12 Введение в CSS-верстку

Каскадные таблицы стилей удобны при форматировании текста, навигационных панелей, изображений и других элементов веб-страницы, но их по-настоящему потрясающие способности проявляются, если нужно скомпоновать макет всей веб-страницы. В то время как HTML-разметка обычно отображает контент страницы на экране сверху вниз, размещая блочные элементы друг за другом, каскадные таблицы стилей позволяют создавать расположенные бок о бок колонки и помещать изображения или текст в любой позиции на странице (и даже наслаивать поверх других элементов), поэтому веб-страницы имеют намного более интересный внешний вид.

CSS-верстка — это достаточно обширная тема. В этой части книги вы подробно узнаете о двух наиболее важных техниках CSS-верстки. А в этой главе приведен краткий обзор принципов, на которых построена разметка, и немного полезных инструкций для решения возникающих в процессе работы проблем.

Типы макетов веб-страниц

Быть веб-дизайнером означает иметь дело с неизвестным. Какими браузерами пользуются ваши посетители? На каких устройствах они просматривают ваш сайт: на смартфоне под управлением операционной системы Android или на iPad? Самая большая проблема, с которой сталкиваются разработчики, состоит в создании привлекательных дизайнов, учитывающих различные размеры экрана. Мониторы различаются размерами и разрешением: от маленьких 15-дюймовых с разрешением 640 × 480 пикселов до огромных 30-дюймовых экранов с разрешением примерно 4096 × 2160 пикселов. Не говоря уже о небольших экранах мобильных телефонов и планшетных компьютеров.

Верстка веб-страниц предполагает три основных подхода к решению упомянутой проблемы. Дизайн практически каждой веб-страницы сводится к использованию одного из двух вариантов: *фиксированного* либо *резинового* дизайна. Фиксированные макеты дают вам наибольший контроль над компоновкой элементов страницы, но могут доставить неудобства отдельным посетителям, использующим устройства, разрешение экранов которых не соответствует предусмотренному вами. Пользователям с устройствами с действительно маленькими экранами придется прокручивать страницу вправо, чтобы все увидеть, а те, у кого устрой-

ства оборудованы экранами с большой диагональю, будут видеть пустоту в области экрана, где мог бы отобразиться контент вашей страницы. Кроме того, владельцам смартфонов для получения нужного им контента придется прибегать к сужению и расширению выводимой на экран информации. Резиновые дизайны (страница увеличивается или уменьшается, чтобы соответствовать размеру окна браузера) делает управление дизайном страницы серьезным испытанием, но при этом эффективнее используется окно браузера. Один из мощных инструментов в подходе к построению веб-страниц — *адаптивный веб-дизайн* — старается решить проблему сильно различающихся по размеру экранов, но за счет усложнения процесса верстки.

- **Фиксированный дизайн.** Некоторые дизайнеры предпочитают плотно распределять страницу по ширине, как показано на рис. 12.1, *вверху*. Независимо от ширины окна браузера ширина контента страницы не изменяется. В некоторых случаях страница «цепляется» к левому краю окна браузера или, что бывает чаще, центрируется. С фиксированным дизайном вам не нужно волноваться по поводу того, что случится с вашей страницей на очень большом (или маленьком) экране.

  Многие сайты с фиксированным дизайном поддерживают ширину экрана менее 1000 пикселов, позволяя окну, полосам прокрутки и другим элементам браузера подходить по размерам для экранов с разрешением 1024 × 768. Очень распространенной является ширина 960 пикселов. Это значение используют *большинство* сайтов с фиксированным дизайном, но в последние годы положение дел существенно изменилось благодаря широкому внедрению смартфонов и планшетных компьютеров, экран которых зачастую имеет меньшую ширину (разрешение), чем среднестатистический сайт с фиксированным дизайном. Ввиду вышесказанного сайтов с фиксированным дизайном становится все меньше. Им на смену приходит адаптивный дизайн, о котором мы и поговорим.

ПРИМЕЧАНИЕ

Чтобы увидеть пример сайта с фиксированным дизайном, посетите ресурс nytimes.com (издание New York Times предоставляет различные версии сайта для мобильных устройств, поэтому для этого примера используйте версию, предназначенную для компьютера).

- **Резиновый дизайн.** Иногда легче плыть по течению вместо того, чтобы бороться с ним. Сайт с резиновым дизайном адаптируется так, чтобы соответствовать любой ширине окна браузера, путем задания процентного значения ширины вместо абсолютного в пикселах. Страница становится шире либо у́же, если посетитель изменяет размеры окна браузера (см. рис. 12.1, *посередине*). Хоть этот тип дизайна наилучшим образом использует доступное пространство окна браузера, вам придется приложить больше усилий, чтобы убедиться в том, что страница хорошо выглядит при различных размерах окна браузера. На очень больших мониторах такой тип дизайна может смотреться слишком размашисто, с длинными строками текста, неудобными для чтения. Существует несколько способов реализации резинового дизайна, например обтекаемые (глава 13) и flex-элементы (глава 14).

Рис. 12.1. Справиться с широким спектром размеров окон браузеров и экранных шрифтов можно несколькими способами

ПРИМЕЧАНИЕ

Чтобы увидеть пример сайта с резиновым дизайном, зайдите на сайт maps.google.com.

- **Адаптивный дизайн.** Эта технология, представленная веб-дизайнером Этаном Маркоттом, предлагает другой способ решения проблемы, возникшей из-за большого разнообразия размеров экранов смартфонов, планшетов и компьютеров. Вместо представления единого макета для всех устройств адаптивный веб-дизайн проводит коррекцию ширины для различных окон браузеров путем изменения их представлений. Например, адаптивная веб-страница может ужиматься с широкого, многоколоночного макета для компьютерных мониторов до одноколо-

ночного макета для смартфонов. Таким образом, адаптивный веб-дизайн сильно напоминает резиновые макеты — конструкции, использующие процентные отношения с целью расширения или сужения в ответ на задаваемую ширину окна браузера. Но в новом дизайне технология пошла дальше путем использования более сложного CSS-кода, так называемых *медиазапросов* для передачи различных дизайнов для браузеров устройств с экранами разной диагонали. Это позволяет создавать существенно отличающиеся макеты в зависимости от устройств, на которых просматривается страница.

В практикуме в конце следующей главы вы создадите страницы с фиксированным и резиновым дизайном. Техника адаптивного веб-дизайна будет подробно рассмотрена в главе 13.

ПРИМЕЧАНИЕ

Свойства max-width и min-width предлагают компромисс между фиксированным и резиновым дизайном (см. раздел «Изменение высоты и ширины» главы 7).

Принцип CSS-верстки

Как говорилось в главе 1, на заре развития Всемирной паутины ограничения HTML-кода вынудили дизайнеров придумывать новые способы оформления сайтов. Самым распространенным инструментом был элемент `table`, который, как изначально предполагалось, должен был служить для отображения информации в виде таблицы, составленной из строк и столбцов с данными. Разработчики использовали HTML-таблицы, чтобы создать своего рода основу для организации контента страницы (рис. 12.2). Однако, поскольку элемент `table` не был предназначен для разметки, дизайнерам часто приходилось управлять им непривычными способами (например, помещая таблицу внутри ячейки *другой* таблицы), чтобы получить нужный результат. Этот метод занимал много времени, добавлял лишний HTML-код и усложнял изменение дизайна в дальнейшем. Но это все, что было у дизайнеров до каскадных таблиц стилей.

В заслугу HTML-таблицам можно поставить следующее: они предоставляют хорошо организованную структуру с унифицированными отдельными ячейками. В CSS-дизайне имеются отдельные элементы с контентом, который нужно разместить в одной из областей страницы. Путем группировки контента во многих отдельных контейнерах и позиционирования этих контейнеров можно создать сложные дизайнерские решения, состоящие из колонок и строк. Самый распространенный элемент, который используется для достижения данной цели, — элемент `div`.

Элемент div

Компоновка макетов приводит к расположению контента в различных областях страницы. В каскадных таблицах стилей для организации контента обычно применяется элемент `div`. Как вы узнали из раздела «Верстка HTML-кода вместе с CSS» главы 1, `div` — это HTML-элемент, у которого нет собственных свойств форматирования (помимо того факта, что браузеры рассматривают элемент как блок с разрывом

строки до и после него). Вместо этого он используется для разметки логического группирования элементов, или *создания разделов* на странице.

В элемент `div` обычно заключается фрагмент HTML-кода, в котором все элементы объединены общим смыслом. Такой элемент, как панель навигации (см. рис. 12.2), обычно занимает верхнюю часть страницы, так что есть смысл задавать элемент `div`, оборачивающий и ее. Кроме того, придется определить элемент `div` для всех основных областей вашей страницы, таких как баннер, область основного контента, боковая панель, нижний колонтитул и т. д. Есть возможность обернуть элементом `div` один или более дополнительных разделов. Одна из распространенных методик состоит в оборачивании в контейнер `div` HTML-кода элемента `body`. Тогда вы сможете установить некоторые основные свойства страницы, применяя каскадные таблицы стилей к этой *обертке* (то есть к элементу `div`). Появится возможность установить ширину для всего контента страницы, задать поля слева/справа или расположить весь контент по центру экрана и добавить фоновый цвет или изображение к основной колонке контента.

Как только будут определены все `div`-контейнеры, добавьте к каждому из них либо класс, либо идентификатор, чтобы было удобнее форматировать каждый контейнер по отдельности. Элемент `div` для области баннера страницы может иметь следующий вид: `<div class="banner">`. Можно обойтись идентификаторами, но один и тот же идентификатор используется на странице однократно, поэтому при наличии элемента, появляющегося несколько раз, следует применять класс. Если, к примеру, существует несколько `div`-контейнеров, оборачивающих фотографии и подписи под ними, можно заключить такие элементы в `div`-контейнер и добавить следующий класс: `<div class="photo">`. (Чтобы разобраться, в каких случаях следует применять `div`-контейнер, см. врезку «HTML-элементы div и span» в главе 3.) Кроме того, если дело касается каскадности (см. главу 5), идентификаторы проявляют всю свою мощь и легко замещают другие стили, зачастую заставляя вас создавать длинные селекторы вроде `#home #banner #nav` для того, чтобы заместить ранее созданные селекторы с использованием идентификаторов. Поэтому многие веб-дизайнеры стараются не добавлять идентификаторы, отдавая предпочтение классам.

Как только будут размечены контейнеры `div`, добавьте к каждому из них либо класс, либо идентификатор, и тогда вы сможете создавать стили CSS для позиционирования блоков на странице, используя обтекаемые элементы (см. главу 13).

ВАЖНОЕ ЗАМЕЧАНИЕ

Находим разумный баланс

Хоть контейнеры `div` крайне необходимы при CSS-верстке, не бросайтесь активно применять их на вашей странице. Распространенное заблуждение состоит в том, что вы должны оборачивать в контейнеры `div` *все* элементы на странице. Допустим, что основная навигационная панель на вашем сайте — неупорядоченный список ссылок (как, например, та, что описана в разделе «Создание панелей навигации» главы 9). Поскольку это важный элемент, вы, возможно, пожелаете обернуть его в контейнер `div`:

```
<div id="mainNav"><ul>...</ul></div>.
```

Однако нет никаких причин добавлять контейнер `div`, если элемент `ul` настолько же удобен. Если элемент `ul` содержит ссылки основной панели навигации, вы можете добавить ваш класс к этому элементу: `<ul class = "mainNav">`.

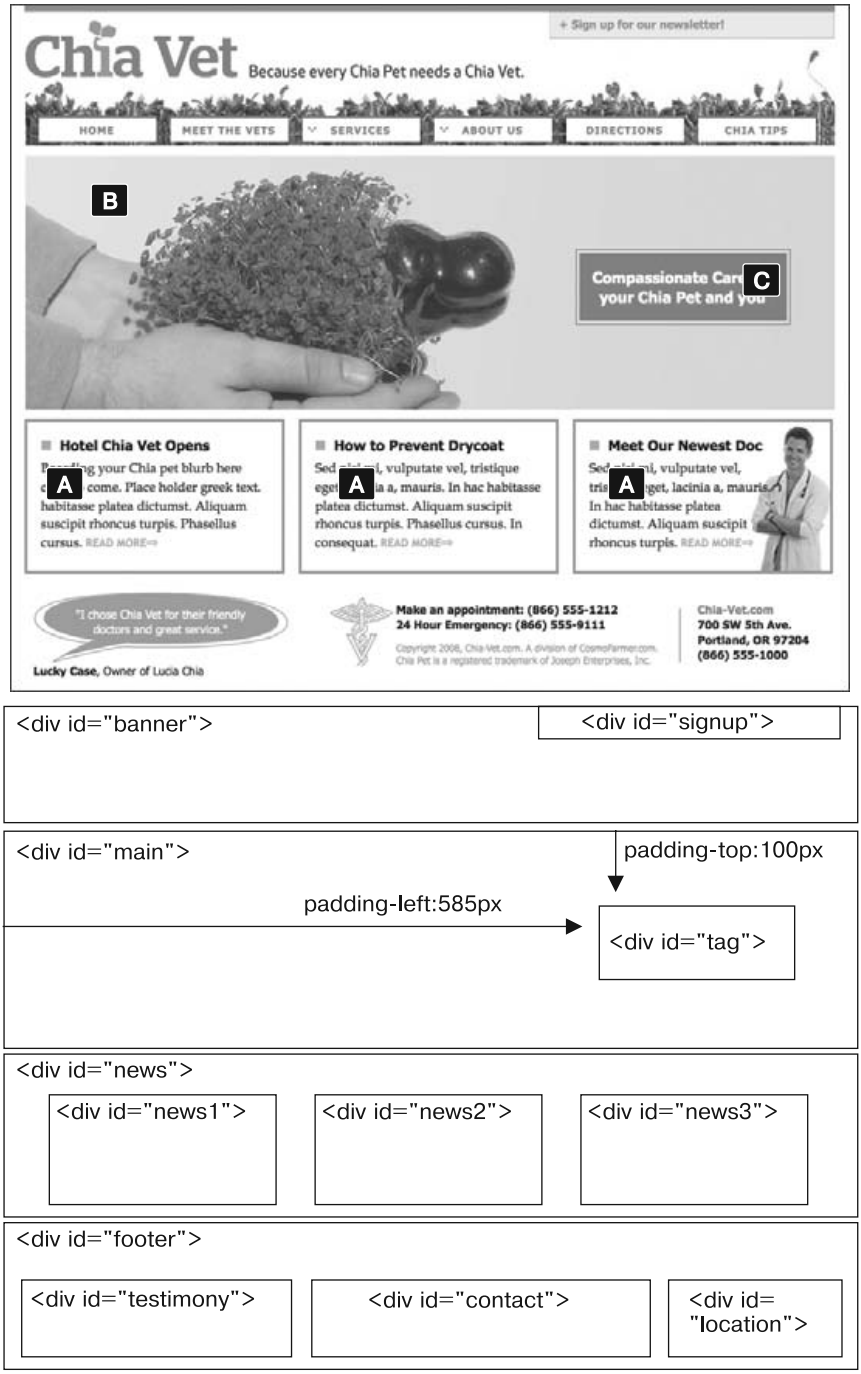

Рис. 12.2. Корректное расположение HTML-элементов, необходимое для преобразования макета Photoshop в код HTML и CSS, включает в себя просмотр структуры страницы и оборачивание связанных групп HTML-элементов в контейнеры div

ВАЖНОЕ ЗАМЕЧАНИЕ

Кроме того, нет необходимости использовать контейнер `div`, если под рукой есть другой, более подходящий HTML-элемент. Например, вы захотели процитировать кусок текста — создать блок, выровненный по правому краю, содержащий важную цитату из контента страницы. В этом случае вместо `div` вы можете использовать элемент `blockquote`, позиционируя его с помощью свойства `float`, которое рассматривалось в разделе «Управление обтеканием контента с помощью плавающих элементов» главы 7.

Но это не значит, что нужно избегать использования элементов `div`. Добавляя их в разумных пределах, вы не увеличите размер файла и не замедлите загрузку страницы. Если контейнер `div` поможет вам решить проблему и ни один из других элементов не подходит, естественно, используйте `div`. Кроме того, контейнер `div` — это единственный способ сгруппировать в единое целое несколько *различных* HTML-элементов. На самом деле не так редко можно увидеть контейнер `div`, обернутый несколькими другими.

Основное правило при работе с HTML — стараться свести количество кода к минимуму, но при этом применять его столько, сколько нужно. Если добавление пары лишних элементов `div` имеет смысл для реализации дизайна, то смело используйте их. Или же попробуйте воспользоваться семантическими элементами, рассматриваемыми в следующем разделе.

Семантические элементы HTML5

До появления HTML5 для организации контента дизайнеры использовали в основном элемент `div`. Но последние версии языка HTML включают другие HTML-элементы, предназначенные для группировки контента конкретного типа. Например, элемент `article` предназначен для содержимого, составляющего одну самодостаточную композицию, например пост блога. Элемент `header` предназначен для представления шапки сайта, навигационных анонсирующих элементов и другого вводного материала страницы или ее раздела. Есть также элемент `footer`, предназначенный для оформления завершающей информации, такой как сведения об авторских правах, имя создателя сайта, дата публикации страницы и т. д. В общем, в HTML5 предоставляются элементы, которые берут на себя работу контейнера `div` для определенных типов контента. (Краткое введение в семантические элементы HTML5 можно найти по адресу tinyurl.com/nsk9768.)

Указанные HTML5-элементы предназначены для *семантического* деления контента веб-страницы. То есть имя элемента, к примеру `header` (в пер. с англ. «шапка» или «верхний колонтитул»), информирует о содержимом этого элемента. Элемент `figure` предназначен для содержания рисунка, как понятно из имени, если перевести его с английского. Иначе говоря, если нужно поместить внутрь элемента содержимое конкретного типа, посмотрите, какой из элементов отвечает этим требованиям. Если содержимое является нижним колонтитулом вашей страницы, то вместо старого доброго контейнера `div` следует применять элемент `footer`. Элемент `div` по-прежнему пригоден для использования. Но теперь он в основном задействуется для группировки контента по соображениям форматирования, например, если нужно добавить границу по краям группы элементов или выровнять эти элементы по левому краю страницы, используя свойство `float`. (Как все это делается, будет рассмотрено в главе 13.)

Все это говорит о том, что, кроме демонстрации превосходства над конкурентами на очередной встрече веб-разработчиков, использование этих семантических

элементов в данный момент не дает никаких особых преимуществ. В будущем программные средства смогут просматривать HTML-код и выделять отдельные публикации на основе элемента `article` или использовать семантические элементы каким-либо другим образом, чтобы улучшить анализ компонентов вашей веб-страницы. Но если вы оттачивали свое мастерство на применении `div`-контейнеров, то пока можете по-прежнему использовать только элементы `div`.

Технологии CSS-верстки

В большинстве веб-страниц для верстки используется свойство `float` (другие варианты CSS-верстки рассмотрены в следующей врезке с информацией для опытных пользователей). Вы уже сталкивались с этим, казалось бы, простым свойством в главе 8, где оно было представлено как способ позиционирования изображения внутри колонки с текстом выравниванием его либо по левому, либо по правому краю. Тот же принцип применяется и к другим элементам: устанавливая для них ширину и выравнивая по левому или по правому краю, вы можете создать колонку (текст, следующий за элементом, обтекает выровненный элемент, как будто тот является еще одной колонкой). Используя свойство `float` с множеством контейнеров `div` или других элементов, вы получаете возможность создавать многоколоночные дизайны. Применяя эту методику дальше, вы будете быстро создавать сложные дизайны, помещая одни обтекаемые элементы `div` внутри других.

Еще одна CSS-технология, *абсолютное позиционирование*, позволяет поместить элемент в любой позиции страницы с точностью до одного пиксела. Вы можете поместить элемент, например, в 100 пикселах от верхнего края окна браузера и в 15 пикселах от левого края. Такие программы предпечатной подготовки, как Adobe InDesign или Apple Pages, работают именно так. К сожалению, изменчивая природа веб-страниц, а также некоторые странные свойства абсолютного позиционирования затрудняют возможность полного контроля над компонентами макета при использовании этой технологии. Как вы узнаете в главе 14, сверстать страницу с помощью абсолютного позиционирования можно, но этот способ лучше подходит для небольших задач вроде позиционирования логотипа в конкретной позиции на странице.

Если пока все эти понятия воспринимаются вами слишком абстрактно, переживать не стоит — работа с использованием всех этих технологий будет рассмотрена в следующих трех главах.

Стратегии верстки

Верстка веб-страницы с помощью каскадных таблиц стилей — это скорее искусство, чем наука. Поэтому нет четкой формулы, которой нужно придерживаться для разметки контента с помощью HTML и форматирования посредством CSS. То, что работает с одним дизайном, может не подойти для другого.

Изучение CSS-верстки происходит на личном опыте. Нужно узнавать, как работают различные свойства каскадных таблиц стилей (особенно абсолютное

позиционирование и обтекаемые элементы), читать о методах верстки, следовать примерам из практикумов, таким как представленные в следующих двух главах, и много-много практиковаться.

Тем не менее определенно есть некоторые стратегии, которые можно принять, приступая к верстке. Они больше похожи на установки или инструкции, а не на жесткие правила. Но, как только вы начнете проектировать дизайн для каких-либо проектов, имейте в виду эти инструкции.

Начиная с контента

Многие дизайнеры хотели бы сразу перейти непосредственно к своей теме — цветам, шрифтам, значкам и изображениям. Но, начиная с визуального дизайна, вы ставите телегу впереди лошади.

Самые важные элементы веб-страницы — это ее контент (заголовки, абзацы текста, фотографии, навигационные ссылки, видео, а также все то, ради чего люди посетят ваш сайт). Посетители захотят выяснить для себя, что ваш сайт может предложить им. Контент стоит во главе угла, и вам сначала нужно подумать, что вы хотите сказать, прежде чем решить, как это должно выглядеть. В конце концов, создавая фантастическую по красоте трехмерную боковую панель, вы ничего не добьетесь, если вам особо нечего отобразить на ней.

Кроме того, идея страницы должна диктовать ее дизайн. Если вы решите, что на домашней странице нужно продавать услуги вашей компании, и захотите выделить выгодное предложение для клиентов, то вполне вероятно, что большое значение будет иметь крупная фотография с приветливыми сотрудниками, как и цитата отзыва одного из довольных клиентов. Поскольку оба этих элемента важны для страницы, вы должны сделать их заметными и неотразимыми.

Стратегия Mobile First

В связи с ростом количества смартфонов и планшетных компьютеров разработчикам пришлось призадуматься насчет сокращения контента до ключевых сообщений и фактов. Это движение под названием *Mobile First* связано с малым размером экрана смартфонов, а также с ограниченным вниманием людей, находящихся в пути. Конструкции Mobile First касаются приоритетного внимания к контенту, а также избавления от его излишнего захламления, включая дополнительную информацию, которая прекрасно помещается на больших экранах компьютерных мониторов, но мешает на экранах значительно меньшего размера и отвлекает от основной информации, которую вы надеялись донести до посетителя.

Учтите, если ваш сайт будет посещать существенное количество людей, использующих смартфоны или планшетные компьютеры с небольшими экранами, не каждый из них захочет прокручивать длинную страницу с контентом (или масштабировать ее, чтобы увидеть все, что доступно на странице). Вместо попытки заполнить каждый оставшийся квадратный сантиметр на 32-дюймовом компьютерном мониторе подумайте об упрощении своего контента, чтобы его усвоение давалось посетителям легко и непосредственно.

ПРИМЕЧАНИЕ

Понятие Mobile First было придумано Люком Врублевски, который написал фантастически короткий трактат на эту тему, доступный по адресу tinyurl.com/kq92pws. Еще одна короткая статья, посвященная стратегии Mobile First, доступна по адресу tinyurl.com/nfew8nz.

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Дополнительные способы верстки с помощью CSS

Сначала для создания многоколоночных макетов веб-дизайнеры использовали таблицы. Затем появилось свойство `float`, благодаря чему был предоставлен другой, менее объемный по коду метод верстки веб-страницы. И хотя большинство разработчиков все еще используют свойство `float`, существуют и другие свойства: часть из них можно применять уже сегодня, а некоторые станут доступны в будущем.

Рабочая группа Консорциума W3C настойчиво ведет к финальным версиям некоторые другие технологии CSS-верстки. Например, модуль *многоколоночной* верстки предоставляет способ получения длинной колонки текста и ее отображения в нескольких смежных колонках. Разговор о нем пойдет в практикуме следующей главы.

Модуль *flex-верстки* предоставляет еще один способ выстраивания блоков содержимого — вертикально, горизонтально, а также в различных направлениях и порядках следования. Обратите внимание: этот модуль имеет хорошую поддержку во многих современных браузерах. Он будет рассмотрен в главе 15.

И наконец, *модульные сетки* являются, наверное, наиболее амбициозной попыткой изменить способ компоновки веб-страниц. Они позволяют разбить страницу на набор строк и столбцов, затем аккуратно прикрепить элементы страницы к этой сетке. Он предназначен для упрощения создания веб-приложений и больше похож на создание компьютерных *программ*. Макеты с модульной системой верстки довольно сложны и (на момент написания данной книги) не поддерживаются большинством браузеров. Как ни странно, Internet Explorer — первый браузер, который начал поддерживать концепцию модульных сеток. Дополнительные сведения о методе можно получить по адресу tinyurl.com/nvqbktp.

Первые наброски

После решения вопроса с контентом можно подумать о его визуальной организации. Некоторые разработчики начинают с HTML-кода: создания `div`-контейнеров, добавления элементов `header`, `article`, `footer` и др. Это вполне разумный подход, поскольку он предоставляет отправную точку по созданию вашего сайта путем перехода непосредственно к HTML-коду.

Но перед переходом к HTML-коду нужно хотя бы набросать план контента веб-страницы. Здесь не требуется ничего особого, сгодятся бумага и карандаш. Поскольку контент планируется поместить в блоки (`div` и другие HTML-элементы) и расположить их на странице, простой набросок группы блоков с колонками и т. д. — быстрый и простой способ проработки макетов различных страниц. Можно быстро понять, куда должен помещаться контент, насколько объемным он будет и каким нужно сделать общий цветовой тон (к примеру, светлым или темным).

СОВЕТ

Ресурс Yahoo! предлагает бесплатный инструментарий Stencil Kit (tinyurl.com/pl7rucb), который может применяться в Illustrator, Visio, OmniGraffle и других графических редакторах для создания макетов веб-страниц. Предоставляемые элементы пользовательского интерфейса, такие как

кнопки, элементы веб-форм, окна и инструменты навигации, позволят составить набросок макета страницы путем простого перетаскивания значков.

При наличии опыта работы с графическими редакторами наподобие Photoshop, Illustrator или Fireworks вы можете воспользоваться ими для создания визуального дизайна. Если вы не сильны в работе с такими программами, то простой рисунок блоков для обозначения различных позиций расположения элементов страницы поможет воплотить замысел о макете страницы. Намного проще будет изменить двухколоночный дизайн на четырехколоночный изменением размеров блоков в программе Illustrator, чем переписыванием кода HTML и CSS.

Но при использовании графических редакторов не стоит тратить слишком много времени на улучшение визуального дизайна. Не тратьте слишком много усилий для воссоздания в Photoshop или Illustrator форматирования, получаемого при использовании свойств каскадных таблиц стилей; для его переделки нужно будет воспользоваться CS-кодом. То есть утвердите предварительный вид своего сайта на бумаге, а затем переходите к редактору HTML-кода, в который поместите контент и используйте свойства каскадных таблиц стилей, изученные с помощью этой книги, для тестирования форматирования.

Определение блоков

После того как вы создали визуальный макет, нужно подумать о том, как создать разметку HTML и CSS для достижения дизайнерских задумок. Этот процесс обычно включает представление различных структурных блоков страницы и идентификацию элементов, которые выглядят как отдельные разделы. Например, на рис. 12.2 присутствует немало элементов, которые выглядят как небольшие разделы: наиболее крупным является раздел с тремя объявлениями внизу (отмечены на рис. 12.2 буквой A). Каждый раздел, как правило, является кандидатом на оборачивание в отдельный элемент `div` (при условии, что нет подходящего HTML-элемента, что мы обсуждали ранее).

Часто визуальная подсказка в макете может помочь решить, нужен ли вообще элемент `div`. Например, линия границы по краям заголовка и нескольких абзацев, означает, что вам понадобится обернуть эту группу HTML-элементов контейнером `div`, к которому применена граница.

Кроме того, если вы видите фрагменты текста, расположенные друг рядом с другом (например, три фрагмента с контентом в нижней части рис. 12.2), вы знаете, что нужно каждый из них заключить в отдельный элемент `div`. HTML-элементы, как правило, сами по себе не располагаются друг рядом с другом, поэтому вам придется использовать некоторые техники верстки (например, обтекаемые элементы, описанные в следующей главе).

Желательно также группировать элементы `div` (и другие элементы), расположенные рядом, в один общий контейнер `div`. Например, в нижней части рис. 12.2 вы видите набор контейнеров `div>`, которые обеспечивают структуру страницы. Элементы `news` и `footer` являются контейнерами для своих собственных наборов `div`-контейнеров. Хоть это и не всегда необходимо, такой подход помогает обеспечивать гибкость. Например, вы можете уменьшить область основного контента

(изображение рук и рекламный слоган) по ширине и переместить раздел с новостями по его правую сторону, который бы сформировал отдельную колонку. Новости могли бы идти друг за другом, а не рядом друг с другом.

Следуя потоку

Элементы обычно не располагаются рядом и не наслаиваются друг на друга. Как правило, они действуют подобно тексту в текстовых редакторах: заполняют всю ширину страницы и следуют от верхнего до нижнего края. Каждый блочный элемент — заголовок, абзац, маркированный список и т. д. — располагается выше следующего блочного элемента. Поскольку это стандартный подход, вам обычно не нужно ничего делать, если вы планировали располагать элементы `div` один за другим.

Например, на рис. 12.2 четыре элемента — `header`, `div`-контейнер `main`, `section` и `footer` — охватывают всю ширину своего контейнера (элемента `body`) и расположены друг за другом по вертикали. Это типичная ситуация для блочных элементов, и вам не нужно делать ничего особенного, применяя каскадные таблицы стилей, чтобы расположить эти четыре элемента `div` таким образом.

Не забывая о фоновых изображениях

Вы, несомненно, видели мозаичные изображения, заполняющие фон веб-страницы, или градиенты, добавляющие эффект глубины баннеру. Свойство `background-image` предоставляет еще один способ добавления фотографий на страницу, не прибегая к помощи элемента `img`. Вставка изображения в качестве фона существующего элемента не только позволяет сохранить пару байтов данных, которые были бы потрачены на элемент `img`, но и упрощает решение некоторых проблем, связанных с компоновкой.

Например, на рис. 12.2 изображение рук по центру (B) на самом деле является обычным фоновым изображением. Это облегчает размещение другого элемента `div` со слоганом Compassionate care… (C), поскольку он расположен поверх фона родительского элемента `div`. Кроме того, фотография врача чуть ниже, в правой части страницы, — обычное фоновое изображение, размещенное в текущем элементе `div`. Добавление небольшого отступа справа сместит текст в этом разделе от фотографии.

ПРИМЕЧАНИЕ

Существуют недостатки использования фотографий в качестве фона контейнеров div (или других HTML-элементов). Во-первых, браузеры обычно не печатают фон, поэтому, если на странице должна быть изображена карта, содержащая какие-либо важные маршруты, вставьте ее с помощью элемента img, а не в качестве фонового изображения. Кроме того, поисковые системы игнорируют каскадные таблицы стилей, поэтому, если вы думаете, что изображение может привлечь дополнительный трафик на ваш сайт, используйте элемент img и добавьте описательный атрибут alt.

Кусочки пазла

Этот совет можно было подавать в разделе «креативного решения проблем». Часто то, что выглядит как одно целое, на самом деле состоит из нескольких частей. Вы

можете увидеть простой пример вышесказанного на рис. 12.2. Здесь есть четыре расположенных друг за другом элемента `div`, каждый из которых имеет белый фон.

Если у вас возникли проблемы с размещением больших элементов на странице (очень большого рисунка, занимающего несколько колонок, или сплошного цвета, заливающего не одну область страницы), подумайте о том, можно ли добиться желаемого внешнего вида страницы, разбив большой элемент на мелкие куски, которые потом сложатся, как кусочки пазла.

Элементы макета

Если вы работали в программе Photoshop, Illustrator или Fireworks, то, вероятно, понимаете концепцию слоев. Слои позволяют создавать отдельные холсты, которые накладываются друг на друга и образуют единое изображение. В этих программах можно легко поместить логотип поверх заголовка с текстом или одну фотографию поверх другой. Если вы хотите сделать то же самое на веб-странице, у вас есть несколько вариантов.

Часто самый простой способ наложить слой поверх фотографии заключается в добавлении изображения в качестве фона другого элемента. Поскольку фоновое изображение следует за элементом, все, что находится внутри этого элемента, — текст, другие изображения — будет расположено поверх фотографии.

Но что делать, если вы хотите наложить фотографию поверх того или иного текста? В таком случае нужно обратиться к единственному свойству, позволяющему наслаивать элементы, — `position`. Мы рассмотрим его в главе 15, поскольку размещение элементов поверх чего-либо требует абсолютного позиционирования.

Не забывая о полях и отступах

Иногда самое простое решение оказывается лучшим. Вам не всегда нужен причудливый CSS-код, чтобы поместить элемент в нужное место на странице. Вспомните, что отступы и поля представляют собой обычное пустое пространство и, используя их, вы можете перемещать элементы по странице. Например, рекламный слоган (см. рис. 12.2, C) размещается простой установкой верхнего и левого отступа для его родительского элемента. Как вы можете видеть на схеме в нижней части на рис. 12.2, слоган помещен внутри другого элемента `div` (`<div class="main">`). Этот элемент на самом деле не имеет другого контента, кроме слогана, — фотография является фоновым изображением, так что добавление отступа перемещает элемент `div` слогана вправо и вниз.