Вадим Дунаев

(X)HTML
скрипты и стили
САМОЕ НЕОБХОДИМОЕ

Санкт-Петербург

«БХВ-Петербург»

2009

УДК 681.3.06
ББК 32.973.26-018.2
Д83

Дунаев В. В.

Д83 (X)HTML, скрипты и стили. Самое необходимое. — СПб.: БХВ-Петербург, 2009. — 496 с.: ил.

ISBN 978-5-9775-0322-8

Рассмотрены приемы создания Web-сайтов с помощью языка разметки гипертекста (HTML, XHTML), каскадных таблиц стилей (CSS) и сценариев на языке JavaScript. Изложены необходимые сведения и приведены многочисленные примеры типичных задач разработки сайтов. Предлагаемые решения инвариантны относительно пяти наиболее популярных браузеров, таких как Microsoft Internet Explorer 7.0, Mozilla Firefox 3.05, Opera 9.63, Apple Safari 3.2.1 для Windows и Google Chrome 1.0. Особенность книги — сравнение различных технологических приемов Web-программирования и решение практических задач несколькими путями для достижения наилучшего результата. Приложения содержат краткие сведения о тегах HTML, свойствах CSS 2 и краткое руководство по JavaScript.

Для начинающих Web-программистов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Леонид Кочин*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 27.02.09.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 39,99.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.003650.04.08 от 14.04.2008 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0322-8

Оглавление

Введение

Как известно, Web-сайты создаются с помощью HTML (языка разметки гипертекста), CSS (каскадных таблиц стилей) и скриптов (сценариев на языке JavaScript и др.). Всем этим средствам и посвящена данная книга, рассчитанная главным образом на новичков. Наряду с краткими теоретическими сведениями читатель найдет здесь многочисленные примеры решения типичных задач разработки сайтов.

Еще совсем недавно можно было писать HTML-коды весьма небрежно. Браузеры (т. е. интернет-обозреватели), особенно Microsoft Internet Explorer, снисходительно пытались "уразуметь" замысел разработчика HTML-кода, чтобы возможно лучше его интерпретировать. Каждый браузер делал это по-своему. В результате Всемирная паутина (World Wide Web) быстро наполнилась различного рода ресурсами, что и требовалось на первых порах. Однако эти ресурсы по-разному отображались различными браузерами. Иначе говоря, возникла проблема межбраузерной совместимости публикуемых в Сети документов. Было время, когда в ожесточенной конкурентной борьбе выяснялось, кто кого, Microsoft или Netscape. Образно выражаясь, обе фирмы держали два фронта: совершенствование своего браузера и влияние на создание некоего общего стандарта, которому должны следовать все разработчики каких бы то ни было Web-браузеров. Для обычных пользователей важнее всего удобство и функциональность браузера, а для разработчиков сайтов — совместимость интернет-обозревателей или инвариантность кода страниц относительно различных браузеров. В популярных статьях о достоинствах и недостатках различных браузеров эти два аспекта, как правило, завуалированы, а потому вызывают недоразумения.

Борьба двух ведущих производителей браузеров закончилась положительно в том смысле, что в настоящее время, похоже, возобладала здравая мысль следовать неким стандартам, разработкой которых занимается организация Word Wide Web Consortium (W3C) **http://www.w3c.org**.

Теперь все современные браузеры следуют стандартам W3C, хотя пока и не в полной мере, что вполне естественно. В будущем, вероятно, расхождения со стандартами если и не будут сведены к нулю, то существенно уменьшатся.

А что же делать разработчикам сайтов сейчас? Лучше всего, пожалуй, писать коды (HTML, CSS, JavaScript) с соблюдением текущих рекомендаций W3C.

Заголовок этой книги почти повторяет название предыдущей: "HTML, скрипты и стили". Однако это не простое переизложение указанного весьма объемистого труда, а новая книга как по структуре, так и по содержанию. Как и прежде, настоящее пособие рассчитано на начинающих, а моя цель состоит в том, чтобы показать, что и как будет работать в ваших проектах. В предлагаемой книге рассматриваются решения, инвариантные относительно пяти наиболее популярных на сегодняшний день браузеров, таких как Microsoft Internet Explorer 7.0, Mozilla Firefox 3.05, Opera 9.63, Safari 3.2.1 для Windows и Google Chrome 1.0.

Данная книга не является справочником по стандартам (X)HTML, CSS и JavaScript. Ее задача состоит в том, чтобы ввести начинающего разработчика в предметную область создания Web-сайтов, показать некоторые приемы обращения с теми или иными технологическими средствами. Недостающие сведения можно получить из соответствующих справочников, которые нетрудно найти в Интернете. Вместе с тем, хочу заметить, что как в предыдущих книгах, так и в этой, я не навязываю читателю каких-либо технических приемов в качестве "правильных" и единственно возможных для достижения успеха. Практически любую цель можно достичь несколькими путями. И начинающие, и опытные разработчики должны самостоятельно выбирать средства, наиболее подходящие для решения стоящей перед ним задачи. В этом и состоит, на мой взгляд, главное в его индивидуальном творчестве.

В *части I* книги (*главы 1—8*) рассматриваются средства HTML, XHTML и CSS на примерах решения основных задач создания сайтов. При этом я старался не использовать тегов и атрибутов, главное назначение которых состоит в задании внешнего вида элементов, предпочитая им инструменты CSS. Иначе говоря, я старался разделить средства (X)HTML и CSS, насколько это казалось мне возможным.

Часть II (*главы 9—14*) посвящена применению сценариев на языке JavaScript к решению некоторых задач, поучительных для освоения программирования. Одни из них вполне готовы для использования в ваших проектах, а другие дожидаются адаптации к ним.

В *приложениях 1* и *2* приведены списки основных тегов HTML и свойств CSS2 соответственно, а в *приложении 3* — краткое руководство по JavaScript.

Ваши отзывы и замечания я буду рад принять в своей гостевой книге на своем сайте по адресу **http://dunaevv1.narod.ru** или **http://dunaev.jino-net.ru**.

ЧАСТЬ I

HTML и стили

Глава 1

Что такое HTML и таблицы стилей

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

1.1. Разметка документа

Допустим, вам требуется создать документ (письмо другу, резюме для работодателя, объявление о продаже автомобиля, прайс-лист, литературное произведение и т. п.), который вы намерены опубликовать — распечатать на принтере, а затем расклеить на заборах, отправить в издательство или разместить на Web-сайте. Ваш документ будет содержать обычный текст и, возможно, картинки (максимум из того, что вы можете позволить себе при публикации документа на бумажных носителях). Разумеется, все то же самое можно опубликовать и в Интернете. Однако даже в этом простейшем случае нередко возникают порой нетривиальные задачи форматирования или, другими словами, оформления документа. Текст должен быть структурирован, т. е. разбит на части (разделы, подразделы, абзацы), некоторые из них должны иметь заголовки. Даже обычное письмо объемом всего в одну страницу, как правило, разбивается на несколько абзацев, не говоря уже о статьях и более крупных произведениях. 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 размещении документа в Интернете вы можете сделать значительно больше, чем при печати: вставить в документ звуковые и видеоклипы, формы,

заполняемые читателем и отправляемые на сервер, элементы интерфейса (ссылки на другие документы, кнопки, поля ввода и т. п.), с помощью которых можно управлять отображением содержимого и решать другие задачи.

Итак, при подготовке практически любого документа к публикации его требуется отформатировать или, иначе говоря, разметить. Разметка документа состоит в том, чтобы с помощью специальных символов явно указать, что и каким образом следует представить пользователю (читателю) этого документа. Набор специальных символов разметки (меток, дескрипторов, команд или тегов), а также правила их употребления вместе составляют то, что называют языком разметки (Markup Language). В настоящее время наиболее широко распространены такие языки, как HTML, XHTML и XML.

Даже в обычных бумажных документах нередко встречаются ссылки на разделы этого же или другого документа (например, "см. разд. 12.5", "в книге [8] можно найти…" и т. п.). В Web-документах щелчок левой кнопкой мыши на ссылке приводит к отображению в браузере соответствующего документа. Такие ссылки в связи с их мощной и удобной функциональностью называют гиперссылками, а тексты, их содержащие, — гипертекстами (HyperText).

Сейчас наиболее широко известным языком разметки документов для публикации во Всемирной паутине (World Wide Web — WWW) является HTML (HyperText Markup Language — язык разметки гипертекста). Его сущность настолько проста, что легко может быть понята рядовыми пользователями компьютера, не имеющими ни знаний, ни опыта программирования. Теги (дескрипторы) HTML позволяют указать, 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 просто перед этим фрагментом написать тег `<p>`, а после него — `</p>`. Другими словами, текст абзаца следует заключить в контейнер `<p>`.

Чтобы несколько элементов документа (например, абзацев) объединить в один раздел, их следует заключить в какой-нибудь подходящий для этой цели контейнер, например `<div>`. Самый крупный контейнер для содержательной части документа — тег `<body>`. В HTML имеются и другие теги для группировки элементов документа в логические блоки. Большинство тегов HTML являются контейнерными, т. е. для них применяется такой синтаксис:

`<тег> содержимое контейнерного тега </тег>`.

Нетрудно заметить, что контейнерные теги аналогичны скобкам при написании математических формул. Открывающей скобке соответствует открывающий тег <тег>, а закрывающей скобке — замыкающий тег </тег>. Например, Этот текст форматируется как абзац .

Внутри контейнерного тега может находиться обычный текст и/или другой тег. Однако в HTML имеются и неконтейнерные теги, у которых нет замыкающей части с косой чертой вида </тег>. Например, для указания перехода на новую строку служит неконтейнерный тег <br>, для которого не предусмотрена замыкающая часть вида </br>. Вставку в документ графического изображения обеспечивает неконтейнерный тег <img>.

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 текст в целом оформлен не по всем правилам HTML, основные Web-браузеры отображают его одинаково и в соответствии с описанными здесь ожиданиями.

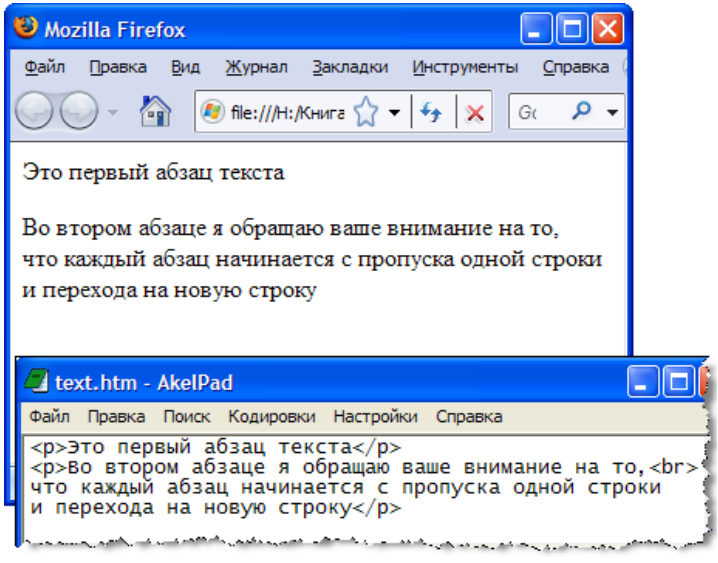

Рис. 1.1. Текст, размеченный тегами абзаца и перевода строки <br>, в текстовом редакторе и в браузере

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. Другими словами, большинство тегов разметки документа

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); содержимое контейнера `<center>` центрируется в окне браузера по горизонтали.

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. Так 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 параметрами шрифта, такими как цвет, размер и начертание. При этом логическая структура документа остается постоянной, а изменяется лишь только ее представление на бумаге или в окне браузера. Однако идея четко разделить структурный и представительский аспекты языка разметки не реализована в полной мере в HTML. В HTML средства структурной декомпозиции документа и его внешнего представления оказались изначально сильно связанными между собой. Именно это обстоятельство и сыграло, на мой взгляд, решающую роль в чрезвычайно широкой популярности данного языка разметки: разработчики Web-страниц хотели видеть результат разметки документа в браузере и получили это (см. рис. 1.1).

Одних только тегов HTML для внешнего представления документов в большинстве практически важных случаев оказалось недостаточно. Поэтому были введены параметры тегов — дополнительные спецификации, называемые обычно атрибутами. 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 указать цвет текста и фона, например

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

Таким образом, развитие HTML пошло в сторону добавления средств, определяющих внешнее представление документов. Стали появляться как новые

теги, так и новые их 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. 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 и Netscape). В декабре 1997 г. спецификация HTML 4, разработанная W3C, стала официальной рекомендацией для производителей Web-браузеров, а с декабря 1999 г. по настоящее время существует версия 4.01. Тем не менее, различия в интерпретации основными браузерами некоторых тегов и атрибутов все же остались.

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 документа по своему усмотрению. Они 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 для представления структуры инструкции по приготовлению яичницы. 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, достаточно сослаться на так называемую каскадную таблицу стилей (CSS — Cascading Style Sheets).

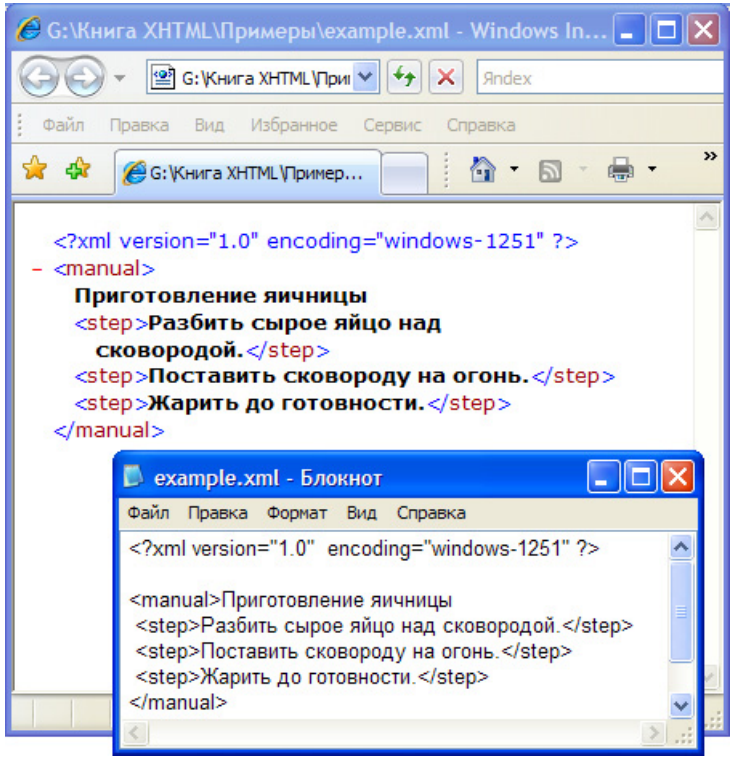

Рис. 1.2. Браузер отображает загруженный в него XML-код

Таблицы стилей сохраняются в отдельных файлах и содержат параметры визуализации элементов (тегов) XML-документа. На рис. 1.3 показан пример, в котором XML-документ 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 не более 200 пикселов (`width:200px`) и начинается с новой строки (`display:block`).

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

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 могут быть созданы более специализированные языки разметки, например, XHTML. Такие языки еще называют словарями данных языка XML. С другой точки зрения, XML — текстовый формат описания иерархических баз данных, которые в середине 1980-х годов были почти полностью вытеснены реляционными базами данных. В данной книге мы не будем более рассматривать XML, а остановимся на одном из его словарей — XHTML.

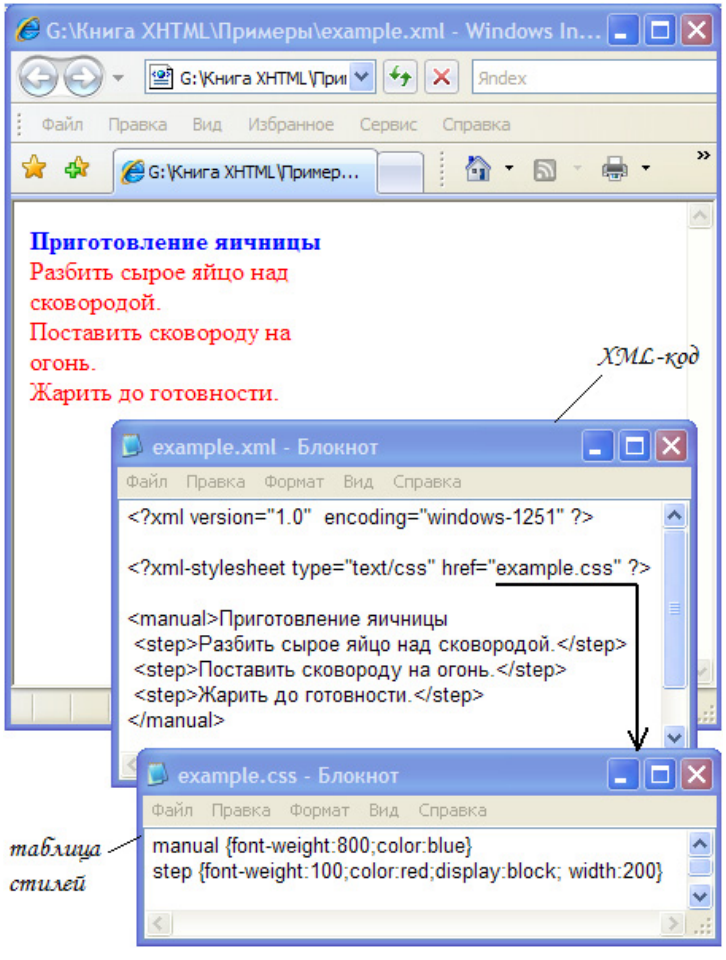

Рис. 1.3. Пример использования таблицы стилей для отображения XML-документа в окне браузера

Еще до появления XHTML многие разработчики Web-сайтов использовали следующий технологический прием. С помощью HTML-тегов создавалась лишь структура документа, при этом теги и атрибуты, предназначенные только лишь для задания внешнего вида элементов, практически не использовались, 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".

В XHTML сохранилось большинство тегов HTML. Однако некоторые старые HTML-теги, введенные в свое время для обеспечения дополнительных возможностей отображения, теперь применять не рекомендуется. Кроме того, все теги должны иметь заключительную часть. Для неконтейнерных HTML-тегов, таких как `<br>` или `<img src="URL-изображения">`, в XHTML-коде следует перед заключительной уголковой скобкой указывать косую черту, например, `<br />` или `<img src="URL-изображения"/>`. В отличие от HTML теги и имена атрибутов в XHTML-документах рекомендуется писать строчными символами (т. е. в нижнем регистре). Имеются и другие особенности, которые мы подробно рассмотрим в следующей главе.

> ***Внимание***
>
> В дальнейшем, когда какие-то сведения справедливы как для HTML, так и для XHTML, мы будем писать (X)HTML.

1.2. Стили

Как уже говорилось, параметры внешнего вида документа можно задать с помощью *каскадных таблиц (листов) стилей* (Cascading Style Sheets, CSS). Таблица стилей, подобно шаблону форматирования текстов, может быть разработана отдельно от конкретного (X)HTML-документа, а затем применена 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 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 фигурные скобки заключен список стилевых параметров — пар вида `имя: значение`; левее этого списка указано имя `h1` тега, к которому данные параметры требуется применить. В результате заголовок первого уровня 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 может быть задано 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`.

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 обеспечивается 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, гиперссылок.

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.

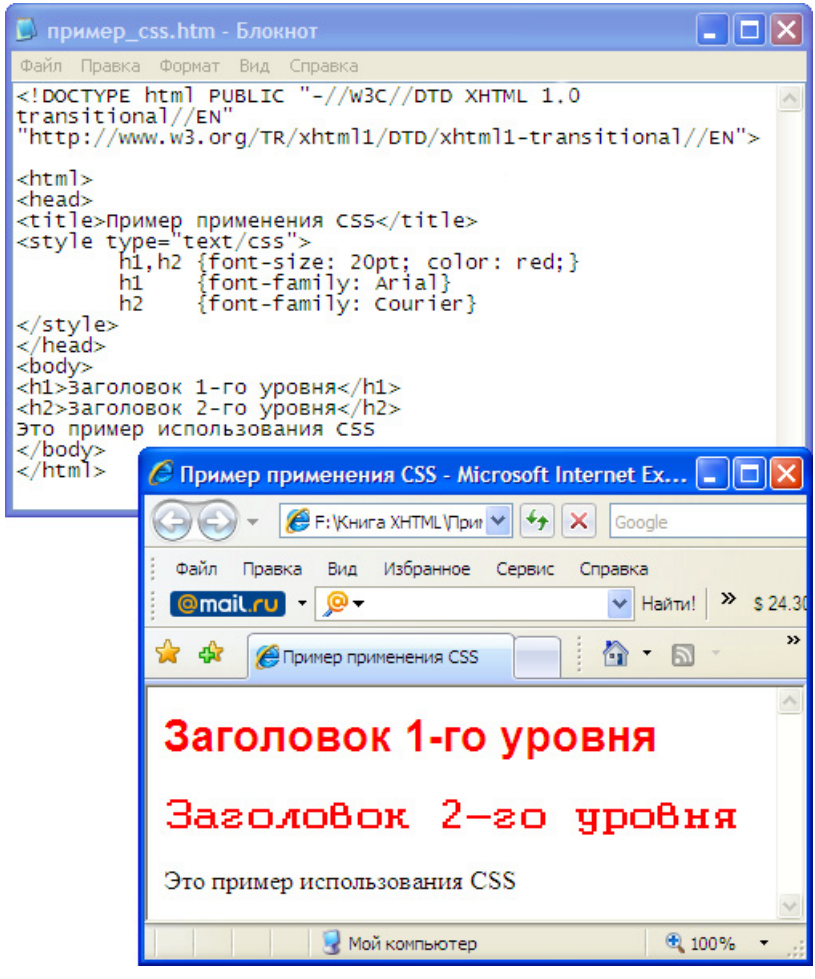

Рис. 1.4. Пример использования таблицы стилей в XHTML-документе

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>` — директиву `@import url(URL css-файла);`.

На рис. 1.4 показан пример таблицы стилей, непосредственно вставленной в XHTML-документ. Если эту таблицу разместить в текстовом файле mystyle.css, то внутри тега `<style>` вместо нее следует указать ссылку на этот файл, например, так:

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

или так:

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

Следует иметь в виду, что браузер Internet Explorer поддерживает ряд своих фирменных (нестандартных) расширений CSS, которые не воспринимаются другими браузерами. К таким расширениям относятся так называемые графические фильтры, которые позволяют в Internet Explorer модифицировать загруженное графическое изображение (например, повернуть его, сделать полупрозрачным и др.). Задать цвет полосы прокрутки окна браузера также можно с помощью специальных параметров CSS, но только для Internet Explorer. Для обеспечения межбраузерной совместимости не рекомендуется применять специфические фирменные расширения CSS.

Глава 2

Структура (X)HTML-документа

Имея дело со сложно устроенным объектом, мы должны, прежде всего, выяснить его структуру, чтобы разложить это сложное образование на более простые для понимания составляющие. (X)HTML-документ состоит из определенных элементов, которые встречаются в нем не вполне случайным образом. Другими словами, (X)HTML-документ обладает некоторой структурой.

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. 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. 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), после чего начинается собственно код — набор тегов (дескрипторов языка (X)HTML).

Теги записываются в одну или несколько строк, что не влияет на отображение соответствующих элементов в окне браузера. Важно, чтобы после открывающей угловой скобки (<) сразу же (без пробелов) следовало имя тега, в то время как закрывающая угловая 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. Основные различия между HTML и XHTML

Для создания сайта необходимо сформировать один или более размеченных документов. Разметку можно реализовать на основе HTML или XHTML. Однако XHTML более строг, ближе к универсальному языку XML и должен поддерживаться одинаковым образом всеми современными браузерами. Поэтому создавать Web-страницы сейчас рекомендуется с помощью языка XHTML. В данной книге все примеры будут написаны на XHTML. Читатели, знакомые только с HTML, увидят не слишком много отличий.

Перейти от HTML-кодов к XHTML-кодам довольно легко. Языки HTML и (X)HTML очень похожи, 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>`). Такие теги имеют открывающий и закрывающий дескриптор; между ними можно поместить другие теги, именно поэтому они называются контейнерными. Вместе с тем, есть и неконтейнерные теги: для дескриптора `<тег>` нет закрывающего дескриптора `</тег>`, например, `<img src="mypicture.jpg">` для вставки графического изображения или `<br>` для перехода на новую строку. Неконтейнерные теги еще называют пустыми. В XHTML для таких тегов необходимо указывать слэш (косую черту) перед закрывающей угловой скобкой: `<тег … />`, например, `<img src="mypicture.jpg"/>` или `<br/>`.
- В некоторых тегах HTML имеются атрибуты без значений (так называемые булевы 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 в HTML.
- В содержательной текстовой части XHTML-документа нельзя непосредственно использовать символы, которые применяются в коде, такие как `<`, `>` и `&`,

причем их нельзя записывать даже в значении URL-адреса. Эти символы следует заменять соответствующими буквенными или числовыми эквивалентами `<`, `>` и `&` (или `&#60;`, `&#62` и `&#38;`) соответственно.

- Кодировкой файла по умолчанию для HTML-документа является ISO 8859-1, а XHTML-документа — UTF-8.

Ошибки в HTML-коде браузеры пытаются преодолеть тем или иным образом, так что даже весьма небрежно написанный код все же отображается, и во многих случаях вполне удовлетворительно. Более других терпим к ошибкам в HTML-коде и лоялен к разработчику браузер Microsoft Internet Explorer. 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 дружествен 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 смысловых ошибок и при попытках обеспечить инвариантность представления документов.

2.2. Определение типа документа

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). В настоящее время наиболее широко распространен XHTML версии 1.0, хотя имеется и версия 1.1 с более жесткими ограничениями; XHTML 2.0 находится в стадии разработки. В данной книге мы рассмотрим только версию 1.0, в которой предусмотрены три основные схемы XHTML: строгая, переходная и фреймовая.

Тип XHTML-документа задается посредством специального дескриптора `<!DOCTYPE...>`. Наиболее употребительны следующие варианты задания содержимого данного дескриптора:

- **Строгая (Strict) схема** — исключает все нерекомендованные консорциумом W3C теги и атрибуты, предназначавшиеся ранее для внешнего

оформления, такие как `<font>`, `<center>`, `align`, `bgcolor` и др., а также применение фреймов. Вместо них следует применять CSS. Пример:

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

Многие разработчики сайтов, приверженцы стандартов, рекомендуют придерживаться строгой схемы. Если вам более всего важна межбраузерная и межплатформенная совместимость (сейчас или в будущем), следуйте этой схеме.

- **Переходная (Transitional) схема** — включает то же, что и схема Strict, а также все нерекомендованные (устаревшие) теги и атрибуты для обеспечения обратной совместимости. Однако данная схема исключает применение фреймов, т. е. тегов `<frameset>` и `<frame>`.

  Пример:

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

  Данная схема целесообразна при переводе HTML-кодов старых документов на язык XHTML, а также для сохранения совместимости с более старыми версиями как браузеров, так и HTML. Эта схема подойдет также и тем, кто создает свои Web-документы, не разбираясь в стандартах и версиях браузеров и (X)HTML. Однако данная схема не предполагает использования фреймов. Если они вам нужны, то выбирайте фреймовую схему.

- **Фреймовая (Frameset) схема** — включает то же, что и схема Transitional, разрешая при этом работу с фреймами.

  Пример:

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

  Если вы применяете фреймы для компоновки Web-страниц и желаете обеспечить межбраузерную совместимость, то данная схема вам подойдет.

> ***Примечание***
>
> Кроме рассмотренных вариантов XHTML существуют и другие: XHTML Basic (Основной) для миниатюрных устройств, XHTML Mobile Profile (Мобильный профиль) — добавляет к Basic элементы, специфические для мобильных телефонов, XHTML 1.1 Module based (Модульный) — с возможностями расширения.

Итак, мы рассмотрели три основных схемы документов на языке XHTML — три определения типа документа (DTD). В HTML-документе также рекомен-

дуется указывать DTD одной из схем, таких же, что и для XHTML. Вот примеры для HTML 4.01:

- `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Strict//EN" "http://www.w3.org/TR/HTML4.01/strict.dtd">`
- `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" "http://www.w3.org/TR/HTML4.01/loose.dtd">`
- `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN" "http://www.w3.org/TR/HTML4.01/frameset.dtd">`

В настоящее время браузеры будут воспринимать HTML-код и без дескриптора `<!DOCTYPE...>`. В этом случае они будут интерпретировать его так, как умеют.

Объявив с помощью дескриптора `<!DOCTYPE...>`, что документ написан на языке XHTML или HTML с той или иной схемой, следует далее придерживаться синтаксиса и соответствующих ограничений. Необходимо также иметь в виду, что ошибка в дескрипторе `<!DOCTYPE...>` может привести к тому, что браузер (прежде всего это касается Internet Explorer) просто проигнорирует его и будет интерпретировать последующий код как HTML 4.0.

Если вы вполне осознанно использовали дескриптор `<!DOCTYPE...>` для задания типа документа (т. е. DTD), то весьма рекомендуется проверить правильность (валидность, как еще говорят) последующего (X)HTML-кода. В идеале все браузеры должны следовать стандартам, а значит, правильные (валидные) документы должны отображаться в них одинаково. Хотя на практике пока это не совсем так, рекомендуется проверять правильность документа, т. е. его соответствие объявленному типу. Это можно сделать с помощью онлайновой службы валидации консорциума W3C. На сайте **http://validator.w3.org** вы можете ввести URL-адрес своего документа, расположенного на Web-сервере, и получить сведения о его валидности. Даже если все правильно, еще нет полной гарантии межбраузерной и межплатформенной совместимости вашего документа. Тем не менее, это шаг по направлению к идеалу, стремиться к которому необходимо, даже если он достижим не в полной мере и не сию минуту.

Стандарт рекомендует перед DTD, заданным тегом `<!DOCTYPE...>`, записывать еще один дескриптор:

```
<?xml version="1.0" encoding="кодировка"?>
```

Примеры:

```
<?xml version="1.0" encoding="windows-1251"?>
<?xml version="1.0" encoding="utf-8"?>
```

Здесь *кодировка* — указывает кодировку текстового файла документа, в которой он сохранен на сервере.

Однако применение указанной записи для (X)HTML-документов не обязательно, а для некоторых браузеров даже вредно. Так, браузер Internet Explorer 6.0 реагирует на данную запись переходом в режим обеспечения обратной совместимости, а не как указание следовать стандартам.

Итак, в XHTML-коде Web-страницы перед тегом `<html>` должен находиться дескриптор `<!DOCTYPE...>` с описанием типа документа; для HTML-кода применение этого тега если и не обязательно, то настоятельно рекомендуется.

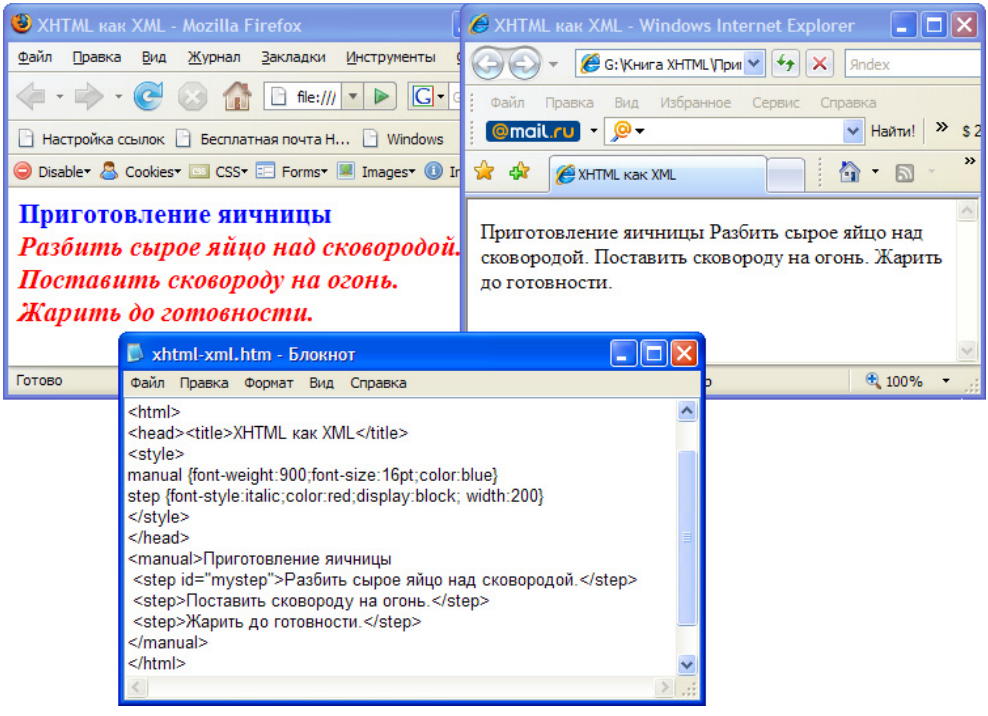

Рис. 2.1. Отображение XHTML-кода с авторскими тегами в Mozilla Firefox и Internet Explorer

Примечание

XHTL-код с соответствующим описанием DTD браузеры Mozilla Firefox 2+ и Opera 9+ интерпретируют так же, как и аналогичный XML-код, чего нельзя сказать, к сожалению, о Microsoft Internet Explorer 7.0 и более ранних его версиях. Иначе говоря, для Firefox и Opera в XHTML-коде допустимы авторские (придуманные, как это делается в XML) теги при условии, что их представление определено в таблице CSS. Таким образом, для этих браузеров XHTML — действительно расширяемый, а не просто синтаксически похожий на XML язык. На рис. 2.1

показаны XHTML-код с авторскими тегами <manual> и <step>, а также его отображение в браузерах Firefox и Internet Explorer. Как видно, Internet Explorer не понимает авторских тегов, а их содержимое выводит на экран как обычный текст, игнорируя CSS. В скором будущем, видимо, все браузеры должны будут обрабатывать XHTML-код по правилам интерпретации XML-кода.

2.3. Структура собственно (X)HTML-кода

Теперь мы рассмотрим структуру собственно (X)HTML-кода, т. е. основные элементы, размещающиеся между дескрипторами <html> и </html>. Контейнерный тег <html> размещается сразу же за дескриптором <!DOCTYPE...>. Он может содержать атрибут lang, значением которого является идентификатор языка текстов вашего документа. Например, <html lang="ru"> для русскоязычных документов, или <html lang="en"> — для англоязычных. Тег <html> может содержать в себе еще несколько контейнеров, как показано на рис. 2.2.

В контейнере <head> должен быть контейнер <title>, в котором указывается название документа, отображаемое в заголовке окна браузера. Если он отсутствует, то браузер Internet Explorer отображает в своем заголовке окна путь к файлу, но другие браузеры могут этого не делать. Кроме того, в <head> помещают служебную информацию, а также те элементы, которые должны быть загружены в браузер прежде содержательной части документа, например, таблицы стилей и сценарии. Видимые элементы документа, относящиеся к его содержанию, в контейнер <head> не помещают. Основная или содержательная часть (тело) документа размещается в контейнере <body>. В приведенном на рис. 2.2 документе ничего содержательного, кроме названия, нет, поэтому клиентская область браузера пуста, а в заголовке его окна отображается название документа — содержимое тега <title>.

Итак, в контейнере <html> располагаются контейнеры <head> и <body>; в <head> вставляют контейнер <title> (обязательный для XHTML-кода и необязательный для HTML-кода), а также, при необходимости, другие теги, содержащие информацию, которая не должна отображаться в клиентской области браузера.

Примечание

Некоторые браузеры (например, Mozilla Firefox 2+ и Opera 9+, но не Internet Explorer 7.0) обрабатывают XHTML-код по правилам XML и допускают авторские теги. Если у вас есть свои собственные теги в различных документах, то может возникнуть коллизия имен тегов и атрибутов. Чтобы ее избежать, следует указывать пространство имен, т. е. URL-адрес файла, содержащего описание тегов и их атрибутов. Например, вы можете указать, что используете пространство имен HTML или же свое собственное пространство, заданное

по правилам описания DTD. В XHTML-документах это делается с помощью атрибута `xmlns` тега `<html>`:

```
<html xmlns="http://www.w3.org/1999/xhtml"/>
```

Подробнее о пространствах имен можно узнать на сайтах **http://www.w3.org/TR/REC-xml-names** (англ.) или **http://www.rol.ru/news/it/helpdesk/xnamsps.htm** (русский перевод).

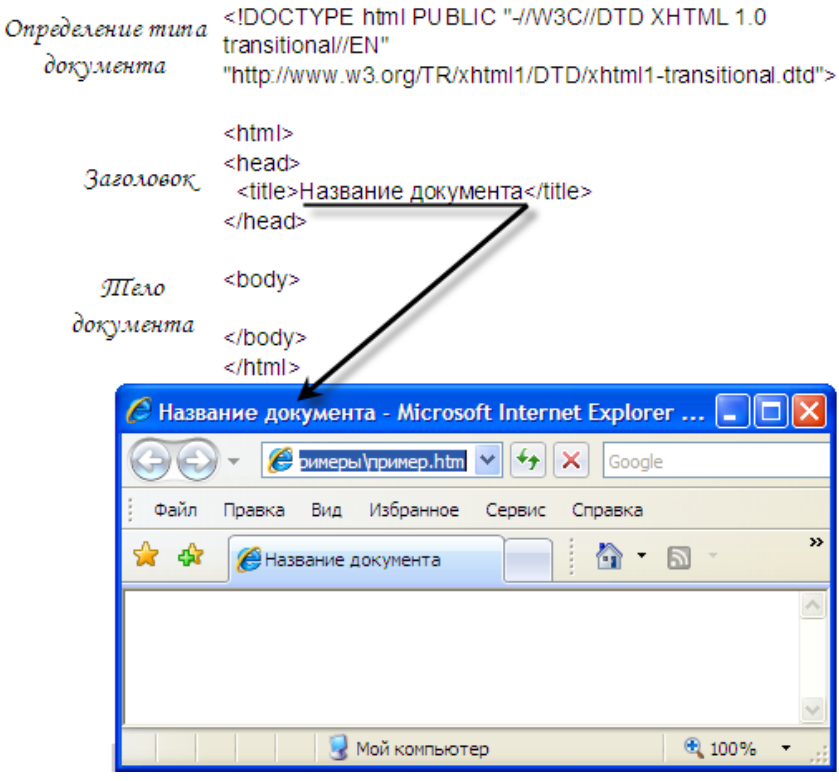

Рис. 2.2. Основные теги (X)HTML-документа

В программы на любых языках часто вставляют комментарии — тексты, которые не интерпретируются, а просто игнорируются исполнительной системой (например, браузером). В (X)HTML для этой цели предусмотрен специальный дескриптор `<!--комментарий -->`. Разумеется, комментарий не отображается, но загружается в браузер вместе с основным кодом и, следовательно, его наличие не способствует ускорению загрузки. Тем не менее, ценность комментариев обнаруживается, когда вам требуется подкорректировать ваш документ, особенно спустя хотя бы несколько недель.