, а нам надо, чтобы она была почти под срез картинки с летящим поросенком слева. Поэтому мы задали жёсткий размер как по ширине, так и по высоте. И обтекание справа. Ну, это уже понятно — список должен быть правее указанного рисунка.

Далее задали правило для заголовка второго уровня. Здесь нам уже всё знакомо.

Для самого нумерованного списка с тэгом **ol** мы указали обтекание слева. То есть оба наших списка (помним, что в коде страницы их два подряд) будут идти не в столбик, а один подле другого рядышком оба-два.

Все имена в списке оформлены в виде ссылок (типо 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-то лежит в зоне действия атрибута **text** и поэтому просто перенял от него часть правил.

Последнее правило здесь для рисунка линии. Оно так же, как и предыдущие картинки, оформлено классом.

Сохранились. А теперь смотрим, что получилось.

В следующий раз прицепим к нашему сайту блок новостей и симпатишный подвальчик - он же **footer**.

Блочная верстка | Урок 8

Продолжаем верстать наш Правильный и соответствующий самым строгим рекомендациям консорциума W3C сайт. Наш девиз — Идеальный XHTML и Кристально Чистый CSS! Шутка.

Сегодня мы практически соберём всю Главную страницу целиком. Останется потом только кой-чего подправить и приукрасить.

Открываем нашу страничку index.html. Находим вот это место:

```
…….
<li><a href="#">Васёк</a></li>
</ol>
</div>
<img class="line" src="line.gif" alt="" />
```

и следом сразу вставляем вот этот кусочек:

```
</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 тут видим? Во-первых, закрывающий тэг ** **, который показывает, что область основного текста закончилась. Далее идёт маркированный список новостей с заголовком 3-го уровня (**h3**).

В самом низу мы добавили “*очистку обтекания*” — **clearfloat** (для неё также в листе стилей пропишем свои правила, а именно очистку с обоих сторон). Для чего это? Вообще наша вёрстка является “*плавающей*“. Вон сколько у нас уже обтекаемых элементов. При этом разные браузеры по-разному воспроизводят такую вёрстку (о том, что IE по-своему “видит” поля и отступы уж и не говорю). В Opera и Firefox можно было бы обойтись и без очистки и сразу ниже писать код футера. Но в IE в таком случае появляются странные подёргивания фона у футера при наведении на ссылки мышки. Браузер словно хочет подпрыгнуть и заполнить все пустующие места какие есть. Поэтому мы просто добавим эту строчку и не станем заморачиваться.

А теперь открываем наш лист стилей и 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;
}
```

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 получилось хитро. Если вернуться чуток назад, то можно заметить, что у заголовка 2-го уровня (у списка учаснегов), также задан размер шрифта в 120%.

Вообще 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 списка учаснегов находится в блоке основного текста, для которого мы задали размер всего шрифта в 0.8em.

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, без маркеров, или вообще свой собственноручно нарисованный маркер. Как раз так мы и поступили. Нарисовали симпатишную зелёную галку с размерами 14x17 пикселей и указали в стиле ссылку на неё — **url(marker.jpg)**. Следом приписали слово **inside**. Что это значит? Маркеры по-умолчанию не входят в сам блок текста. Когда мы задаём поля для текста, то маркеры не слушаются и вылезают за края. Это не всегда есть гут. Поэтому мы принудительно указали им быть inside, то есть внутри колонки с текстом.

Сохраняемся и смотрим, что вышло!

Блочная верстка | Урок 9

Это заключительный урок из серии “Блочная верстка веб-сайта”. Нам осталось только оформить “подвал” (footer), да сделать еще пару-тройку замечаний.

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. Очень часто можно наблюдать такую картину: верхнее горизонтальное меню под “шапкой” практически без изменений дублируется в левой колонке. Выглядит это, по меньшей мере, бестолково. Уж лучше продублировать основные разделы сайта в подвале. Пользы будет несомненно больше.

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

Запишем в коде страницы сразу после этого места:

```
……..
…намечаются до самого утра. При наличии на небе луны — будет весело.
</ul>
</div>
<div class="clearfloat"></div>
```

Следующий 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>© PIG.RU, 2007 | All right reserved. | <a href="http://validator.w3.org/check?uri=http://www.dizweb.ru/pig/index.html">XHTML</a> | <a href="http://jigsaw.w3.org/css-validator/validator?uri=http://www.dizweb.ru/pig/style.css">CSS</a> | e-mail: <a href="mailto:piggs@pig.ru">piggs@pig.ru</a></p>
</div>
```

Здесь у нас слово «Главная» не является ссылкой (об этом мы уже говорили), а следом идут обычные ссылки на другие разделы сайта. Всего в футере два абзаца. В первом ссылки на разделы, а во втором как раз и есть всякие копирайты, ссылка на “мыло” и ещё две интересные ссылочки. Вот они уже как раз для форсу бандитского! Это ссылки на **валидаторы** — конторы, которые проверяют вашу страницу на соответствие стандартам. Ссылки устроены так, что нажимая на них, вы сразу даёте команду этим самым валидаторам проверить ваши странички (которые уже лежат в сети конечно же). И тут же получаете ответ. Сие придаёт вашим страничкам солидность и всеобщее наше верстальское одобрение.

Вот практически и вся страница. Проверьте только чтобы в конце кода у вас было два **/div** подряд. Вот так:

```
…….
href="mailto:piggs@pig.ru">piggs@pig.ru</a></p>
</div>
</div>
</body>
</html>
```

Второй **/div** — это закрытие области контейнера.

Теперь допишем в листе стилей оставшийся код:

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

#footer a {
color : #ff0;
}

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

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

```
.clearfloat {
clear : both;
}
```

Здесь мы задали фон серо-зелёного цвета, цвет текста — белый. А вот нижняя строчка в первом правиле (**clear: both;**) означает, что с обоих сторон подвала не должно быть ничего. То есть он у нас занимает всю ширину нижней части контейнера.

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 текста мы задали по центру.

Самое последнее правило как раз служит для блока очистки, заданного атрибутом **clearfloat**, о котором мы говорили, верстая блок новостей.

Сохраняемся, смотрим. Всё вроде бы хорошо, но как-то неуютно (особенно на белом фоне экрана). Добавим-ка мы всей странице тонкую светло-серую рамку! Откроем лист стилей, найдем вот это место:

```
#container {
width : 760px;
margin : 0 auto;
}
```

И добавим следующую строчку:

```
border : 1px solid #999;
```

Должно получиться следующее:

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

Вот, собственно, и всё! Результат можно посмотреть здесь: Рэзультат.

Реальный сайт о парасенгах можно посмотреть здесь — Pigfly.ru. От первоначального макета мало что осталось. Ну, главное идея, а править сайт можно бесконечно. Хотя идея тоже изменилась, и теперь Летающие Парасенги — это сайт О Счастье, Удаче и Богатстве и даже чуточку о Дзэн.

Напоминаю, что наша верстка является “*плавающей*“. Грубо говоря — элементы на странице располагаются как бы в потоке, занимая свободные места. Поэтому мы и ограничили часть пространства контейнером как коробкой, в которую уложили поочередно все элементы. Кроме плавающей, конечно же, есть и жесткая привязка элементов к странице. В самом общем случае для этого каждому элементу строго задается местоположение, например — расстоянием в пикселах от верхней и левой стороны экрана. Но это уже совсем другая история.

Удачи! ☺

Полезняшки

Здесь я решил собрать в кучку некоторые подсказки по синтаксису CSS, которыми сам пользуюсь часто:

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 Вердана, либо Ариал, либо любой другой без засечек. Слово sans-serif как раз и означает, что 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%; или
font-size: 1.2em; или
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 шестнадцатиричным числом вида #000000. Как я уже и говорил ранее, при одинаковых числах в парах можно делать сокращенную запись #000.

Пример: color: #fff; или
color: #f4f5f7;

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. Ну и справа, слева понятно тоже как.

Можно задать 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. Здесь те же правила, что и в случае с отступами и полями, о которых мы говорили в уроке.

5. Свойства списков.

Задается свойство следующей строкой:

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(bullet.gif);

Понятно, что картинка bullet.gif уже должна существовать в папке с вашим сайтом.

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: o 10px 10px 0;
}
```

Это означает, что для любого рисунка с примененным к нему классом pic, обтекающий текст будет располагаться справа, причем сам рисунок будет плотно прилегать кверху с слева к блоку, а справа и снизу у него будут отступы по 10 пикселей.

И еще несколько полезных ссылок:

Чтобы проверить правильно ли вы оформили страницу и лист стилей, можете загрузить их прямо со своего компа на сайт валидаторов и получить развернутое сообщение об ошибках, если таковые имеются.

Для проверки HTML-кода топаем на validator.w3.org

Для проверки CSS идем на jigsaw.w3.org

На этом все! Следите за новостями на сайте Websovet.Com ибо не за горами новая книга о блочной верстке и CSS «Средний уровень» — вариант для продвинутых верстальщиков.

ENJOY!

Приложения

1. HTML-код урока полностью.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">

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

<title>Пиггасы | Главная</title>

</head>
<body>
<div id="container">
<div id="header">
</div>

<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>

<div id="text">
<img class="img1" src="pig1.jpg" alt="Летящий свин" />
<p>Летать всегда! Летать везде! Летать много, очень-очень много и всегда с улыбкой на морде лица &amp;#151; 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="venzel.gif" alt="" />
<img class="img2" src="pig2.jpg" alt="Пиггсы на проводе" />

<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>
</ol>
<ol>
<li><a href="#">Косолапыч</a></li>
```

```
<li><a href="#">Тушка</a></li>
<li><a href="#">Свин Полезный</a></li>
<li><a href="#">Сало</a></li>
<li><a href="#">Кнопка</a></li>
<li><a href="#">Васёк</a></li>
</ol>
</div>
<img class="line" src="line.gif" alt="" />
</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 &amp;#151; будет весело.</li>
</ul>
</div>
<div class="clearfloat"></div>
<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>© PIG.RU, 2007 | All right reserved. | <a href="http://validator.w3.org/check?uri=http://www.dizweb.ru/pig/index.html">XHTML</a> | <a href="http://jigsaw.w3.org/css-validator/validator?uri=http://www.dizweb.ru/pig/style.css">CSS</a> | | e-mail: <a href="mailto:piggs@pig.ru">piggs@pig.ru</a></p>
</div>
</div>
</body>
</html>
```

2. CSS-код урока полностью:

```
/*(C) PIG.RU, 2007  | piggs@pig.ru */

* {
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;
}

#header {
background : url(header.jpg);
width : 760px;
```

```
height : 158px;
}

#nav {
background : url(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;
}

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

a {
text-decoration : none;
}

#text {
width : 545px;
font-size : 0.8em;
color : #333;
margin : 10px auto;
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 : 1px dotted #f36;
}

.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;
}

#members {
width : 300px;
height : 190px;
float : right;
}

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

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

#members li {
margin : 0 5px;
}

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

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

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

#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;
}

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

#footer a {
color : #ff0;
}

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

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

.clearfloat {
clear : both;
}
```