**Позиционирование: всему свое место**

Михаил Дубаков

**Позиционирование с помощью таблиц.**

Как вы знаете, верстка состоит из создания блоков и их позиционирования. Получается, что позиционирование является последней преградой на пути к полному осознанию HTML-верстки. Давайте сделаем решительный шаг.

Как я неоднократно повторял, таблицы отличаются исключительной надежностью и предсказуемостью. На основе таблиц хороший верстальщик может без проверки результата сверстать достаточно сложный макет так, что он будет правильно смотреться во всех популярных браузерах. Таблицы обладают достаточно гибкостью, чтобы позволить сверстать макет практически любой сложности. Откуда у таблиц такие свойства? Все очень просто:

спецификация таблиц в стандарте HTML 4 удачная

популярные браузеры отлично поддерживают реализацию таблиц

поддержка таблиц идентична во всех популярных браузерах

Вот три фактора, которые определяют надежность и предсказуемость таблиц. Что еще надо верстальщикам? Зачем ломать копья, придумывать какие-то новые стандарты, пытаться их воплотить и сделать на их основе нормальный сайт?

Наверное, не от хорошей жизни, как вы думаете? Вот и я думаю, что у таблиц есть определенные недостатки. Недостатки следующие:

нелогичность использования таблиц для создания структуры документа, что неудивительно, ведь таблицы изначально для этого и не предназначались

много лишнего и ненужного кода, что ведет к увеличению веса документа

относительная сложность модификации кода

плохое восприятие кода документа. Это самый противный недостаток. Он особенно ярко проявляется, когда приходится вносить изменения в сайт, который был сверстан год назад.

На самом деле, недостатки не критичные. Действительно, ведь уже несколько лет верстальщики успешно используют таблицы. Но всегда хочется лучшего. Стремление к простоте и логичности неотъемлемое свойство любого нормального верстальщика.

Именно из-за стремления к логичности создали каскадные таблицы стилей. Именно из-за него некоторые уже сейчас пытаются использовать CSS-блоки для верстки сайта. Казалось бы, что простота не такая уж необходимая вещь, чего ради нее так страдать? Возьмем идеальный вариант, когда спецификация CSS очень хорошая и корректно поддерживается всеми браузерами. Что тогда получит верстальщик?

Во-первых, экономия времени. Все будет делаться быстрее, дорабатываться быстрее и переделываться быстрее. Оптимизация процесса всегда крайне полезная вещь. Управляющие проектами будут страшно довольны, а верстальщики выкроят лишний час в день на изучение интересной новой технологии, например.

Во-вторых, чисто эстетическое удовольствие. Когда все получается гораздо проще и красивее, человек чувствует себя лучше. Поднимается настроение и работа кажется не такой уж противной.

В-третьих, документы станут легче, они будут быстрее загружаться и отображаться браузерами. Какой владелец сайта не хочет, чтобы страницы загружались не за 6 секунд, а за 4 секунды?

Для меня важны все пункты, но в особенности второй. Я отношусь к тому сорту людей, для которых простота и понятность — главное. И неважно, касается это дизайна, HTML-кода или PHP-скрипта. Все должно быть как можно понятнее и проще. Наверняка есть люди, которым кажется, что такие вещи совершенно не важны. Они получат минимум преимуществ, используя CSS.

Помечтали и хватит, давайте спустимся с небес на землю. Если использовать CSS по максимуму, то про экономию времени можно забыть. Вам придется писать кросс-браузерный код и проводить много времени, решая проблемы совместимости. Иногда поведение браузера настолько странное, что просто диву даешься. Сложность HTML-кода, конечно, значительно уменьшиться, но зато CSS-код будет очень даже непростым. Единственное, на что можно надеяться, — это уменьшение веса страниц.

Конечно, во многих случаях можно относительно легко написать кросс-браузерную таблицу стилей. Но не всегда. Во всяком случае, бывает сложно сразу найти правильный подход к макету. Иногда кажется, что лучше всего верстать таблицами, а оказывается, что и CSS-блоками было бы отлично. Оценка макета и выбор правильного подхода приходят с опытом. Мы с вами еще будем говорить о том, как смотреть на макет и как выбирать способ верстки.

В четвертой главе был пример, как не надо верстать сайты. В этой главе мы будем всячески истязать все ту же страницу и приводить ее в человеческий вид. Сначала с помощью обычных таблиц, затем с помощью CSS-блоков и CSS-позиционирования. Посмотрим, какой способ верстки для данного примера окажется более правильным. Честно говоря, мне и самому интересно, но почему-то кажется, то документ можно отлично сверстать с помощью CSS-позиционирования.

Для тех, кто забыл как выглядит главная страница сайта компании PopStar, приведен рис. 9.1.



Рис. 9.1 Макет сайта компании PopStar

Я пока не буду приводить код макета, который был сверстан макет в четвертой главе, а сразу начну верстать правильно. Все же мы с вами уже достаточно серьезно продвинулись вперед, и верстать неправильно не хотим. Заодно освоим основные принципы табличной верстки.

Итак, в четвертой главе верстальщик все запихнул (более мягкое слово сложно найти) в одну единственную таблицу. Он нарушил главный принцип табличной верстки.

ПРИНЦИП 1

Каждый горизонтальный блок по возможности надо заключать в отдельную таблицу. Чем больше отдельных таблиц, тем лучше.

Преимущества такие:

Браузер будет показывать таблицы по мере загрузки. Если все поместить в одну таблицу, то браузер покажет странице только после завершения загрузки всего кода. Визуально будет казаться, что страница загружается быстрее.

Каждую таблицу можно более гибко настроить для конкретного блока. Например, первый блок можно сверстать таблицей из трех колонок, а второй — из пяти. Если же использовать одну таблицу, то придется и первый, и второй блок верстать с помощью пяти колонок.

Если посмотреть на рис. 9.1, то сразу бросаются в глаза три больших блока, которые можно заключить в отдельные таблицы:

Первый блок содержит логотип и заголовок «Новости»

Второй блок содержит центральную часть (меню, текст и сами новости)

Третий блок содержит копирайт

Вот мы и определили общую структуру документа. Сейчас самое время заняться каждый блоком в отдельности и подумать, как его лучше сверстать. При этом хорошо бы иметь ввиду второй принцип табличной верстки:

ПРИНЦИП 2

Чем меньше используется столбцов и строк, тем лучше. Не надо усложнять то, что можно сделать просто. Облегчите жизнь себе и браузеру.

Займемся первым блоком. Его можно сверстать с помощью таблицы, которая содержит один ряд и четыре столбца, как показано на рис. 9.2



Рис. 9.2 Первый блок, разбитый на столбцы

Первая ячейка таблицы содержит одну линию, которую можно реализовать отдельной картинкой. Вторая ячейка содержит логотип, так что без картинки здесь никак не обойтись. Третья ячейка опять содержит линию. Возникает подозрение, что в данном случае лучше реализовать линию с помощью фонового рисунка. Отличная идея! На волне фоновой идеи можно еще немного подумать и понять, что первые три ячейки хорошо бы вообще объединить в одну. Действительно, отобьем картинку с логотипом от левого края с помощью невидимой распорки, зададим большую ширину для ячейки, сделаем фон — и все. Последняя четвертая ячейка (которая уже успела стать второй), будет иметь небольшую картинку, чтобы сделать левый край плашки, а затем все продолжится фоновым рисунком.

Вот мы и определились с первым блоком. У нас получится одна большая картинка, и три совсем маленьких и легких. Вспомните, как делал верстальщик в четвертой главе. Он весь первый блок верстал с помощью больших картинок. Конечно, такой подход противоречит третьему принципу табличной верстки:

ПРИНЦИП 3

Необходимо использовать минимум изображений как можно меньшего размера.

В нашем случае меньше изображений использовать практически невозможно.

После того, как первый блок разложен по полочкам, возможны два варианта развития событий:

Продолжать в том же духе с остальными блоками, пока все не станет совершенно ясно

Сверстать первый блок, а затем заняться остальными.

Мне ближе второй подход. Наверное потому, что мне нравится как можно раньше видеть результат. Кроме того, я точно не успею забыть, что успел придумать для первого блока.

Раз уж мне нравится второй подход, его и будет применять. Начнем верстать первый блок, чтобы на практике закрепить теоретические изыскания. Для начала как обычно надо заготовить картинки:

Логотип: logo.gif

Первый фоновый рисунок: bg1.gif

Тоненький рисунок начала плашки, которая над словом «новости» (его нельзя сделать фоном, так что приходится вставлять рисунком): block-left.gif

Второй фоновый рисунок для плашки: bg2.gif

Слово «новости» тоже надо делать рисунком, так задумал дизайнер: news.gif

Все необходимые картинки готовы, можно приступать. Помним, что нужна таблица всего из двух ячеек. В табличной верстке всегда вначале создается таблица с заданными параметрами. В нашем случае суммарная ширина таблицы 750 пикселей. Определяем, что ширина первой ячейки должна быть 517 пикселей, значит ширина второй — 233 пикселя. Расстояния между ячейками и отступы надо обнулить. Создаем такую таблицу:

<table cellspacing="0" cellpadding="0" width="750">

<tr>

<td width="517"></td>

<td width="233"></td>

</tr>

</table>

Начинаем вставлять изображения. Фоном для первой ячейки служит bg1.gif. Сразу же вставляем и логотип.

<td background="i/bg1.gif" width="517">

<img src="i/logo.gif" width="272" height="74" border="0" alt="PopStar">

</td>

Смотрим на результат и видим, что фон с картинкой отлично состыковались, но картинка прибита к левому краю. Помните, я говорил, что ее можно отбить с помощью распорки? В данном случае есть способ и получше. У тега <img> есть атрибут hspace, который задает поля картинки по горизонтали. Дело в том, что поля задаются и справа, и слева, но как раз сейчас это роли не играет. Так что можно отбить логотип вот так:

<img src="i/logo.gif" width="272" height="74" border="0" alt="PopStar" hspace="30">

Перейдем ко второй ячейке. Зададим там фон, и вставим два рисунка. Заметьте, что «новости» располагаются сразу под плашкой, так что между рисунками стоит перевод строки:

<td background="i/bg2.gif" width="233">

<img src="i/block-left.gif" width="1" height="68">

<br><img src="i/news.gif" width="59" height="9" alt="новости">

</td>

Посмотрев на результат и замечаем, что рисунок block-left.gif не стыкуется с фоновым. Немного подумав, вспоминаем, что он должен быть прибит к верхнему краю ячейки, но на самом деле этого нет. Просто у ряда таблицы надо прописать valign="top". В результате имеем код:

<table cellspacing="0" cellpadding="0" width="750">

<tr valign="top">

<td background="i/bg1.gif" width="517">

<img src="i/logo.gif" width="272" height="74" border="0" alt="PopStar" hspace="30">

</td>

<td background="i/bg2.gif" width="233">

<img src="i/block-left.gif" width="1" height="68">

<br><img src="i/news.gif" width="59" height="9" alt="новости">

</td>

</tr>

</table>

Пока получается достаточно компактно и просто, как вы считаете? Обратите внимание на некоторые детали. Я везде указывал размеры картинок. При табличной верстке указывать размеры необходимо, потому что браузер будет быстрее обрабатывать таблицы. Кроме того, я не указал атрибут alt для картинки lock-left.gif. Конечно, такой код противоречит стандарту HTML, но в данном случае картинка не несет никакой смысловой нагрузки. По спецификации в подобных случаях надо задавать пустой атрибут, то есть alt="", потому что голосовые браузеры в этом случае не будут говорить «неизвестное изображение», а тактично промолчат. Учитывая тот факт, что таких браузеров в русскоязычном интернете катастрофически мало, можно их проигнорировать (Я признаю, что моя позиция не совсем корректна. На западе сейчас активно развивается такое движение, как accessibility, то есть общедоступность содержания страниц.<).

Приверженцы доступности сайтов немедленно возмутятся и будут утверждать, что такой подход некрасивый и в корне неверный по отношению к пользователям. Да, я согласен, но компактность кода для меня важнее. Не надо бросать в меня помидоры, сами решайте, как верстать вам.

Давайте займемся вторым блоком. Его можно реализовать с помощью таблицы из пять столбцов. В первой ячейке будет меню, вторая — просто для отбивки, в третьей — текст, четвертая опять же для отбивки, а в пятой – новости. Будем заполнять столбцы таблицы по порядку.

Меню делается очень просто: надо заготовить четыре картинки с пунктами меню, поставить после каждой из них перевод строки и задать в ячейке таблицы выравнивание по правому краю.

<table cellspacing="0" cellpadding="0" width="750">

<tr>

<td align="right">

<a href="about.html"><img src="i/about.gif" width="122" height="20" border="0" alt="О компании"></a><br>

<a href="stars.html"><img src="i/stars.gif" width="122" height="17" border="0" alt="Наши звездочки"></a><br>

<a href="services.html"><img src="i/services.gif" width="122" height="20" border="0" alt="Услуги"></a><br>

<a href="contact.html"><img src="i/contact.gif" width="122" height="15" border="0" alt="Контакты"></a>

</td>

<td></td>

<td></td>

<td></td>

<td></td>

</tr>

</table>

Обратите внимание, мы у картинок прописывали border="0", потому что картинка будет являться ссылкой.

С пустыми столбцами пока ничего делать не будем. Затем вставляем текст и новости. В четвертой главы мы уже этим занимались, так что повторять нет особого смысла. Гораздо интереснее подбирать ширины ячеек. Методом научного тыка (а еще лучше измерением ширины в Photoshop) устанавливаем, что ширина первой ячейки должна быть 125 пикселей, ширины отбивок по 15 пикселей, ширина новостей 233 пикселя, а все остальное достается центральной области.

Кроме того, при просмотре результата в браузере обнаруживается, что я забыл прописать выравнивание по верхнему краю содержимого ячеек. То есть в тег <tr> надо добавить атрибут valign="top".

Остается последний третий блок. Его можно сделать таблицей из двух ячеек, как показано на рис. 9.3



Рис. 9.3 Две ячейки таблицы третьего блока

В первой ячейке будет фоновый рисунок, который будет образовывать плашку, и маленькая картинка с черной полоской для окончания плашка. Вторая ячейка будет содержать фоновый рисунок, который образует линию, а также копирайт. Надпись копирайта можно отбить от левого края распоркой.

Фон для плашки я называю bg3.gif, черную полоску для плашки block-right.gif, а фон для линии bg4.gif. Создаем таблицу:

<table cellspacing="0" cellpadding="0" width="750" border="0">

<tr valign="top">

<td background="i/bg3.gif" width="208" align="right">

<img src="i/block-right.gif" width="1" height="18">

</td>

<td background="i/bg4.gif" width="542">

<img src="i/0.gif" width="309" height="1">

<font face="Verdana" size="1">2003 &copy; PopStar

</td>

</tr>

</table>

Принцип ее создания практически такой же, как и для самого первого блока. Я так же прописываю фон. Есть два отличия: надо в первой ячейке выровнять содержимое по правому краю, чтобы block-right.gif был справа, а не слева. Используется распорка для позиционирования копирайта в необходимое положение.

Собственно, на этом верстка практически закончена. Разве что осталось обнулить поля элемента <body> с помощью атрибутов marginheight, marginwidth, topmargin и leftmargin.

Если сравнить с тем, как был сверстан макет в четвертой главе, то выяснится, что вес самого HTML-документа даже увеличился на 200 байт, но зато на 1,5 килобайта уменьшился вес картинок. То есть в итоге общий вес документа все же уменьшился на 1,3 килобайта. Конечно, цифры смешные, но и документ уж больно простой.

На примере такого макета хорошо виден основной принцип позиционирования с помощью таблиц: для точного позиционирования элемента внутри таблицы используются точные значения ширины ячеек и распорки. Постоянный расчет ширины ячеек и подбор ширины или высоты распорки: неизбежное занятие верстальщика. Например, если надо отбить элемент на 9 пикселей от верхнего края, то необходимо вставить такую распорку:

<img src="i/0.gif" width="1" height="9"><br>

Если надо отбить элемент от левого края на 123 пикселя, то для этого подойдет вот такая распорка:

<img src="i/0.gif" width="123" height="1">

Если дизайн сложный, то на странице может быть несколько десятков распорок. Как вы понимаете, такое обилие картинок ни к чему хорошему не ведет.

В чем же причина такого частого использования таблиц и картинок? На самом деле за ответом далеко ходит не надо, достаточно посмотреть, для каких элементов в HTML можно устанавливать геометрические размеры. Оказывается, что для пяти элементов, связанных с таблицами (<table>, <th>, <td>, <col>, <colgroup>), картинок (кроме того, и для элемента <object>), а также <hr>, <iframe> и <pre>.

Внутренние фреймы и <pre> использовать для точного позиционирования не представляется возможным. В качестве распорки можно попытаться использовать <hr>, правда его нельзя сделать прозрачным, и в некоторым случаях такое поведение будет мешать. Так что распорка невидимой картинкой более универсальна. Адекватной замены таблицам нет в принципе. Вот и получается господство таблиц и картинок. Что поделать, такова жизнь верстальщика.