**Блоки в документах**

Михаил Дубаков

Любой документ состоит из блоков. Например, основной текст, заголовки, форма поиска, меню, новости и т.п. Все блоки каким-то образом позиционируются на странице. Обычно для этого используются таблицы. Вопросы позиционирования и создания общей структуры страниц мы с вами начнем рассматривать очень скоро, а пока поговорим об отдельных блоках, которые бывают на страницах.

Оформление блоков вещь интересная, но нас, как верстальщиков, в первую очередь интересует верстка блоков. Конечно, для уникально оформленного блока придется создавать уникальный HTML-код, но все же есть достаточно много универсальных приемов, которые встречаются сплошь и рядом.

**Выделение блоков**

Практически всегда сайт можно разбить на блоки. В сети встречаются оригинальные сайты дизайнеров, на которых сложно найти блоки, потому что даже меню иногда так запрятано, что на его обнаружение уходит 5 минут. Но все стандартные сайты, с которыми вам и предстоит иметь дело, на блоки разбить можно. На рис. 1 показана первая страница сайта компании «Дизайн Артель», которая занимается веб-разработками<. Она приведена в качестве примера, потому что очень четко видны отдельные блоки.

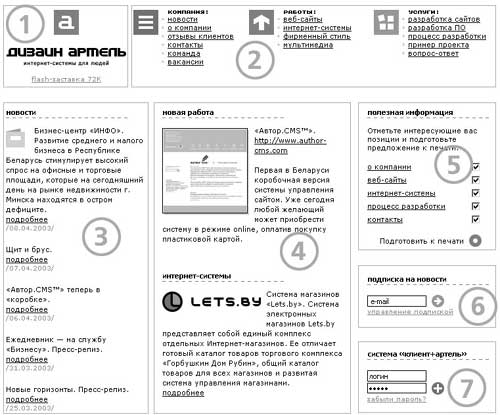


Рис. 1 Главная страница сайта www.artel.by, разбитая на блоки

Я выделил 7 блоков, на рисунке они обозначены цифрами в кружочках. Пройдемся по всем блокам:

Блок с логотипом. Иногда его еще называют блок для бренда. Логотип обычно располагается в левом верхнем углу.

Блок с меню. Меню на сайте нестандартное, расположенное в три столбика.

Блок новостей или лента новостей. Новости выводятся достаточно стандартно в один столбик. Каждая новость имеет дату и ссылку «подробнее».

Два информационных блока. Один с последней работой, а второй с рекламой системы.

Блок для распечатки информации о компании. Обычная форма с чекбоксами и графической кнопкой.

Блок подписки на новости. Поле для ввода электронного адреса и графическая кнопка.

Блок входа для клиентов. Форма с полями для ввода логина и пароля с графической кнопкой.

Дизайн страницы на рис. 1 строится именно на оформлении отдельных блоков и на небольшом количестве ярких картинок. Так бывает нечасто, поэтому обычно блоки на странице несколько сложнее идентифицировать.

Мы сейчас можем как бы структурировано взглянуть на верстку отдельной страницы. Можно выделить две задачи:

создание общей структуры страницы

создание отдельных блоков

Процесс верстки в таком случае можно разбить два этапа. На первом этапе верстальщик создает структуру (каркас). Обычно для этого используются таблицы. После того, как структура страницы создана, можно приступать к верстке отдельных блоков, которые заполняют пустой каркас.

Я, например, перемешиваю два этапа. Я не сразу создаю весь каркас, а только какую-то его часть. Затем имеющуюся часть каркаса я заполняю блоками и перехожу к созданию следующего элемента каркаса. Однако в самом начале я прикидываю общую структуру страницы и методы ее реализации. Если этого не сделать, то можно наткнуться на проблему, для решения которой потребуется полностью переделывать весь каркас.

В качестве небольшой тренировки давайте придумаем каркас для страницы, показанной на рис. 1. Видно, что блоки 1 и 2 можно вынести в отдельную таблицу. Вообще существует общее правило для табличной верстки: использовать как можно больше отдельных таблиц. Если все содержимое страницы поместить в одну таблицу, то посетитель увидит содержание только тогда, когда загрузится все содержимое таблицы. Если же разбить содержимое страницы на несколько разных таблиц, то они будут загружаться последователь и посетитель увидит некоторую часть информации раньше. Так вот, блок 1 и 2 можно вынести в одну таблицу, которая может состоять из четырех столбцов.

Остальные блоки отлично укладываются во вторую таблицу из трех столбцов, они просто сами в нее просятся, так что логично удовлетворить их желание. В итоге у нас получится две отдельных таблицы, одна из которых содержит блоки 1 и 2, а вторая – все остальные блоки.

Как видите, в приведенном примере продумать заранее каркас страницы очень просто. Конечно, подобные ситуации бывают не всегда. Иногда приходится проводить много времени, решая какую-нибудь принципиальную проблему, касающуюся структуры. Воображение дизайнеров (особенно тех, которые с HTML не знакомы) создает порой такие шедевры, которые перевести в HTML невероятно сложно.

Итак, чтобы верстать сайты, необходимо научится делать две вещи: создавать каркас страницы и отдельные блоки. В большинстве случаев создание каркаса задача более сложная, но исключения бывают и нередко. Я решил все же сначала поговорить об отдельных блоках, потому что имеющихся знаний для этого достаточно, а для разговора о структуре страницы – нет.

Так как все же идентифицировать блока на макете, которые радостный дизайнер с неприкрытой гордостью передал вам на верстку? Конечно, для этого надо знать, какие вообще виды блоков бывают. Рассмотрением видов блоков мы сейчас и займемся.

**Формы**

Формы присутствуют на любом корпоративном сайте. Учитывая, что профессиональные верстальщики в основном делают именно корпоративные сайты, нам не избежать знакомства с формами. В основном на сайте присутствуют такие формы: поиск, вход для клиентов, подписка на рассылку, быстрый переход по разделам с помощью селектора. Собственно, на этих примерах мы наглядно рассмотрим, какие бывают типы форм, как они обычно оформляются и, главное, как верстаются. Начнем с самого простого – формы поиска.

**Поиск**

Простейшая форма поиска состоит всего из двух элементов: поля ввода запроса и кнопки запуска поиска. Наиболее привычным является расположение этих элементов на одной горизонтали: так, как показано на рис.2.



Рис. 2 Стандартная форма поиска

Кнопка запуска поиска может быть как текстовой, так и графической. Обычная текстовая кнопка реализуется вот так:

<input type=”submit” value=”искать”>

Графическая кнопка рисуется дизайнером, затем вырезается верстальщиком и сохраняется в каталоге с остальными картинками. Как вы уже знаете, каталог с картинками я всегда называю i. Есть у меня и стандартное название кнопки submit, такую кнопку я обычно называю go. При таких названиях графическая кнопка вставляется вот так:

<input type=”image” src=”i/go.gif” alt=”искать”>

Стандартное решение при верстке блока формы поиска, показанного на рис. 2, сделать таблицу из одной строки и двух ячеек. В первую ячейку вставляется поле ввода текста, а во вторую – кнопка. Код будет таким:

<table cellpadding="0" cellspacing="0">

<form>

  <tr>

    <td><input type="text" name="search">&nbsp;</td>

    <td><input type="submit" value="искать"></td>

  </tr>

</form>

</table>

На что надо обратить внимание? Во-первых, на то, что тег <form> стоит между тегами <table> и <tr>. Ни один валидатор кода не пропустит такую конструкцию и будет вас клятвенно заверять, что ни в коем случае нельзя так надругаться над HTML и вставлять тег <form> в неположенные места. Да, нельзя, но часто необходимо. Дело в том, что браузеры Internet Explorer, Mozilla и Opera имеют совершенно разные точки зрения на то, как надо отображать элемент <form>. Internet Explorer уверен, что надо сделать приличный отступ от элемента <form>, Opera делает отступ поменьше, а Mozilla вообще считает, что никакого отступа делать не надо. Эти несоответствия решаются так, как показано в коде, то есть помещением тега <form> между тегами <table> и <tr>. В этом случае отступы отсутствуют во всех браузерах, что обычно бывает необходимо.

Во-вторых, в таблице обнулены отступы и расстояния между ячейками. Практически всегда дизайнеры привязывают элементы к какой-либо вертикальной или горизонтальной линии, чтобы композиция была лучше. Если не обнулить отступы, то в итоге получим два пикселя несовпадения. Конечно, это не очень-то и бросается в глаза, но зоркий глаз дизайнера сразу заметит несовпадение и попросит исправить. Обязательно попросит, поверьте моему опыту.

В-третьих, при обнулении отступов элементы формы слипаются. Проблему можно решить, поставив после текстового поля неразрывный пробел &nbsp;.

В реальной жизни формы редко оставляют в первозданном виде и все чаще пишут на них стили. Наиболее часто используется стиль, которые вместо псевдотрехмерной рамки вокруг полей создает обычную рамку. Некоторые объявляют такой стиль для всех элементов <input>, но я считаю это плохой практикой, потому что рамки вокруг чекбоксов смотрятся ужасно. Для текстовых полей я создаю отдельный класс, который обычно называю input. Если кнопка отсылки содержимого формы обычная неграфическая, то для нее я создаю класс submit. Например, я хочу сделать рамку шириной 1 пиксель темно-серого цвета, и задать оранжевый фон для кнопки. Стили будут такими:

INPUT.input {border: 1px solid #666}

INPUT.submit {background: #F90; border: 1px solid #666}

Форма поиска изменится и будет выглядеть значительно лучше.



Рис. 3 Форма поиска с рамками

Иногда не этом не останавливаются. Например, высота кнопки и высота поля ввода немного отличается. Это может раздражать эстетов. Проблему легко решить, если задать одинаковую высоту и у поля, и у кнопки. Для задания высоты существует свойство height.

INPUT.input {border: 1px solid #666; height: 21px}

INPUT.submit {background: #F90; border: 1px solid #666; height: 21px}

Кроме того, надпись на кнопке читается неважно. Лучше изменить цвет шрифта на белый и сделать его полужирным.

INPUT.submit {color: #FFF; font: bold 12px Verdana; background: #F90; border: 1px solid #666; height: 21px}

После таких преобразований форма поиска будет выглядеть так, как показано на рис. 4.



Рис. 4 Форма поиска с заданной высотой и измененным шрифтом на кнопке

Конечно, бывают совершенно уникальные случаи оформления форм, но нас они не особо интересуют. Так что с поиском закончим.

**Вход для клиентов**

Еще одной популярной формой является вход для клиентов. На рис. 1 такая форма представлена блоком 7. Обычно форма состоит из двух полей (логина и пароля) и кнопки отправки. Иногда рядом вставляют ссылку на сервис «забыл пароль». Пользователи очень часто забывают свои пароли, так что подобный сервис крайне полезен.

Типичных расположений элементов формы два:

все элементы располагаются в одну строку

поле для ввода логина располагается в первой строке, а поле для ввода пароля и кнопка отправки – во второй строке (именно так сделана форма на рис. 1)

Давайте сначала рассмотрим первый вариант. Для него отлично подойдет таблица из трех ячеек. Будем использовать картинку в качестве кнопки отправки и сразу зададим ширину полей, с помощью атрибута size. Код будет простым:

<table cellpadding="0" cellspacing="0">

<form>

  <tr>

    <td><input type="text" name="login" size="14">&nbsp;</td>

    <td><input type="password" name="pass" size="14">&nbsp;</td>

    <td><input type="image" src="i/go.gif" alt="Войти"></td>

  </tr>

</form>

</table>

В браузере такая форма будет выглядеть так, как показано на рис. 5.



Рис. 5 Простая форма входа для клиентов

Для полного счастья осталось написать стили на поля формы. Я знаю, что высота картинки ровно 19 пикселей, поэтому задам высоту полей тоже 19 пикселей. Стиль получится несложным:

INPUT.input {border: 1px solid #666; height: 19px}

Надо бы еще назвать поля, чтобы пользователь знал, куда что вводить. Иногда название поля пишут прямо в самом поле. Для этого можно использовать атрибут value. Кроме того, нам надо задать класс input для обоих полей, так что HTML-код формы немного модифицируется:

<table cellpadding="0" cellspacing="0">

<form>

  <tr>

    <td><input type="text" name="login" size="14" class="input" value="логин">&nbsp;</td>

    <td><input type="password" name="pass" size="14" class="input" value="пароль">&nbsp;</td>

    <td><input type="image" src="i/go.gif" alt="Войти"></td>

  </tr>

</form>

</table>

А выглядеть форма будет так, как показано на рис. 6



Рис. 6 Вид формы после написания стилей

Некоторые могут возразить, что использовать таблицу для верстки такой простой формы совершенно необязательно. Давайте попробуем сделать как можно проще и посмотрим, что из этого выйдет. Если убрать, таблицу, то код станет очень простым:

<form>

<input type="text" name="login" size="14" class="input" value="логин">&nbsp;

<input type="password" name="pass" size="14" class="input" value="пароль">&nbsp;

<input type="image" src="i/go.gif" alt="Войти">

</form>

Он будет отображаться браузером так, как показано на рис. 7.



Рис. 7 Форма без таблиц

Как видите, кнопка отправки сместилась вверх, и форма выглядит очень неаккуратно. Ни один дизайнер не пропустит такую ошибку и обязательно попросит исправить. Более того, ни один приличный верстальщик не должен совершать такие ошибки.

Проблема возникает из-за того, что нельзя осуществить выравнивание элементов по вертикали, если они не вставлены в ячейки таблицы. Проблема, на самом деле, глобальная, и именно она является причиной использования таблиц даже в таких простых случаях. Что самое противное, даже каскадные таблицы стилей не предоставляют хорошей возможности выравнивания блоков по вертикали! Поэтому все равно придется пользоваться обычными таблицами.

Перейдем к второму типу формы входа для клиентов. Таблиц тоже будет достаточно простой. Первая строка содержит одну объединенную ячейку с полем для логина, а вторая строка содержит две ячейки:

<table cellpadding="0" cellspacing="0">

<form>

  <tr>

    <td colspan="2"><input type="text" name="login" size="14" class="input" value="логин"></td>

  </tr>

  <tr>

    <td><input type="password" name="pass" size="14" class="input" value="пароль">&nbsp;</td>

    <td><input type="image" src="i/go.gif" alt="Войти"></td>

  </tr>

</form>

</table>

Кажется, что все в порядке и должно работать. И оно работает в браузере Internet Explorer, но не работает в Mozilla и Opera. Например, в Mozilla форма будет выглядеть так, как показано на рис. 8



Рис. 8 Вид формы в браузере Mozilla

Как видите, поля слиплись из-за того, что расстояния между ячейками таблицы и отступы в ячейках обнулены. Форма смотрится неважно, поэтому надо как-нибудь раздвинуть поля. Если прописать в таблице атрибут cellspacing=”4”, то это решит проблему, но сдвинет все поля на четыре пикселя вправо. Часто это неприемлемо, потому что нарушается выравнивание элементов дизайна вдоль определенной вертикали, страница теряет целостность и может выглядеть несколько незаконченной. Дизайнеры сразу замечают диссонанс и страшно возмущаются, что совершенно справедливо.

Другой вариант решения – вставить после первого поля распорку, то есть невидимую картинку:

<tr>

  <td colspan="2">

    <input type="text" name="login" size="14" class="input" value="логин">

    <br><img src="i/0.gif" width="1" height="4"></td>

</tr>

Я задал высоту картинки 4 пикселя. Обратите внимание, что перед картинкой обязательно должен стоять перевод строки, то есть тег <br>, иначе никакого эффекта не будет.

Такой вариант, в принципе, жизнеспособен и я им часто пользуюсь, но проблему можно решить и с помощью CSS. Нам надо, чтобы у первой ячейки был отступ в 4 пикселя снизу, а у других ячеек не было. Логично, что надо выделить первую ячейку и написать для нее стиль. Выделить лучше всего с помощью класса. Например, назовем класс expand-form. Для задания отступов блока в CSS существует свойство padding. Кроме того, можно задавать отступы для определенной стороны блока с помощью свойств padding-top, padding-right, padding-bottom, padding-left. В нашем случае нужно воспользоваться свойствомpadding-bottom. Стиль для класса expand-form будет очень простым:

.expand-form {padding-bottom: 4px}

А к ячейке таблицы с полем для логина цепляем название класса:

<td colspan="2" class=”expand-form”>

После таких преобразований поля раздвинутся, и форма будет выглядеть так, как показано на рис. 9



Рис. 9 Форма входа для клиентов после устранения слипания полей

Как видите, даже с такими простыми формами возникают определенные проблемы. Я, например, поначалу тратил очень много времени на поиски решения. Только с опытом приходит четкое знание того, как надо действовать в определенной ситуации.

**Навигация**

Навигация – это важнейший элемент любого сайта. Она должна четко показывать посетителю, где он находится и куда может попасть. Крайне желательно еще и показывать, где он уже побывал. Навигация по сайту состоит из нескольких элементов: самые разнообразные меню, внутренние ссылки, система поиска, карта сайта и прочие вспомогательные элементы. В данном разделе мы рассмотрим, какие бывают элементы навигации и как они верстаются.

**Меню**

Меню присутствует практически на любом сайте. Невозможно себе представить корпоративный сайт без меню. Четкую классификацию видов меню построить достаточно сложно, но перед нами такой задачи не стоит. Если вкратце, то с одной стороны меню бывают вертикальные и горизонтальные, а с другой – текстовые, графические и смешанного типа. Сложно сказать, какие виды меню встречаются чаще, но мы кратко рассмотри все типы.

**Вертикальные**

Для начала рассмотрим вертикальные меню и начнем с текстового. Текстовое вертикальное меню обычно оформляют в виде списка. Пример текстового вертикального меню показан на рис. 10

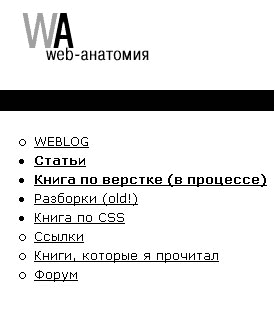


Рис. 10 Текстовое вертикальное меню на моем сайте «Веб-анатомия»

Обратите внимание, что разделы, у которых есть подразделы, имеют черные кружочки, а конечные разделы имеют белые кружочки. Такое отличие помогает посетителям работать с сайтом. Очень быстро они обучаются четко определять, чего ждать от раздела. Давайте с вами воспроизведем примерно такое же меню. Допустим, у нас есть четыре пункта меню: «О компании», «Продукция», «Услуги» и «Контакт». Сразу создаем список:

<ul>

  <li><a href="">О компании</a></li>

  <li><a href="">Продукция</a></li>

  <li><a href="">Услуги</a></li>

  <li><a href="">Контакт</a></li>

</ul>

Обычно на ссылки в меню требуется написать стили, которые отличаются от стилей для обычных ссылок. Например, ссылки могут отличаться цветом, шрифтом или не иметь подчеркивания. Надо бы ссылки в меню как-то обозначить. Для этого можно использовать класс, но лучше всего воспользоваться контекстным селектором. Мы называем как-нибудь блок с меню (то есть задаем блоку с меню ID), например, menu, а затем пишем стиль для всех ссылок, которые находятся внутри блоки с id=”menu”. HTML-код изменится весьма незначительно:

<ul id=”menu”>

Допустим, нам надо, чтобы ссылки в меню были черного цвета. Тогда напишем вот такой стиль:

#menu A {color: #000}

Вот так выглядит полезное использование контекстного селектора. В результате сокращается код, потому что не надо для каждой ссылки добавлять атрибут class=”menu”. На элементы списка в меню зададим шрифт. Например, Arial размером 12 пикселей. Для этого тоже отлично подойдет контекстный селектор:

#menu LI {font: 12px Arial}

Кроме того, бывает полезно слегка увеличить высоту строки в меню:

#menu LI {font: 12px/1.6 Arial}

Наконец, нам надо выделить разделы, у которых нет подразделов. Для таких элементов списка надо задать вывод незакрашенных кружочков. Кстати говоря, кружки, квадратики и прочие символы, которые обозначают элемент списка, называют маркерами. Выделять элементы списка можно с помощью класса, назовем его empty. Для задания вида маркера в CSS есть свойство list-style-type. Используем селектор по классу empty и напишем такой стиль:

LI.empty {list-style-type: circle}

Итоговый HTML-код будет следующим:

<ul id="menu">

  <li class="empty"><a href="">О компании</a></li>

  <li><a href="">Продукция</a></li>

  <li><a href="">Услуги</a></li>

  <li class="empty"><a href="">Контакт</a></li>

</ul>

А само меню будет выглядеть так, как показано на рис. 11

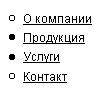


Рис. 11 Вертикальное текстовое меню, реализованное с помощью ненумерованного списка

Некоторые виды вертикальных текстовых меню можно реализовать только с помощью таблицы. Например, на рис. 12 показано меню, пункты которого имеют заданную ширину и разделяются пунктирной линией. Текущий пункт выделен фоном, а ссылки не имеют подчеркивания.

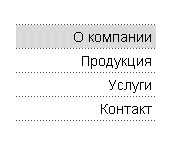


Рис. 12 Вертикальное меню, реализованное с помощью таблицы

Давайте с вами пройдем нелегким путем, чтобы создать точно такое же меню. Путь будет достаточно познавательный. Начнем с того, что сделаем таблицу из четырех строк и одного столбца. В ячейки таблицы вставим пункты меню. Получится вот такой код:

<table>

  <tr><td><a href="">О компании</a></td></tr>

  <tr><td><a href="">Продукция</a></td></tr>

  <tr><td><a href="">Услуги</a></td></tr>

  <tr><td><a href="">Контакт</a></td></tr>

</table>

Конечно, меню пока еще выглядит совсем не так, как надо. Сейчас это просто таблица со ссылками, которые никак не отделены друг от друга и даже на меню не очень похожи. Давайте потихоньку его преображать. Можно начать со ссылок. Надо, чтобы они не имели подчеркивания, были черного цвета и выводились шрифтом Arial размером 14 пикселей. Как и в прошлом примере, мы будем пользоваться контекстным селектором, для чего назовем таблицу menu. Тогда стиль на ссылки будет таким

#menu A {color: #000; font: 14px Arial; text-decoration: none}

Самое сложное - сделать пунктирные разделители между ячейками таблицы. Я для таких целей всегда использую фоновое изображение. Сейчас нам нужен фоновый рисунок шириной 3 пикселя и высотой 1 пиксель. Первый пиксель будет черного цвета, а два других белого. Если повторять такой маленький рисунок по оси Х, то как раз получится пунктирная линия. Думаю, вы легко себе представите, как такой рисунок формирует пунктир. Допустим, мы такое изображение сделали, назвали его bg1.gif и сохранили в каталог i. Теперь надо задать фон для ячеек таблицы. Для адресации к этим ячейкам будем использовать контекстный селектор.

#menu TD {background: url(i/bg1.gif) repeat-x}

То есть все ячейки таблицы сid=”menu” будут иметь фоновый рисунок. Фоновое изображение будет повторяться только по оси Х. Имеет смысл посмотреть на промежуточный результат. Он показан на рис. 13.

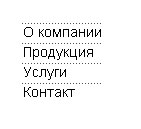


Рис. 13 Промежуточный результат создания вертикального меню

Если сравнить промежуточный результат с рис. 12, то становится ясно, что до него еще далеко. Во-первых, не хватает пунктира после последнего пункта. Это и понятно, потому что ячеек ровно столько, сколько пунктов меню, а пунктирных линий на одну больше. Проблема решается добавлением одно лишней строки с пустой ячейкой в конце таблицы. Кроме того, нет выравнивания по правому краю. Необходимо в стиль для ячеек таблицы добавить выравнивание с помощью свойства text-align.

#menu TD {background: url(i/bg1.gif) repeat-x; text-align: right}

Дальше надо задать ширину меню и увеличить отступы внутри ячейки, чтобы создать необходимое пространство. Дизайнер сделал меню шириной 148 пикселей. Мы зададим ширину таблицы 140 пикселей и отступы 4 пикселям. В сумме получится как раз 148 пикселей. HTML-код немного преобразуется:

<table id="menu" width="140" cellpadding="4" cellspacing="0">

  <tr><td><a href="">О компании</a></td></tr>

  <tr><td><a href="">Продукция</a></td></tr>

  <tr><td><a href="">Услуги</a></td></tr>

  <tr><td><a href="">Контакт</a></td></tr>

  <tr><td></td></tr>

</table>

Осталось сделать совсем чуть-чуть, а именно выделить пункт текущего раздела серым фоном. Для этого можно ввести отдельный класс. Назовем его current. Стиль для класса current будет таким:

#menu TD.current {background: #DDD url(i/bg1.gif) repeat-x}

У вас должен возникнуть вопрос, зачем дублировать фоновый рисунок и в стиле для класса current, если он уже прописан для ячеек таблицы меню? Ответ: если этого не сделать, то фоновый цвет закроет фоновое изображение и пунктира в той ячейке, которая имеет класс current, просто не будет.

Итоговый код вот такой:

#menu TD {background: url(i/bg1.gif) repeat-x; text-align: right}

#menu TD.current {background: #DDD url(i/bg1.gif) repeat-x}

#menu A {color: #000; font: 14px Arial; text-decoration: none}

. . .

<table id="menu" width="140" cellpadding="4" cellspacing="0">

  <tr><td class="current"><a href="">О компании</a></td></tr>

  <tr><td><a href="">Продукция</a></td></tr>

  <tr><td><a href="">Услуги</a></td></tr>

  <tr><td><a href="">Контакт</a></td></tr>

  <tr><td></td></tr>

</table>

Проще всего реализуются графические вертикальные меню. Обычно для них хватает таблицы из одного столбца с обнуленными отступами и расстояниями между ячейками. Приводить реальный пример нет особого смысла.

**Горизонтальные**

Чаще всего используются графические горизонтальные меню. Принципиально есть две их разновидности: резиновые и фиксированные. Резиновые меню растягиваются при увеличении окна браузера, а фиксированные – нет.

Фиксированное меню реализуются очень просто: создается таблица из одного ряда и необходимого количества столбцов, в которые помещаются пункты меню. У таблицы обнуляются отступы и расстояния между ячейками и жестко устанавливается ширина.

Давайте лучше с вами реализуем резиновое меню, которое показано на рис. 14



Рис. 14 Графическое горизонтальное резиновое меню на сайте www.rubi.ru

Особенность меню в том, что текущий пункт помечен галочкой сверху. Для начала нам надо заготовить пять картинок: четыре с пунктами меню и один с галочкой. Затем начнем создавать таблицу. Понятно, что надо сделать два ряда: в первом будет галочка, а во втором пункты меню. Каждый ряд будет состоять из четырех ячеек.

Ширину каждой ячейки надо задавать пропорционально относительно ширины таблицы. Раз у нас ширина таблицы 100% а ячеек четыре, то ширина каждой ячейки будет 25%. Картинку в каждой ячейке надо центрировать.

Код таблицы будет вот такой:

<table width="100%" id="menu" cellpadding="0" cellspacing="0">

<tr>

  <td><img src="i/current.gif" width="32" height="6" border="0" alt=""></td>

  <td></td>

  <td></td>

  <td></td>

</tr>

<tr>

  <td width="25%"><a href=""><img src="i/catalogue.gif" width="40" height="30" border="0" alt=""></a></td>

  <td width="25%"><a href=""><img src="i/forum.gif" width="31" height="30" border="0" alt=""></a></td>

  <td width="25%"><a href=""><img src="i/konkurs.gif" width="49" height="30" border="0" alt=""></a></td>

  <td width="25%"><a href=""><img src="i/reklama.gif" width="40" height="30" border="0" alt=""></a></td>

</tr>

</table>

Для центрирования картинок внутри таблицы можно написать такой стиль:

#menu TD {text-align: center}

Меню будет выглядеть практически правильно, как видно на рис. 15.

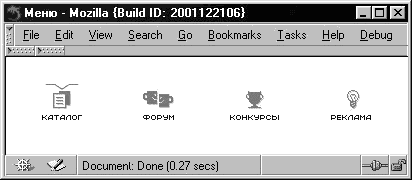


Рис. 15 Вид резинового горизонтального меню с небольшой ошибкой

Ошибка в том, что галочка прилипает к пункту меню. На макете отступ между ними составляет ровно 11 пикселей, так что необходимо его реализовать. Есть два способа: с помощью HTML и с помощью CSS. Для начала попробуем сделать отступ стандартными средствами HTML.

Для этого, конечно же, нужна распорка, то есть невидимый 0.gif. Его надо вставить в ячейку таблицы сразу после галочки и задать ему высоту 11 пикселей:

<td>

  <img src="i/current.gif" width="32" height="6" border="0" alt="">

  <br>

  <img src="i/0.gif" width="1" height="11">

</td>

Второй способ менее надежный, но немного более красивый. Картинку с галочкой можно взять в отдельный блок и задать для блока нижний отступ:

<td>

  <div style="padding-bottom: 11px">

    <img src="i/current.gif" width="32" height="6" border="0" alt="">

  </div>

</td>

На рис. 16 показан окончательный вид меню при разных размерах окна браузера.



Рис. 16 Вид горизонтального резинового меню при разных размерах окна браузера

Как видите, меню ведет себя совершенно правильно, то есть как и ожидалось.

Конечно, существует еще очень много видов меню, которые реализуются совершенно уникальным образом. Но, как вы понимаете, всего многообразия мы не охватим, а посвящать целую главу видам меню довольно скучно. Когда в конце книги мы будем рассматривать процесс верстки реальных сайтов, то еще встретимся с парой-тройкой меню, а пока на этом разговор о меню закончим.

**Дублирующая навигация**

Дублирующая навигация обычно употребляется на больших сайтах. Она располагается внизу страницы и представляет собой дублирование ссылок на самые важные разделы сайта. Типичное дублирующее меню выглядит так, как показано на рис. 17



Рис. 17 Дублирующее меню

Реализуется оно очень просто. Единственная особенность в том, что иногда ссылки в дублирующем меню отличаются от обычных ссылок. Реализовать такое отличие проще всего с помощью контекстного селектора. Например, следующим образом:

#double\_menu A {color: #999; font-size: 11px}

. . .

<div id=”double\_menu”>

  <a href=””>О компании</a> | <a href=””>Партнеры</a> | <a href=””>Решения</a> | <a href=””>Продукты</a> | <a href=””>Цены</a> | <a href=””>Контакты</a>

</div>

Карта сайта

Карта сайта представляет собой развернутое дерево разделов сайта. Обычно оно реализуется с помощью вложенных ненумерованных списков.

<ul>

  <li><a href="">О компании</a></li>

  <li><a href="">Продукция</a>

    <ul>

       <li><a href="">Надувные матрешки</a></li>

       <li><a href="">Электробалалайки</a></li>

    </ul>

  </li>

  <li><a href="">Контакты</a></li>

</ul>

Конечно, если у сайта всего пять разделов, то карта сайта и не нужна вовсе, но в данном случае нет смысла приводить полную карту сайта из сотни разделов. Без стилей такая карта смотрится достаточно тривиально, как видно на рис. 18

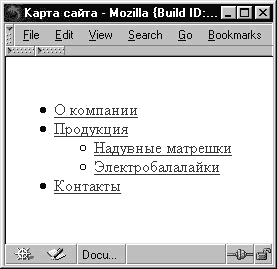


Рис. 18 Карта очень маленького сайта без дополнительного оформления

Давайте сделаем карту более эстетичной. Для начала надо задать шрифт для элементов UL:

UL {font: 11px/1.6 Verdana}

Для того, чтобы различать карту сайта и обычные списки в тексте, надо присвоить ей уникальный идентификатор, то есть id. Например, назовем нашу карту sitemap. Зададим цвет ссылок в карте. Конечно же, будем использовать контекстный селектор:

UL#sitemap A {color: #666}

Теперь можно заняться маркерами. По умолчанию у списка первого уровня маркерами являются закрашенные кружочки, а у списка второго уровня – незакрашенные, что видно на рис. 18. Давайте для элементов списка первого уровня в карте сайта зададим в качестве маркера квадратик, а для элементов списка второго уровня нарисуем свою картинку, например, маленькую стрелочку.

Для задания типа маркера есть свойство list-style-type. Его и применим:

UL#sitemap {color: #666; list-style-type: square}

Для задания изображения в качестве маркера существует свойство list-style-image. Нам надо задать его только для списков, вложенных в список с id=”sitemap”, так что будем использовать контекстный селектор:

UL#sitemap UL {list-style-image: url(i/marker.gif)}

После таких нехитрых стилей карта сайта, показанная на рис. 19, станет значительно более привлекательной.

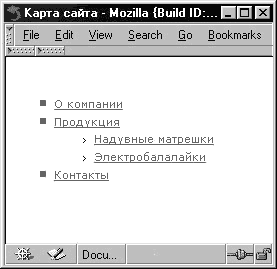


Рис. 19 Карта сайта после написания нескольких стилей

На этом обсуждение навигационных блоков закончим. У нас впереди еще много интересного и полезного, в частности информационные блоки и текстовые элементы.

**Текстовые элементы**

Бывает, что дизайн сайта достаточно неплохой, но текст оформлен совершенно небрежно. Зачастую, оформления текста просто нет, хотя в дизайне сайта он иногда играет не менее важную роль, чем графика.

Печально, что с завидным постоянством оформление текста должны брать на себя верстальщики, потому что многие дизайнеры считают это занятие малозначительным и простым. Как вы понимаете, не все так просто, как кажется многим дизайнерам.

Возникает вопрос, что же делать несчастным верстальщикам? Во-первых, можно тактично намекнуть дизайнеру, что неплохо бы немного поработать над оформлением текста. Если дизайнер высокомерно машет рукой, углубляясь в Adobe Photoshop 7, остается два варианта: оставить все как есть или попробовать оформить текст самостоятельно. Я всегда выбираю второй вариант, но прекрасно понимаю, что далеко не любой верстальщик сможет хорошо оформить текст, поэтому вам советовать не буду. Кроме того, я не буду учить здесь дизайну, потому что вряд ли смогу, но некоторые примеры оформления текста привести полезно, хотя бы с точки зрения их реализации с помощью HTML-кода.

**Простой текст**

Простой текст оформлять легче всего. Максимум, что можно сделать – это задать параметры шрифта и межстрочное расстояние. Мой любимый шрифт для простого текста – это Verdana. И обычно я устанавливают размер шрифта 12 пикселей. Так что в тех таблицах стилей, которые пишу я, очень часто можно встретить вот такую строку:

P,TD,UL,LI {font: 12px Verdana}

Часто слегка увеличенное межстрочное расстояние улучшаем восприятие текста, наполняет его воздухом. Стандартное расстояние составляет 120% от высоты шрифта, но можно увеличивать до 140-160%. Однако для таблиц увеличивать межстрочное расстояние нецелесообразно, так что еще чаще в моих таблицах стилей встречаются вот такие объявления:

P,UL,LI {font: 12px/1.4 Verdana}

TD {font: 12px Verdana}

Любые другие способы оформления простого текста обычно совершенно лишние и вредные.

**Заголовки**

Существует множество способов оформить заголовки: оставить все как есть, изменить шрифт и цвет, использовать разрядку, подчеркивание, цвет фона или же полностью создавать заголовки с помощью картинок (последний способ ужасно негибкий и годится разве что для очень маленьких и редко изменяемых сайтов).

Конечно, надо быть немного дизайнером чтобы прилично оформить заголовки. Но отличный верстальщик всегда немного дизайнер.

Итак, начнем со шрифта. У меня есть любимый шрифт для заголовков – это Arial. Заголовки первого уровня должны быть достаточно большими, так что обычным стилем для заголовка <h1> является такой:

H1 {font: 22px Arial}

Обычно, в документах используется не более трех уровней заголовков. Так что стили надо писать для их всех. Часто стили отличаются только размером шрифта. Типичная тройка стилей для заголовков вот такая:

H1 {font: 22px Arial}

H2 {font: 18px Arial}

H3 {font: bold 14px Arial}

Обратите внимание, что для заголовков <h3> шрифт задан полужирный, потому что иначе они не очень выделяются, что нехорошо.

Иногда очень неплохо смотрится заголовок с подчеркиванием. Подчеркивание представляем собой рамку, которая задается для нижней части блока. Вот так:

H1 {font: 22px Arial; border-bottom: 1px solid #000}

Как выглядит заголовок с подчеркиванием, показано на рис. 20

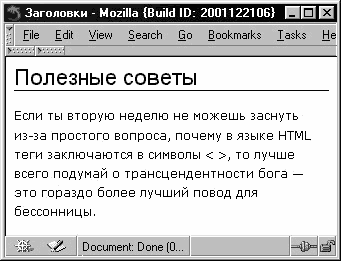


Рис. 20 Заголовок с подчеркиванием

Бывают случаи, когда надо задать фон для заголовка. Но здесь возникает одна сложность, если задать фон для всего блока заголовка, например, вот так:

H1 {font: 22px Arial; border-bottom: 1px solid #000; background: #DDD}

то весь блок и окрасится, включая пустое пространство справа, как показано на рис. 21



Рис. 21 Окрашивание всего блока заголовка

Дело в том, что существует два типа блоков: строчные и структурные. Структурные блоки образуют элементы<p>,<div>,<h1> и т.д. Строчные блоки образуются элементы <b>, <i>, <span> и т.п. Строчные блоки всегда находятся внутри структурных блоков. Главное отличие в том, что структурные блоки имеют вначале и в конце переводы строк и по ширине занимают все предоставляемое им пространство. Ширина строчных блоков строго ограничена содержанием блока. Например, в блоке<b>важно</b> есть пять букв, которые и будут определять ширину блока.

Вернемся к нашему фону. Если требуется, чтобы фон окрашивал весь структурный блок, то все в порядке. Но бывают случаи, когда надо, чтобы фон был только под буквами. Этого можно добиться, если в структурный блок <h1> встроить строчный блок. В HTML имеется специальный строчный блок <span>, который создан как раз для подобных целей. HTML-код будет таким:

<h1><span>Полезные советы</span></h1>

Стили для заголовка будут обычными, но кроме них надо задать фон для элемента <span>, который находится внутри элемента<h1>. Как вы понимаете, для этого будем использовать контекстный селектор. В итоге получится два объявления стилей:

H1 {font: 22px Arial; border-bottom: 1px solid #000}

H1 SPAN {background: #DDD}

А заголовок будет выглядеть так, как показано на рис. 22.

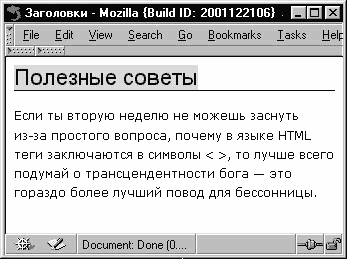


Рис. 22 Оформление заголовка с помощью фона для строчного элемента и рамки

Есть еще пара параметров заголовка, которые часто приходится изменять. Например, бывает необходимость убрать пространство после заголовка, то есть чтобы он вплотную прилегал к тексту.

На первый взгляд, такое поведение реализовать просто, но только на первый. Кстати говоря, это довольно показательный случай работы с каскадными таблицами стилей в той области, когда приходится касаться блоковой модели, совместимости между браузерами и позиционирования. Здесь будут встречаться вещи, с которыми вы пока не знакомы. Кое-что я буду объяснять, но в целом попытки решения проблемы носят чисто показательный характер. Итак, я прошел примерно следующим путем.

Я знаю, что для блоков можно задавать отрицательные поля. В случае с заголовком можно попробовать задать отрицательно нижнее поле и тогда заголовок подтянется к тексту. Методом подбора я установил, что подходящим размером поля будет –20 пикселей. Объявление стиля будет таким:

H1 {font: 22px Arial; margin-bottom: -20px}

Заголовок прекрасно выглядит в браузерах Internet Explorer и Opera, но когда я посмотрел тестовую страничку в браузере Mozilla, то разочарованно поджал губы. На рис. 23 вид в браузере Mozilla показан справа.

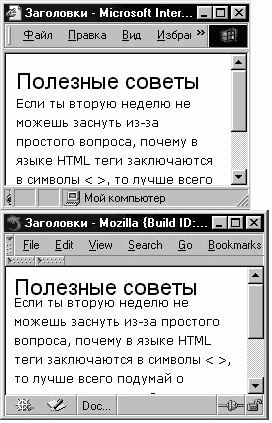


Рис. 23 Несовпадение нижнего поля для заголовка в браузерах Internet Explorer и Mozilla

Лично меня вид заголовка в браузере Mozilla совершенно не устраивает, поэтому начинается поиск обходных путей. Сразу вспоминается такая вещь, как относительное позиционирование. С помощью относительного позиционирования можно сдвигать блоки. Вообще существует два основных вида позиционирования: абсолютное и относительное. Абсолютное полностью вырывает блоки из нормального потока и помещает их ровно туда, куда вы захотите. Нормальный поток – это обычное расположение элементов в HTML, то есть блоки следуют друг за другом. Средствами HTML нельзя изменить нормальный поток, поэтому для верстки приходится пользоваться таблицами.

Абсолютное позиционирование перспективный вид верстки для сайтов с фиксированной шириной, но о нем мы еще будем говорить много и подробно. Относительное позиционирование оставляет блоки в нормальном потоке, но их можно смещать на необходимое расстояние.

Возвращаясь к заголовку, сразу хочется написать вот такой стиль:

H1 {font: 22px Arial; position: relative; bottom: -20px}

Но посмотрев на результат сразу становится понятно, что относительно позиционирование проблемы не решает. Мало того, что расстояние все равно разное, так еще заголовки сместились на 20 пикселей вниз. То есть верхнее поле заголовка как бы увеличилось. Конечно, меня такое поведение не устраивает, так что я пока возвращаю все обратно и думаю дальше.

Дальше обычно мысли развиваются в стандартном направлении. Раз проблема в браузерах, то надо для каждого из них реализовать специфический код. Методом проб устанавливается, что для Mozilla правильным полем будет – 14 пикселей. Остается выяснить, как в одной таблице стилей можно записать два разных объявления для заголовка <h1> таким образом, чтобы одно объявление понимал браузер Internet Explorer, а другое – Mozilla.

Задача, надо сказать, нетривиальная. Я знаю, что есть селекторы, которые Mozilla понимает, а Internet Explorer – нет. Например, селектор наследника, который позволяет применять стили к потомкам элемента. Чтобы лучше понять, как работают селекторы наследника, рассмотрим небольшой пример.

Допустим, у нас есть заголовки, следующие непосредственно после элемента <body>, а есть заголовки, находящиеся внутри самых различных блоков. Заголовки надо выводить шрифтом разного размера. Этого можно добиться с помощью селектора наследника:

BODY > H1 {font: 22px Arial}

DIV > H1 {font: 14px Arial}

Честно говоря, особой пользы от селектора наследника нет, но меня они интересуют по другой причине: их не понимает браузер Internet Explorer, но отлично понимает браузер Mozilla! То есть селектор наследника можно использовать для разделения объявлений.

Вот и воспользуемся им:

H1 {font: 22px Arial; margin-bottom: -20px}

BODY>H1 {margin-bottom: -14px}

После этого нижнее поле заголовка будет совершенно одинаковое в браузерах Internet Explorer и Mozilla, но тут я решаю проверить результат в Opera и вижу, что поле больше, чем необходимо… Получается, что надо разделить объявления стилей для Opera и IE с одной стороны, и Mozilla с другой стороны. Честно признаюсь, я не знаю, как реализовать такое разделение средствами CSS. Конечно, можно написать скрипт, который определяет браузер и динамически подключает в зависимости от браузера разные таблицы стилей, но мне такое решение проблемы на кажется красивым.

Похоже, я зашел в тупик. Что же делать? Первый вариант, не обращать внимания, что в Opera нижнее поле на несколько пикселей больше. На самом деле, в большинстве случаев ничего страшного в этом нет. Конечно, если стиль более сложный и критичный, то так поступать нельзя. В крайнем случае можно все же разделить разные таблицы стилей для разных браузеров с помощью специального скрипта, который будет отличать браузеры и динамически подключать необходимую таблицу стилей.

Вот такие проблемы возникают, когда пытаешься реализовать очень простой, на первый взгляд, стиль для заголовков. Что поделать, такова жизнь…

**Таблицы**

Оформлению таблиц можно посвятить отдельную главу, но мы ограничимся несколькими страницами. Например, у нас есть простая таблица, показанная на рис. 24

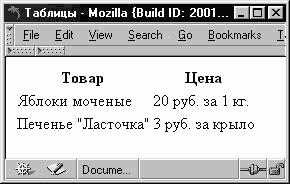


Рис. 24 Очень простая таблица без стилей

Кода пока приводить не буду, потому что он совершенно обычный. Разве что заголовок таблицы сделан с помощью элементов <th>, а не <td>, что совершенно правильно. Сейчас на таблицу не написано ни одного стиля, так что начнем пробовать на ней разные варианты оформления.

Сначала зададим шрифт:

TH, TD {font: 12px Verdana}

Смотрим результат в браузере и обнаруживаем, что заголовки не выводятся полужирным. Стиль по умолчанию мы переписали, так что надо явно задать для заголовков полужирный шрифт. Для этого придется разделить объявления:

TH {font: bold 12px Verdana}

TD {font: 12px Verdana}

В самой таблице слегка увеличим отступы в ячейках, прописав cellpadding=”4”. Сейчас можно браться за оформление. Одно из самых распространенных – задание фона для ячеек. Действительно, обычно получается неплохо. Мы зададим для обычных ячеек светло-серый фон, а для заголовка чуть темнее, чтобы явно его выделить. Как задается фон вы знаете:

TH {background: #DDD; font: bold 12px Verdana}

TD {background: #EEE; font: 12px Verdana}

Результат показан на рис. 25

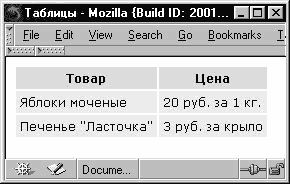


Рис. 25 Таблица с фоном для ячеек

Как видите, действительно неплохо смотрится. Давайте попробуем другие виды оформления. Еще можно отделять строки таблицы линиями.

При таком способе оформления обычно обнуляют расстояние между ячейками, потому что иначе линия будет прерывистой, что некрасиво. При это устанавливают достаточно большие отступы, мы установим cellpadding=”6”. Линия реализуется с помощью рамки. Мы установим рамку для нижней части блока, который представляет собой ячейку. Кроме того, для заголовка зададим фон:

TH {background: #DDD; border-bottom: 1px solid #666; font: bold 12px Verdana}

TD {border-bottom: 1px solid #666; font: 12px Verdana}

Результат будет довольно симпатичный, он показан на рис. 26

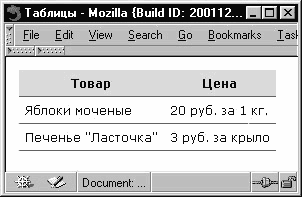


Рис. 26 Таблица с линиями между строками

Фантазия дизайнеров безгранична, поэтому приходится верстать самые разнообразные таблицы с фоновыми изображениями, рисунками в заголовке и т.д. Для нас важен принцип написания стилей для таблиц, а его мы уже освоили.

**Списки**

Про списки мы достаточно подробно говорили, когда рассматривали меню. Особого смысла повторяться нет, но вообще можно немного обсудить разницу в списках между браузерами. Создадим простой список и посмотрим, как он будет выглядеть в различных браузерах.

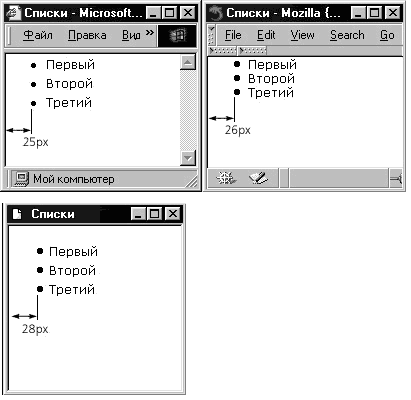


Рис. 27 Вид простого списка в браузерах Internet Explorer, Mozilla и Opera

Как видите различия есть. Во-первых, списки имеют разные поля слева. Конечно, в обычном тексте это не принципиально, но когда необходимы жесткие выравнивания меню, то разница в 3 пикселя может быть заметна. Во-вторых, размер маркеров у списка в Internet Explorer заметно меньше, чем у остальных браузеров. Мне кажется, что в IE размер подобран лучше, и список смотрится аккуратнее. И, наконец, межстрочное расстояние в списках разное. Если в Internet Explorer оно равно 160%, то в Mozilla – 120% (как и элемента <p>). Кажется, что в Opera тоже задано 160%, но на самом деле, как это ни удивительно, в Opera тоже 120%. Кроме того, в Opera есть отступ сверху, тогда как в других браузерах нет.

При написании стилей необходимо учитывать отличия в отображении списков, если это необходимо. Например, если вы хотите, чтобы маркер был одинаковый во всех браузерах, то придется его нарисовать самостоятельно и подключить этот рисунок. К сожалению, размер стандартного маркера невозможно изменить. Если вы назвали файл с маркером circle.gif, то подключить его ко всем ненумерованным спискам очень просто:

UL {list-style-image: url(i/circle.gif)}

Нам осталось рассмотреть гиперссылки.

Гиперссылки

Оформление гиперссылок вещь ответственная. Посетитель должен четко знать, что является ссылкой, а что нет. Вообще оформлением ссылок обязан заниматься дизайнер, потому что отдавать это на откуп верстальщику опасно.

Если ссылка находится в тексте, то подчеркивание крайне желательно, потому что оно однозначно ассоциируется с гиперссылкой. Часто его и оставляют, ограничиваясь только заданием другого цвета. Как вы знаете, по умолчанию ссылки имеют синий цвет. Многие специалисты по юзабилити яростно советуют не изменять прекрасный и привычный синий цвет, но эстетический взгляд дизайнера не в силах смотреть на синие ссылки и всячески старается поменять цвет. В этом вопросе я с дизайнерами солидарен.

Обычно различные эффекты применяют при наведении мыши на ссылку: изменяют цвет и фон, убирают подчеркивание, задают рамки и т.п. Имеет смысл познакомиться с эффектами поближе.

Начнем с того, что в CSS имеется псевдо-класс hover, который позволяет задавать стили для состояния при наведении курсора мыши на элемент. Например, ссылки у нас черного цвета, а при наведении курсора мыши надо изменять цвет на оранжевый. Такое поведение реализуется двумя объявлениями:

A {color: #000}

A:hover {color: #F90}

Псевдо-класс отличается от обычного тем, что его не надо явно указывать в коде документа. Если вы объявляете обычный класс, то в коде обязательно надо его указать, например, <span class=”date”>. Псевдо-класс не надо указывать в коде, а стили, написанные к нему, автоматически применяются ко всем элементам с данным поведением. В нашем случае псевдо-класс hover задан для всех ссылок, так что при наведении курсора мыши ссылки будут менять цвет. Надеюсь, вам все стало понятно, так что перейдем к более высоким материям.

Обычным способом выделения ссылок при наведении является исчезновение подчеркивания. По ходу книги оно уже неоднократно реализовывалось. Кроме того, иногда изменяют фон. Можно и совместить два этих приема:

A {color: #000}

A:hover {background: #DDD; text-decoration: none}

Результат показан на рис. 28

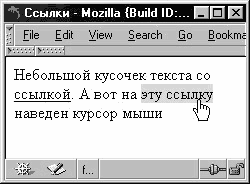


Рис. 28 Изменение фона и устранение подчеркивания ссылки

Еще одни интересный эффект – включение рамок при наведении курсора. Правда, он не работает в браузере Internet Explorer 5, но ничего страшного в этом нет. На первый взгляд, он должен очень просто реализовываться:

A {color: #000}

A:hover {border: 1px solid #000; text-decoration: none}

На самом деле при таком стиле ссылка при наведении будет смещаться вправо на 1 пиксель из-за рамки. Смотрится это некрасиво, а устраняется очень просто:

A:hover {border: 1px solid #000; text-decoration: none; margin-left: -1px}

В результате получается эффект, который показан на рис. 29

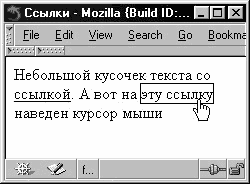


Рис. 29 Рамка вокруг ссылки

Еще можно изменять сплошную линию на пунктир с помощью фонового рисунка. Создаете одно звено пунктирной линии и сохраняете его в файл bg1.gif. После этого останется написать стиль, который дублирует фон по оси Х и позиционирует его вниз блока ссылки:

A:hover {background: url(i/bg1.gif) repeat-x bottom; text-decoration: none}

Результат показан на рис. 30

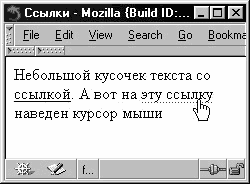


Рис. 30 Замена подчеркивания пунктирной линией

Конечно, оформление ссылок не ограничивается приведенными выше примерами. Вы можете включить фантазию и придумать совершенно необычные и оригинальные эффекты, но осторожно. Есть вещи, которые практически всегда плохо использовать. Например, не рекомендуется изменять размер и насыщенность шрифта. Еще хуже изменять сам шрифт. Например, если у вас в тексте шрифт Verdana, а при наведении курсора мыши он изменяется на Courier, то могут возникать проблемы в браузере и пользователи воспринимают такое изменение с раздражением.

Кроме того, если вы изменяете цвет или фон, то надо внимательно смотреть, насколько контрастны новые цвета. Например, если у вас черная ссылка на белом фоне, а при наведении фон меняется на темно-серый, ссылка читаться не будет.

**Резюме**

Я не затронул еще некоторые виды блоков, такие как локальные баннеры, специальные предложения, новости и т.п. В конце книги будут реальные примеры, где такие блоки обязательно встретятся.

Не стоит увлекаться видами блоков и их классификацией, отдавая все свое свободное время попыткам найти новый способ оформления гиперссылки. Хотя, с другой стороны, занятие это интересное и часто полезное, как полезна любая исследовательская деятельность. Данная глава, скорее, попытка дать общее представление о приемах, которыми вы будете пользоваться при верстке. Каждый приличный сайт уникален и все равно придется искать свои варианты решения. Но с течением времени все реже.