Вадим Дунаев

bhv

HTML, скрипты и стили

4-е издание

- HTML, XHTML
- CSS
- JavaScript, PHP
- SVG

Наиболее полное руководство

В ПОДЛИННИКЕ®

УДК 004.43+004.738.5
ББК 32.973.26-018.1
Д83

Дунаев В. В.

Д83 HTML, скрипты и стили. — 4-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2015. — 816 с.: ил. — (В подлиннике)

ISBN 978-5-9775-3317-1

Рассмотрены средства создания Web-сайтов — языки разметки гипертекста (XHTML, HTML 4 и HTML 5), каскадные таблицы стилей (CSS 2 и CSS 3), а также языки сценариев JavaScript и PHP. Изложены краткие теоретические сведения и приведены многочисленные примеры решения типичных задач разработки сайтов. Четвертое издание книги является результатом существенной переработки третьего издания с учетом современного состояния HTML, CSS и ведущих браузеров, удалены некоторые разделы, ставшие неактуальными, а также исправлены замеченные ошибки.

Для Web-дизайнеров

УДК 004.43+004.738.5
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Екатерина Капалыгина*
Редактор	*Леонид Кочин*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Марины Дамбиевой*

Подписано в печать 31.07.14.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 65,79.
Тираж 1300 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-5-9775-3317-1

Оглавление

ГЛАВА 1

Что такое HTML и таблицы стилей CSS

В данной главе мы рассмотрим язык разметки и таблицы стилей. Здесь важно, не вникая в тонкости синтаксиса, понять их назначение, взаимосвязь и основные направления применения. Далее мы все проясним и поставим на свои места.

1.1. Языки разметки документа

Допустим, вам требуется создать документ (письмо другу, резюме для работодателя, объявление о продаже какого-то имущества, прайс-лист, литературное произведение и т. п.), который вы намерены опубликовать — распечатать на принтере, а затем отправить в издательство или разместить на Web-сайте. Ваш документ будет содержать обычный текст и, возможно, картинки. Вот и все, что вы можете позволить себе при публикации документа на бумажных носителях. Разумеется, все то же самое можно опубликовать и в Интернете, стремясь максимально расширить аудиторию читателей. Однако даже в этом простейшем случае нередко возникают порой нетривиальные задачи форматирования (другими словами, оформления) документа. Текст должен быть структурирован, т. е. разбит на части (разделы, подразделы, абзацы), иногда снабженные заголовками. Даже обычное письмо объемом всего в одну страницу, как правило, разбивают на несколько абзацев, не говоря уже о статьях и более крупных произведениях, таких как повести и романы. Заголовки и абзацы выделяются, по крайней мере, своим местоположением относительно друг друга. Кроме того, заголовки разделов и даже некоторые словосочетания обычно выделяют из общего текстового потока шрифтом — гарнитурой, размером, цветом и другими характеристиками. Если в тексте предусмотрены графические иллюстрации, то задача форматирования документа еще более усложняется: необходимо определить, каким образом текст будет "обтекать" вставленные картинки.

При публикации документа в Интернете вы можете сделать значительно больше, чем при печати: вставить в документ аудио- и видеоклипы, формы, заполняемые читателем и отправляемые на сервер, элементы интерфейса (ссылки на другие документы, кнопки, поля ввода, переключатели и т. п.), с помощью которых можно управлять отображением содержимого и решать другие задачи.

Даже в обычных бумажных документах нередко встречаются ссылки на разделы этого же или другого документа (например, "см. разд. 2.5", "в книге [7] можно найти...",

"в приложении 1 приведен список ..." и т. п.). Встретив такую ссылку в книге, читатель при желании должен либо перелистывать ее страницы, либо искать другую книгу, чтобы затем в поисках нужной информации "порыться" в ней. В Web-документах, просматриваемых в браузерах, щелчок левой кнопкой мыши на ссылке приводит к автоматическому поиску в Сети и отображению в окне браузера соответствующего (релевантного) документа. Такие ссылки в связи с их мощной и удобной функциональностью называют гиперссылками, а тексты, их содержащие, — гипертекстами (HyperText). Гиперссылки могут указывать на документы различного характера — обычные и гипертексты, графические изображения, звуковые и видеоклипы и др. Активизация гиперссылки означает запрос к Web-серверу указанного в ней документа и, если он найден в Сети, пересылку и отображение его в Web-браузере. Благодаря гиперссылкам множество различных документов можно связать в некую единую систему, образовав сайт или, другими словами, Web-узел. Так вы можете создать, по крайней мере, один документ, содержащий гиперссылки на другие, чтобы не только представить самого себя в Интернете, но и послужить проводником к другим ресурсам Всемирной сети. Простая в своей основе идея и адекватный ей инструмент гиперссылки позволил создать и далее развивать технологии, лежащие в фундаменте Всемирной паутины (World Wide Web), преобразившей в последнее время все информационное пространство и коммуникации в нем.

Итак, при подготовке любого документа к публикации его требуется отформатировать или, иначе говоря, разметить — с помощью специальных символов явно указать, что и каким образом следует представить пользователю (читателю) этого документа. Набор специальных символов (меток, дескрипторов, команд или тегов), а также правила их употребления составляют то, что называют языком разметки (Markup Language). В настоящее время существуют несколько таких языков, среди которых наиболее популярны HTML (HyperText Markup Language — язык разметки гипертекста), XHTML (eXtensible HyperText Markup Language — расширяемый язык разметки гипертекста) и XML (eXtensible Markup Language — расширяемый язык разметки). HTML и XHTML применяются для разметки Web-документов, а XML — как средство структуризации информации для обмена между компьютерными программами. Между этими языками очень много общего, хотя имеются и различия, о которых будет рассказано позже.

Разметка документа, как уже упоминалось, осуществляется посредством специальных дескрипторов, называемых тегами (tag — метка, признак, ярлык). В языках разметки в большинстве случаев они имеют вид: *<имя_тега>* и *</имя_тега>*. Например, `<p>` и `</p>`, где `p` — имя тега. Эти две формы одного и того же тега играют роль подобно открывающей и закрывающей скобкам при написании математических формул. Между ними можно разместить некоторое содержимое — обычный текст и/или тег, содержащий, возможно, другие теги. Тег, имеющий открывающую и замыкающую части, называют контейнерным. Вот его синтаксис:

< имя_тега > содержимое контейнерного тега </ имя_тега >

Существуют и неконтейнерные теги (без замыкающей части), например `<img>` — для вставки графического изображения и `<br>` — для указания перехода на новую строку.

Теги не отображаются в окне браузера, а лишь указывают ему, какие элементы входят в документ и как они между собой соотносятся. Так, текстовый документ состоит из абзацев, которые могут объединяться в разделы. Разделы, в свою очередь, могут входить

в более крупные структурные единицы, например главы или части и т. д. Язык HTML обеспечивает поддержку иерархической декомпозиции документа довольно простыми средствами. Например, чтобы указать, что данный фрагмент текста относится к одному и тому же абзацу, достаточно перед этим фрагментом написать тег , а после него — . Другими словами, текст абзаца следует заключить в контейнер , например

```
<p>Здравствуйте, дорогой Иван Иванович.</p>.
```

Чтобы объединить в один раздел несколько элементов документа (например, абзацев), их следует заключить в какой-нибудь подходящий для этой цели контейнер, например (листинг 1.1).

**Листинг 1.1. Пример использования контейнера **

```
<div>
    <p>
      Здравствуйте, дорогой Иван Иванович.
    </p>
    <p>
      В первых строках своего письма передаю поклон Марье Ивановне.
    </p>
</div>
```

Здесь контейнер содержит два контейнера , каждый из которых заключает в себя некий текст.

Самый крупный контейнер для содержательной части документа — тег <body> (тело документа), он содержит другие контейнеры. Например, формируя письмо, мы могли бы написать фрагмент кода на языке разметки, приведенный в листинге 1.2.

Листинг 1.2. Фрагмент кода на языке разметки

```
<body>
  <div>
    <p>
     Здравствуйте, дорогой Иван Иванович.
    </p>
    <p>
     В первых строках своего письма передаю поклон Марье Ивановне.
    </p>
  </div>
</body>
```

Итак, мы уже упомянули несколько тегов: , <br>, <body>. В HTML имеется много других тегов для группировки элементов документа в логические блоки, образующие в совокупности структуру документа.

На рис. 1.1 в окне обычного текстового редактора показан пример текста, размеченного тегами абзаца и перевода строки <br>, а также его вид в окне браузера. Хотя данный документ в целом оформлен не по всем правилам и даже совсем не по правилам HTML, основные Web-браузеры все-таки отображают его, причем одинаково. Дело в

том, что файл text.htm, содержащий теги HTML, был открыт в Web-браузере. Все современные браузеры при открытии в них файлов с расширениями htm или html пытаются интерпретировать последние как HTML-коды, т. е. как поток тегов языка разметки. Однако это еще не означает, что формировать Web-документ можно "как попало", вспоминая и используя те или иные теги. Правила создания Web-документов существуют, и их настоятельно рекомендуется соблюдать, чтобы избежать несовместимости с различными браузерами и других проблем.

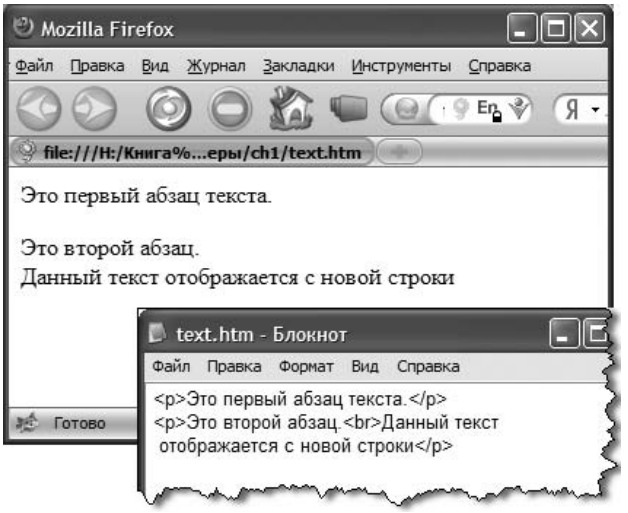

Рис. 1.1. Текст, размеченный тегами абзаца <p> и перевода строки <br>, в текстовом редакторе и в браузере

Обратите внимание, что в рассматриваемом примере расположение текста в окнах редактора и браузера различается. В текстовом редакторе символы могут располагаться в одной или нескольких строках, что еще не определяет однозначно их расстановку в окне браузера. При наборе символов в текстовом редакторе нажатие на клавиши <Enter>, <Tab> и т. п. приводит к форматированию текста, визуально контролируемому с помощью дисплея. Текстовый редактор отвечает на нажатие указанных клавиш расстановкой в вводимом тексте специальных управляющих символов, которые не видны в общем потоке печатаемого текста. Для управления расположением текста в окне браузера требуется явное задание управляющих символов в виде специальных тегов языка разметки. В рассматриваемом примере тег абзаца <p> указывает браузеру отобразить заключенный в него текст с новой строки и с отступом в одну строку. В обычном текстовом редакторе такого же эффекта можно добиться, дважды нажав на клавишу <Enter> в конце введенного текста. Далее, переход на новую строку внутри второго абзаца задан с помощью тега <br>. Без него этот переход либо отсутствовал бы, либо произошел лишь при достаточно малой ширине окна браузера. Результат зависит от используемого браузера. Дело в том, что браузеры пытаются, так или иначе, перенести поток символов на новую строку, если ширина окна не позволяет расположить текст вдоль одной горизонтальной линии.

Чтобы текст и другие элементы документа отображались инвариантно относительно различных браузеров, необходимо принудительно указывать переносы, отступы и дру-

гие управляющие символы с помощью тегов языка разметки и специальных средств (атрибутов тегов и параметров CSS, о чем будет рассказано далее).

О правилах оформления HTML-документов мы поговорим позже, а сейчас остановимся на одной важной особенности HTML. Не все, но многие теги определяют не только структуру документа, но и внешний вид его элементов в окне браузера, т. е. задают еще и способ отображения соответствующей информации в браузере. Например, тег абзаца `<p>` определяет, что все следующее за ним будет отображено с новой строки и с пропуском одной строки; текст, заключенный между тегами `<h1>` и `</h1>`, будет выведен с новой строки максимальным по размеру шрифтом, а последующая информация будет размещена с новой строки (иначе говоря, контейнер `<h1>` задает заголовок первого уровня); текст в контейнере `<i>` выводится курсивом.

Таким образом, разметка тегами HTML в большинстве случаев предопределяет не только структурную декомпозицию документа, но и его внешний вид в окне браузера. Разумеется, структурная декомпозиция, в конечном счете, должна быть как-то выражена внешним образом. Например, листая книгу, мы видим ее структуру в виде иерархии глав, разделов, подразделов и абзацев. Вместе с тем, нетрудно понять различие между собственно структурной декомпозицией и внешним представлением ее элементов. В самом деле, главы, разделы и подразделы можно отобразить как линейно следующие друг за другом, так и расположенные на страницах в двух и более колонках; заголовкам можно придать тот или иной вид, манипулируя такими параметрами шрифта, как цвет, размер и начертание. При этом логическая структура документа остается постоянной, а изменяется только ее представление на бумаге или в окне браузера. Однако идея четко разделить структурный и "представительский" аспекты языка разметки в HTML не реализована в полной мере. В HTML средства структурной декомпозиции документа и его внешнего представления оказались изначально сильно связаны между собой. Это обстоятельство и сыграло, на мой взгляд, решающую роль в чрезвычайно широкой популярности данного языка разметки: разработчики Web-страниц хотели сразу видеть результат разметки документа в браузере и получили это (см. рис. 1.1).

Для определения элементов, вставляемых в документ, одних только тегов бывает недостаточно. Так, для вставки графического изображения служит тег `<img>`. Однако требуется еще указать источник изображения, т. е. адрес или, точнее, URL (Uniform Resource Locator — унифицированный указатель ресурса) файла. В подобных случаях используются параметры тегов, называемые еще атрибутами. Один или несколько атрибутов записывают в открывающей части тега в произвольном порядке в следующем виде: *`имя_атрибута="значение"`*. Атрибуты отделяют друг от друга пробелами. Вот пример тега для вставки графического изображения:

```
<img src="images/mypicture.jpg">
```

Тег `<img>` не имеет заключительной части вида `</img>`.

Чтобы указать размеры графического изображения на странице (в пикселах), можно записать следующее выражение:

```
<img src="images/mypicture.jpg" width="200" height="100">
```

Здесь `src`, `width` и `height` — имена атрибутов тега `<img>`, за которыми следует знак равенства и, далее, значения, указанные в кавычках, двойных или одинарных.

Атрибуты позволяют указать, например, цвет фона, размеры, начертание и цвет шрифта, характеристики выравнивания элементов документа и т. п. Так, в контейнерном теге `<body>`, задающем основную часть документа, с помощью атрибутов `text` и `bgcolor` можно определить цвет текста и фона:

```
<body text="#ff0000" bgcolor="#00ffee">.
```

Таким образом, развитие HTML пошло в сторону добавления средств, определяющих внешнее представление документов. Стали появляться новые теги и их новые атрибуты. На первых порах производители браузеров (Microsoft, Netscape, Sun Microsystems и др.) создавали свои варианты HTML, отличающиеся не только наборами тегов, но и способами их интерпретации. В сложившихся условиях один и тот же документ мог выглядеть по-разному в зависимости от браузера, что явно противоречило цели всемирного распространения информации. Очевидно, стандарты стали насущной необходимостью.

Разработкой стандарта для HTML занялась международная организация World Wide Web Consortium (W3C, Консорциум Всемирной паутины), в состав которой вошли крупнейшие производители программного обеспечения для работы в Интернете (в том числе Microsoft, Sun Microsystems, Netscape и др.). Сведения о стандартах (спецификациях) можно найти на официальном сайте W3C по адресу **www.w3.org**. Схема подготовки спецификаций консорциумом примерно такая: сначала выпускается проект или черновик (draft) спецификации, в результате обсуждения которого появляется ее рабочий вариант. Последний предлагается к обсуждению в течение некоторого времени, по окончании которого рабочий вариант может стать рекомендацией — официальным вариантом спецификации. Браузеры должны интерпретировать язык разметки соответственно стандарту, хотя на практике это происходит не в полной мере. Одно дело идея, другое — ее реализация.

В июле 1997 г. консорциум W3C выпустил проект спецификации HTML 4.0, который уже в декабре стал официальной рекомендацией для производителей Web-браузеров, а с декабря 1999 г. существует версия 4.01. Тем не менее различия в интерпретации основными браузерами некоторых тегов и атрибутов официальной спецификации HTML все же остались. Для обеспечения дополнительных возможностей визуального представления, программирования поведения, динамичности и интерактивности документов появились каскадные таблицы стилей (CSS — Cascading Style Sheets) и скрипты (сценарии), написанные на специальных языках, наиболее популярный из которых — JavaScript.

Параллельно с HTML развивался другой язык разметки — XML (eXtensible Murkup Language — расширяемый язык разметки). Он был создан как средство структуризации информации для обмена между программами. Web-браузер — одна из них, но далеко не единственная. При этом на способ внешнего представления структурированных данных никаких ограничений не накладывалось. Главная задача, которая ставилась при разработке этого языка, состояла в обеспечении совместимости между различными системами обработки структурированных данных. Передающей стороне не важно, как будет отображен XML-документ получателем, требуется лишь, чтобы он однозначно разобрался в структуре принятого документа.

В XML, так же как и в HTML, есть теги и атрибуты. Однако, в отличие от HTML, в языке XML они не предопределены изначально, а могут создаваться автором доку-

мента по своему усмотрению. Теги XML задают лишь структуру документа, но не его представление. На рис. 1.2 показан фрагмент XML-кода в текстовом редакторе и результат его обработки Web-браузером. Теги `<manual>` и `<step>` были придуманы автором данной книги для представления структуры инструкции по созданию сайта. Web-браузер не "знает" таких тегов, а потому просто показывает XML-код без какой-либо его интерпретации. Внешний вид элементов XML-документа определяется отнюдь не тегами, а специальными программами-анализаторами, часто называемыми парсерами (parser). Основные Web-браузеры имеют такие встроенные парсеры, но чтобы подключить их к работе и заставить отобразить собственно информацию (без тегов), необходимо указать, что содержимое документа является XML-кодом и, кроме того, сослаться на каскадную таблицу стилей (CSS).

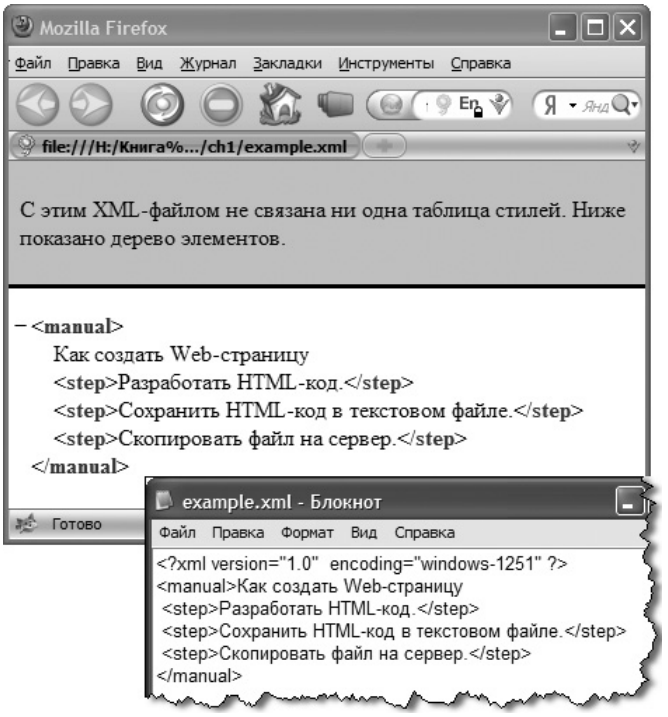

Рис. 1.2. Внешний вид XML-кода, загруженного в браузер

Таблицы стилей сохраняются в отдельных файлах и содержат параметры визуализации элементов (тегов) XML-документа. На рис. 1.3 показан пример, в котором XML-документ, представляющий инструкцию, в первой строке содержит дескриптор, указывающий тип документа, а во второй строке — ссылку на файл example.css с правилами внешнего представления тегов `<manual>` и `<step>`. В данном случае браузер не отображает теги, а выводит результат их интерпретации. Так, в таблице стилей указаны цвет и "жирность" шрифта, а также требование, согласно которому каждый элемент `<step>` (шаг инструкции) занимает по горизонтали 200 пикселов (`width:200px`) с отступом 20 пикселов слева (`padding-left:20px`) и начинается с новой строки (`display:block`).

При желании можно модифицировать параметры отображения элементов документа, оставляя неизменной его логическую структуру.

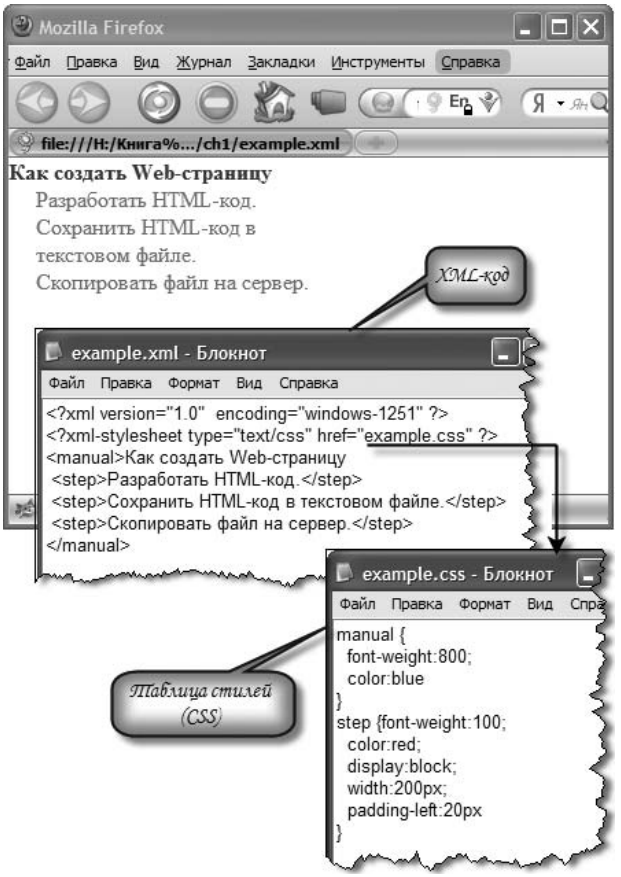

Рис. 1.3. Пример использования таблицы стилей для отображения XML-документа в окне браузера

Итак, в XML воплотилась хорошая идея разделения между структурным и внешним представлением содержания документа. В HTML это не так, но его можно немного модернизировать, чтобы существенно приблизить к XML. На пути к достижению этой цели появился XHTML (eXtensible HyperText Markup Language — расширяемый язык разметки гипертекста).

> ***Примечание***
>
> XML — подмножество более сложного языка SGML (Standard Generalized Markup Language — стандартный обобщенный язык разметки). На основе XML созданы более специализированные языки разметки, например XHTML, SVG, RSS, MathML и др. Такие языки еще называют словарями данных языка XML. С другой точки зрения XML представляет собой текстовый формат описания иерархических баз данных.

Еще до появления XHTML многие разработчики Web-сайтов практиковали следующий технологический прием. С помощью HTML-тегов создавалась лишь структура документа, при этом теги и атрибуты, предназначенные только для задания внешнего вида элементов, практически не использовались (насколько это было возможно), а отображение элементов документа в окне браузера или при печати определялось только в каскадных таблицах стилей. Данный прием можно назвать "применением HTML в стиле XML".

Разметку Web-страницы можно реализовать на основе языка как HTML, так и XHTML. Однако XHTML более строг, ближе к универсальному языку XML и должен поддерживаться одинаковым образом всеми современными браузерами. В данной книге все примеры написаны с использованием синтаксиса XHTML.

Читатели, знакомые только с HTML, увидят не слишком много отличий этого языка от XHTML. Перейти от HTML-кодов к XHTML-кодам довольно легко. Языки HTML и XHTML очень схожи, но различия все же есть. Вот основные (но далеко не все) из них:

- язык HTML регистронезависимый: не имеет значения, строчными или прописными буквами записаны имена тегов и атрибутов. В XHTML для записи имен тегов и атрибутов рекомендуется указывать только строчные буквы. Например, вместо

  ```
  <IMG SRC="mypicture.jpg">
  ```

  следует писать

  ```
  <img src="mypicture.jpg"/>;
  ```

- в HTML большинство тегов контейнерные (например, `<h1>...</h1>`), т. е. имеют открывающий и закрывающий дескрипторы. Вместе с тем есть и неконтейнерные теги: для дескриптора `<тег>` нет закрывающего дескриптора `</тег>`. Например, для вставки графического изображения в HTML применяется тег `<img src="mypicture.jpg">`, а в XHTML — `<img src="mypicture.jpg"/>`. Неконтейнерные теги еще называют пустыми. В XHTML для таких тегов необходимо указывать слэш (косую черту) перед закрывающей уголковой скобкой: `<тег ... />`, например

  `<img src="mypicture.jpg"/>` или `<br/>`;

- в некоторых тегах HTML имеются атрибуты без значений (так называемые булевы или логические атрибуты), например `<option selected>`. В XHTML для подобных атрибутов следует явно указывать строковое значение, совпадающее с именем соответствующего атрибута, например

  ```
  <option selected="selected" />;
  ```

- в XHTML значения атрибутов тегов нужно заключать в кавычки. В HTML это делать не обязательно. Например, в XHTML следует писать `<img src="mypicture.jpg"/>`, а не `<img src=mypicture.jpg>`, как допустимо (хотя тоже не рекомендуется) в HTML. Некоторые атрибуты принимают числовые значения, которые в HTML можно (хотя и не рекомендуется) указывать без кавычек, но в XHTML их всегда следует заключать в кавычки, двойные или одинарные, например

  ```
  <img src="mypicture.jpg" width="100" height="150"/>;
  ```

- в содержательной текстовой части XHTML-документа нельзя непосредственно использовать символы, которые применяются в коде, такие как `<`, `>` и `&`, причем их недопустимо записывать даже в значении URL-адреса. Эти специальные символы следует заменять соответствующими буквенными или числовыми эквивалентами `&lt;`, `&gt;` и `&amp;` (или `&#60;`, `&#62;` и `&#38;`) соответственно. Об эквивалентах специальных символов рассказано в *разд. 7.3*;

- кодировка файла по умолчанию для документа HTML 4 — ISO 8859-1, а для документов XHTML и HTML 5 — UTF-8.

Ошибки в HTML-коде браузеры пытаются преодолеть тем или иным образом, так что даже весьма небрежно написанный код все же как-то отображается (во многих случаях вполне удовлетворительно). По рекомендации W3C в случае ошибки в XHTML-коде браузеры должны сообщать об этом и прекращать дальнейшую обработку. Таким образом, XHTML-код подвергается более тщательному анализу прежде, чем начнется его интерпретация — отображение в окне браузера. HTML-код, напротив, не анализируется предварительно на правильность, а отображается в порядке следования тегов. При этом сообщения об ошибках не выводятся, а отображение продолжается, даже если ошибки все же возникли. Очевидно, HTML более "дружественен" разработчику, чем XHTML. Но более других эту "дружественность" ценят малоопытные и небрежные разработчики своих домашних страниц, а не авторы серьезных Web-проектов. Для последних однозначность интерпретации и дисциплина важнее "угодливости" браузеров, вредность которой проявляется при поиске трудноуловимых смысловых ошибок и при попытках обеспечить межбраузерную инвариантность представления документов.

В XHTML сохранилось большинство тегов HTML 4. Однако некоторые старые HTML-теги, введенные в свое время для обеспечения дополнительных возможностей отображения, теперь применять не рекомендуется.

В настоящее время существуют официальные версии XHTML 1.0 и 1.1 и черновая спецификация XHTML 2.0. Однако работа над последней остановлена в пользу создания нового стандарта — HTML 5.

Начиная с 2004 г. совместными усилиями рабочих групп W3C HTML WG и WHATWG с привлечением таких компаний, как Apple, Google, Mozilla, Opera, Microsoft и др., ведется работа по созданию спецификации HTML 5. К середине 2010 г. существовал лишь черновой ее вариант, а к началу 2014 г. — кандидат в рекомендации (**www.w3.org/TR/html5**). Все современные ведущие браузеры поддерживают основные элементы HTML 5.

HTML 5 призван стать преемником как HTML 4, так и XHTML 1.0, обеспечивая совместимость с ними, по крайней мере, в самом важном. В HTML 5 возможен синтаксис как обычного HTML, так и XHTML. Вместе с тем, ряд тегов и атрибутов, не рекомендованных к использованию в прежних версиях, в новом стандарте отсутствуют. Например, теги `<applet>` для вставки Java-апплетов, `<center>` для центрирования содержимого по горизонтали и `<frameset>`, `<frame>` для разбиения окна браузера на статические фреймы в HTML 5 не поддерживаются. С целью обеспечения обратной совместимости в спецификации HTML 5 описывается, как должен поступить браузер при наличии непредусмотренных тегов. Уже существующие сайты не должны пострадать из-за появления нового языка разметки.

В HTML 5 вводится в оборот ряд новых тегов с отчетливо выраженной семантикой, предназначенных для повышения эффективности разработки Web-приложений, а также для помощи поисковым системам. Так, для создания логической структуры и упрощения типовой верстки документа предлагаются теги `<header>` (заголовок), `<nav>` (навигационная панель), `<article>` (статья), `<section>` (раздел, секция статьи), `<aside>` (боковая колонка), `<footer>` ("подвал"), `<menu>` (меню) и др. В прежних версиях указанные элементы обычно создаются тегами `<div>` и сразу понять смысл их применения довольно трудно. Иначе говоря, семантика для `<div>` неоднозначна.

Элементы `<input>` полей ввода данных в HTML 5 приобрели новые атрибуты, заметно облегчающие создание пользовательского интерфейса: `time`, `email`, `url`, `required` и др. Эти атрибуты во многих случаях позволят обойтись без скриптов, проверяющих правильность введенных данных.

Особого внимания заслуживает элемент `<canvas>`, предоставляющий область на странице, в которой с помощью скриптов на языке JavaScript можно рисовать изображения, подобные создаваемым посредством SVG или Flash. Для вставки мультимедиа в HTML 5 предусмотрены теги `<audio>` и `<video>`, обеспечивающие внедрение на страницу звука и видео с возможностью управления.

HTML 5 предусматривает многие другие полезные и удобные возможности. В данной книге будут описаны лишь те из них, которые уже поддерживаются ведущими браузерами.

1.2. Что такое таблицы стилей

Как уже говорилось, параметры внешнего вида документа можно задать с помощью *каскадных таблиц (листов) стилей* (Cascading Style Sheets, CSS). Таблицу стилей, подобно шаблону форматирования текстов, можно разработать отдельно от конкретного (X)HTML-документа, а затем применить к нему. Модификация содержимого таблицы стилей меняет внешний вид (X)HTML-документов, не затрагивая их структуры и информационного содержания. Одна и та же таблица стилей может применяться к нескольким документам и, наоборот, к одному и тому же документу может быть применено несколько таблиц стилей. В последнем случае браузер учитывает приоритеты таблиц и по определенным правилам разрешает возникающие конфликты, в результате чего таблицы выстраиваются неким каскадом (отсюда и название — каскадные таблицы стилей).

Кроме технологичности стилизации (X)HTML-документов, CSS обеспечивает еще и произвольное позиционирование элементов. Для любого элемента можно задать размеры и координаты расположения, а также другие параметры визуализации. Так что, применяя CSS, можно обойтись без тегов таблиц и фреймов, которые широко используются как средство компоновки Web-страниц.

CSS содержит наборы стилевых параметров или, другими словами, правила форматирования. Например, если требуется определить для всех заголовков первого уровня шрифт Courier 24 пункта красного цвета, то соответствующее правило можно записать так:

```
h1 {font-family: Courier; font-size: 24pt; color: red}
```

Здесь в фигурных скобках указан список стилевых параметров — пар вида *имя:значение*, разделяемых точкой с запятой; левее этого списка указано имя `h1` тега, к которому данные параметры требуется применить. В результате заголовок первого уровня (содержимое тега `<h1>`) приобретает специальные параметры, установленные автором стиля, а не принятые по умолчанию, когда авторский стиль не задан.

Одно и то же правило можно применить к различным элементам и, наоборот, для одного и того же элемента можно задать несколько правил. Следующее правило определяет

одинаковые стилевые параметры одновременно для заголовков первого и второго уровней:

```
h1, h2 {font-family: Courier; font-size: 24pt; color: red}
```

А вот пример каскадного задания стилевых параметров:

```
h1, h2 {font-size: 24pt; color: red}
h1 {font-family: Arial}
h2 {font-family: Courier}
```

Здесь сначала задаются размер и цвет шрифта для обоих заголовков, а затем для каждого из них определяется своя гарнитура шрифта (имя шрифта).

Стилевые параметры, заключенные в фигурные скобки, можно назначить не только именам тегов, но и тегам по значениям их атрибута `id` (идентификатора элемента). Так можно изменить представление не всех, например, заголовков первого уровня (тегов `<h1>`), а только тех, которые имеют указанное значение атрибута `id`. Существует также возможность определить правила, изначально не связанные ни с какими элементами документа, а затем сослаться на них из любого элемента с помощью атрибута `class`. Указатели на элементы, к которым применяются стилевые параметры в фигурных скобках, называются селекторами. Селекторы, как мы только что видели, могут быть различных видов. Подробнее о них будет рассказано в *разд. 3.2*.

Правила CSS можно применить к любому видимому элементу (X)HTML-документа. Почти каждому элементу, заданному тем или иным тегом, можно придать произвольное представление. Поэтому оказывается вполне достаточно одного или нескольких тегов, например `<div>` и/или `<span>`, а визуальное разнообразие их вхождений в документ можно обеспечить посредством только CSS. Однако при этом, возможно, будет нанесен ущерб свойству (X)HTML представлять логическую структуру документа. Действительно, глядя на (X)HTML-документ, содержащий одни только теги `<div>`, будет трудно понять, где заголовок текстового документа, а где абзац основного текста. Поэтому многие опытные разработчики рекомендуют не придерживаться чрезмерного "тегового минимализма", а использовать основные структурообразующие теги по прямому назначению. Например, теги `<h1>`, ..., `<h6>` следует применять только для создания заголовков, а не, скажем, оформления гиперссылок. Кстати, HTML 5 явно поддерживает идею семантической определенности тегов, вводя в оборот элементы `<header>`, `<nav>`, `<article>`, `<section>` и т. п.

Правила CSS можно записать непосредственно в (X)HTML-документе или сохранить в отдельном файле, чтобы применить их одновременно к нескольким документам. В последнем случае браузер кэширует таблицу стилей и, следовательно, последующие страницы, задействующие ту же таблицу стилей, загружаются быстрее. Правила, записываемые непосредственно в самом (X)HTML-документе, заключают в контейнерный тег `<style>`, а для ссылки на таблицу стилей, расположенную во внешнем файле, в контейнер `<head>` вставляют тег

```
<link href="URL css-файла".../>
```

или в контейнер `<style>` добавляют директиву

```
@import url("URL css-файла");
```

На рис. 1.4 показан пример таблицы стилей, непосредственно вставленной в (X)HTML-документ. Если правила этой таблицы разместить в текстовом файле mystyle.css, то вместо тега `<style>` следует указать ссылку на данный файл, например, так:

```
<link rel="stylesheet" type="text/css"
href="http://www.anyserver.ru/mystyle.css" />
```

или (в рамках тега `<style>`) так:

```
@import url("http://www.anyserver.ru/mystyle.css");
```

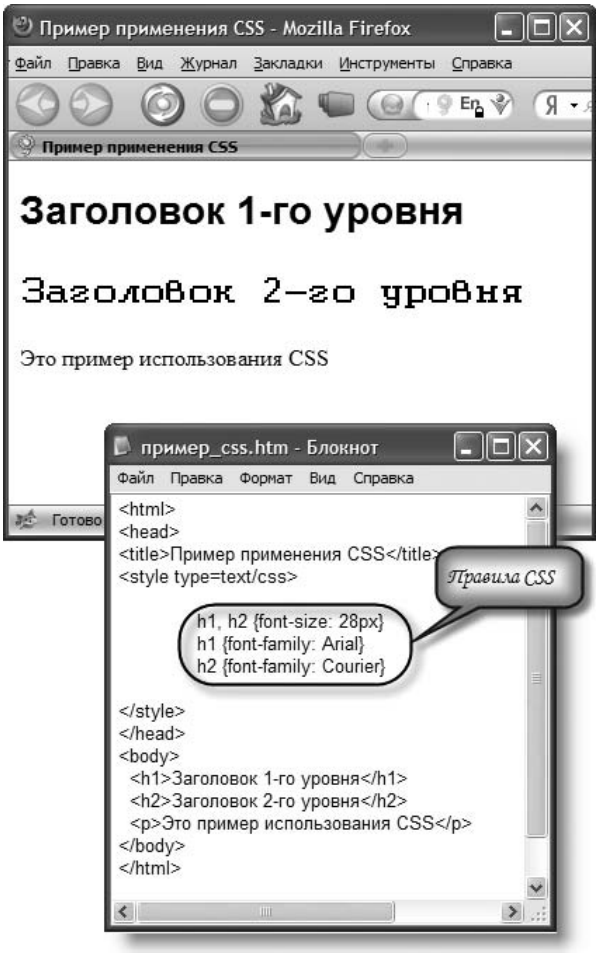

Рис. 1.4. Пример использования таблицы стилей в (X)HTML-документе

Спецификация каскадных таблиц стилей уровня 1 (CSS 1) была принята в качестве рекомендации W3C в декабре 1996 г., а в январе 1999 г. вышел ее откорректированный вариант. Рекомендация спецификации уровня 2 (CSS 2) была выпущена в мае 1998 г. Она сохраняла обратную совместимость с CSS 1. Пересмотренный ее вариант (CSS 2.1) увидел свет в апреле 2008 г. и вошел составной частью в спецификацию CSS 3, достигшую стадии рабочего проекта (working draft). CSS 3 предоставляет существенно новые возможности, начиная от простых визуальных эффектов (тени, скругленные углы)

и заканчивая трансформацией и анимацией. С текущим состоянием работ по CSS 3 можно познакомиться, например, по адресу **www.w3.org/Style/CSS/current-work**.

Хотя спецификация CSS 3 пока находится в разработке, различные браузеры уже поддерживают в той или иной степени многие из ее элементов. При этом имена параметров, возможно, придется модифицировать путем добавления префикса в зависимости от движка браузера. Так, для Mozilla Firefox (движок Gecko) добавляется префикс `-moz-`, для Google Chrome и Apple Safari (движок WebKit) — `-webkit-`, для Opera — `-o-`, для Microsoft Internet Explorer (движок Trident) — `-ms-`. В последующих версиях браузеров необходимость добавления к именам параметров специфических префиксов, возможно, отпадет.

Более подробно таблицы стилей мы рассмотрим в *главе 3*, а сейчас вернемся к языку разметки документа.

ГЛАВА 2

Структура (X)HTML-документа

Как известно, Web-узел (сайт) состоит из одной или нескольких страниц, каждая из которых представлена своим кодом, написанным на языке разметки HTML или XHTML. Такой код мы будем также называть (X)HTML-документом. Он имеет определенную структуру и сохраняется в обычном текстовом файле с типичными расширениями htm или html. Файл с (X)HTML-документом главной страницы сайта, которая загружается в браузер первой, обычно называется index.htm или index.html и размещается в корневой папке сайта на Web-сервере. При активизации ссылки на сайт без указания имени файла загружается именно index.htm или index.html, если он будет найден на сервере по указанному адресу в корневой папке сайта. Разумеется, вы можете назначить любое имя файлу с (X)HTML-кодом вашей главной страницы, но тогда URL-адрес, помимо адреса сайта, должен будет содержать и имя файла.

Прежде всего, (X)HTML-документ, соответствующий вашей Web-странице, должен содержать описание типа документа (как требует стандарт DTD — Document Type Definition), после чего начинается собственно код — набор тегов, т. е. дескрипторов языка разметки (X)HTML.

Теги записывают в одну или несколько строк, с отступами или без, что не влияет на отображение соответствующих элементов в окне браузера. Выбирая способ расположения элементов кода, следует исходить из удобства его чтения, но не забывать при этом, что символы пробелов, табуляции и перевода строки вносят свою лепту в общий объем файла. Важно, чтобы после открывающей уголковой скобки (<) сразу же (без пробелов) следовало имя тега, в то время как закрывающая уголковая скобка может находиться сколь угодно далеко. Это не единственное правило (X)HTML, но новички очень часто игнорируют его, из-за чего возникают досадные ошибки.

2.1. Определение типа документа: дескриптор <!DOCTYPE...>

В *разд. 1.1* упоминалось, что для разметки документа могут использоваться различные языки, такие как HTML, XHTML, XML и др. Для правильной интерпретации языка, на котором написан документ, браузеру следует сообщить, какой именно язык выбран. Это делается посредством специального дескриптора (тега).

Собственно (X)HTML-код Web-страницы заключается в контейнерном теге <html> (т. е. между дескрипторами <html> и </html>). Однако при создании настоящего (X)HTML-документа рекомендуется дать его описание, расположенное перед тегом <html> и состоящее из одного или двух специальных дескрипторов. В одном из таких дескрипторов указывается тип документа (DTD — Document Type Definition).

Тип документа HTML 5 задается специальным дескриптором:

```
<!DOCTYPE html>
```

Если вы хотите создать Web-страницу (HTML-документ) с самого начала (как говорят, с нуля), то в первой строке своего HTML-кода напишите <!DOCTYPE html> и можете сразу перейти к следующему разделу.

Дескриптор <!DOCTYPE html> подойдет и в случае, если код вашего документа использует лишь возможности прежнего HTML 4 или же написан с применением синтаксиса XHTML. Стандарт HTM 5 предоставляет новые возможности и обеспечивает совместимость с устаревшими версиями путем указаний, как браузеры должны действовать, если встретились с теми или иными записями в документе, которые не соответствуют HTML 5.

Ранее, до появления HTML 5, для документов XHTML и HTML 4 дескриптор <!DOCTYPE...> имел более сложную структуру. Вы можете заменить DTD своих старых (X)HTML-документов новым дескриптором <!DOCTYPE html>. Однако в этом нет необходимости. Поскольку многие разработчики ничего не изменяют в своих DTD, я приведу в качестве справочной информации общую схему и примеры DTD, используемые для XHTML, HTML 4 и некоторых других языков разметки. Эта схема имеет следующий вид:

```
<!DOCTYPE элемент_верхнего_уровня публичность
"регистрация//организация//тип имя//язык" "url">.
```

Атрибуты дескриптора <!DOCTYPE...>:

- *элемент верхнего уровня* — для XHTML и HTML элемент верхнего уровня — html;
- *публичность* — возможны значения PUBLIC (для публичных документов) и SYSTEM (для системных ресурсов);
- *регистрация* — возможны значения + (разработчик DTD зарегистрирован в международной организации по стандартизации ISO) и – (разработчик не зарегистрирован в ISO); для консорциума W3C указывается –;
- *организация* — уникальное название организации, разработавшей данный DTD; для (X)HTML-документов такой организацией является W3C;
- *тип* — тип описываемого документа; для (X)HTML-документов указывается DTD;
- *имя* — уникальное имя документа с описанием DTD, например XHTML 1.0 strict (для XHTML версии 1.0 строгой схемы) или HTML 4.01 transitional (для HTML версии 4.01 переходной схемы);
- *язык* — язык, на котором описан объект; значение состоит из двух букв в верхнем регистре; для (X)HTML-документов указывается английский язык (EN);
- *url* — URL-адрес файла с описанием DTD, например http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd.