И. Квинт

на 100 %

HTML, XHTML и CSS

Освойте на 100 %:

- структуру HTML-документа
- оформление HTML-документа средствами CSS
- основы языка JavaScript
- язык XHTML

ПИТЕР

Игорь Квинт

HTML, XHTML и CSS на 100%

«Питер»
2010

Квинт И.

HTML, XHTML и CSS на 100% / И. Квинт — «Питер», 2010

Вы хотите создать собственный сайт на просторах Интернета? Причем желательно, чтобы он был красивым, удобным и «неглючным»? Казалось бы, чего проще: существует столько программ – конструкторов сайтов. Однако чтобы создать действительно профессиональный сайт, подобных утилит недостаточно. Вам потребуется знание языков программирования HTML и XHTML, а также каскадных таблиц стилей CSS. И в этом случае книга, которую вы держите в руках, – именно то, что вам необходимо. С ее помощью вы научитесь создавать красиво оформленные, быстрые и профессиональные сайты. С этой книгой ваша страничка никогда не останется незамеченной в Сети!

Содержание

Игорь Квинт
HTML, XHTML и CSS на 100%

Введение

Всемирная паутина стала одним из наиболее значительных достижений XX века. В наше время стремительных перемен такой возраст можно уже считать существенным, но технологии работы в Интернете развиваются до сих пор. В основе всех этих технологий лежит язык HTML (HyperText Markup Language – язык гипертекстовой разметки), переживший несколько этапов развития, которые завершались появлением новых версий.

В настоящее время используется последняя версия языка HTML под названием XHTML (eXtensible HyperText Markup Language – расширяемый язык гипертекстовой разметки), включающая поддержку языка XML (eXtensible Markup Language – расширяемый язык разметки). За исключением поддержки XML обе спецификации языка (HTML и XHTML) в общем аналогичны, поэтому основное внимание в книге уделено языку HTML как наиболее часто используемому средству для создания сайтов. В последней же главе описаны основы языка XHTML, его отличия от HTML, а также требования, которым должен следовать разработчик веб-страниц, чтобы его сайт удовлетворял требованиям языка XHTML.

С помощью собственно HTML можно создавать веб-страницы со статическим (неизменным) содержимым. Однако при первом же путешествии по Всемирной паутине вы можете увидеть, что содержимое сайтов подобными страницами не ограничивается. Посетив любой крупный сайт, сразу можно заметить, что на его страницах есть компоненты, реагирующие на щелчки кнопкой мыши. Такие страницы называются *динамическими.* Для их создания используются небольшие программы, внедренные в HTML-код данной страницы. Эти приложения называются *сценариями* (script). Наиболее популярным языком создания сценариев на сегодняшний момент является JavaScript. В данной книге приведены общие сведения об этом языке, достаточные для написания и отладки небольших сценариев, которые создаются для большинства сайтов. Цель этого издания – научить читателей создавать такие сайты. У вас появятся практические навыки работы с HTML-кодом и JavaScript, необходимые практикующему веб-дизайнеру. Прочитав книгу, вы сможете создавать динамичные сайты профессионального вида и содержания.

Книга начинается с описания структуры документа HTML. В первой главе рассказано, из каких компонентов состоит HTML-код, применяемый для создания любой веб-страницы. Вводятся основные понятия и даются сведения о синтаксических конструкциях языка HTML (элементах и атрибутах), указывается, что такое заголовок и тело документа HTML.

В следующих трех главах описываются методы создания и оформления основных частей документа HTML. Вы познакомитесь со способами ввода текста и его форматирования, настройки внешнего вида таблиц, а также включения в страницу веб-графики и мультимедийной информации (аудио– и видеоклипов). Эти элементы дизайна улучшают восприятие веб-страницы, превращая скучный документ в яркую и красочную витрину вашего сайта, которая не оставит равнодушным ни одного посетителя.

В очередных двух главах описываются *фреймы* и *формы.* Фрейм – это вставленная в HTML-страницу другая HTML-страница. Такая структура очень удобна, когда на веб-страницу нужно поместить несколько различных компонентов, отображаемых поочередно по желанию посетителя. С формами знаком практически каждый, кто пользовался почтовыми сервисами на большинстве сайтов. Формы позволяют поместить на веб-страницу элементы

управления, как в обычном интерфейсе Windows, и использовать их для ввода и отправления писем, данных интерактивных опросов и т. д.

Следующие главы книги посвящены очень важным средствам веб-дизайна – языкам CSS (Cascading Style Sheets – каскадные таблицы стилей) и JavaScript. С помощью JavaScript, как вы уже знаете, создаются динамичные веб-страницы, позволяющие вести интерактивное взаимодействие с посетителем сайта. По сути сценарии JavaScript позволяют превратить веб-страницу в небольшую программу. Вы освоите концепцию языка, его синтаксис и способы внедрения в HTML-код.

Язык CSS – очень эффективный и современный инструмент оформления вебстраниц, напоминающий стилевое форматирование, которое применяется в Word. Вместо того чтобы каждый раз вводить множество параметров форматирования части документа, вы определяете стиль и применяете его там, где необходимо. Язык CSS позволяет сделать это в простой и удобной форме.

Материал книги исчерпывающе объясняет все основные моменты веб-дизайна динамичных и статичных веб-страниц, причем в простой и удобной для усвоения форме – в виде набора пошаговых процедур, реализующих конкретные операции веб-дизайна, с хорошо подобранными и выразительными примерами. Шаг за шагом выполняя эти процедуры, вы станете настоящим мастером веб-дизайна, способным решать любые задачи по созданию профессиональных сайтов.

От главы коллектива авторов

Высказать замечания и пожелания, задать вопросы по этой книге вы можете по адресу alexanderzhadaev@sigmaplus.mcdir.ru или посетив мою домашнюю страничку www.sigmaplus.mcdir.ru. Меня очень интересует ваше мнение о книге, и я обязательно учту все ваши пожелания и замечания в следующих изданиях.

Александр Жадаев

От издательства

Ваши замечания, предложения и вопросы отправляйте по адресу электронной почты gromakovski@minsk.piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На сайте издательства http://www.piter.com вы найдете подробную информацию о наших книгах.

Глава 1
Правила построения HTML-страниц

Для создания веб-страниц часто используется язык гипертекстовой разметки HTML. Конечно, каждый сайт индивидуален, но существуют общие правила построения HTML-страниц. Им обязательно нужно следовать – только в этом случае ваш HTML-код будет верно распознаваться и отображаться браузером.

1.1. Общие понятия HTML

Существуют общие правила записи HTML-документов и принципы, используемые при создании сайтов. Их мы и рассмотрим в этом разделе.

Элемент

Основой языка HTML является *элемент.* Он несет в себе определенную информацию, может описывать документ в целом или способ форматирования текста. Элементы можно назвать основой построения сайта. Все, что вы захотите создать на вашей странице, будет сделано с помощью элементов.

Название элемента помещается в угловые скобки, например <P>. Полученное выражение называется *тегом.* В данном случае это открывающий тег. Иногда необходимо задать парный закрывающий тег, который записывается так: </P>. В основном парные теги используются при форматировании текста, они задают начало и конец блока форматирования. Иногда закрывающий тег вообще не требуется, а иногда его можно пропустить, однако для корректной обработки документа рекомендуется всегда использовать закрывающий тег.

Кратко функции данных тегов можно описать так: открывающий тег включает форматирование, а закрывающий выключает. При этом основным отличием в записи тегов, кроме постановки символа / в закрывающем теге, является отсутствие атрибутов у последнего.

Примером необходимости использования закрывающего тега является работа с элементом P, который обозначает абзац:

```
<P>Текст абзаца</P>
```

Однако и в данном случае закрывающий тег является необязательным, но желательным. Элемент IMG, который добавляет картинку на сайт, наоборот, не требует наличия закрывающего тега. По назначению элемента зачастую можно догадаться, требуется ли ему закрывающий тег.

Элементы применяются для того, чтобы сказать браузеру, какой блок вы хотите видеть в определенном месте страницы, а также какую информацию этот блок должен содержать. Кроме того, браузеру нужно сообщить, как отображать эту информацию. Для этого используют атрибуты элементов.

Атрибут

С помощью *атрибутов* можно указывать различные способы отображения информации, добавляемой с помощью одинаковых элементов, а в некоторых случаях применение элемента без атрибутов не дает результатов. Например, в одном абзаце нужно выровнять текст по левому краю, а в другом – по правому. Чтобы задать выравнивание абзаца, используем атрибут align элемента P:

```
<P align="left">Выравнивание по левому краю</P>
<P align="right">Выравнивание по правому краю</P>
```

Значения атрибутов задаются после знака равенства в кавычках и могут быть разными. Некоторым атрибутам присущ набор фиксированных значений, для других количество значений не ограничено.

Элементы и их атрибуты являются основой языка HTML, но для правильного отображения страницы в браузерах еще важно верно создать структуру документа. Для этого существуют строгие правила. Есть элементы, без которых HTML-документ не может обойтись, потому что именно они определяют его структуру.

1.2. Структура HTML-документа

Для создания структуры документа и хранения служебной информации в нем предусмотрено много элементов, которые охватывают все необходимые пункты построения документа.

Из листинга 1.1 видно, что HTML-документ содержит следующие компоненты:

• строку объявления типа документа;
• декларативный заголовок;
• тело документа.

Листинг 1.1. Описание документа в элементе DOCTYPE

```
<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Transitional//EN»
"http://www.w3.org/TR/html4/loose.dtd">
<HTML>
<HEAD>
<TITLE>Это листинг структуры документа HTML</TITLE>
...Элементы заголовка...
</HEAD>
<BODY>
...Тело документа...
</BODY>
</HTML>
```

Объявление типа документа

В начале каждого HTML-документа следует помещать строку объявления такого рода:

<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Transitional//EN» «http:// www.w3.org/ TR/html4/loose.dtd»>

Эта строка дает браузеру общую информацию об HTML-документе. Несмотря на то что вам вряд ли придется заполнять строку иначе, рассмотрим ее по частям и выясним, какую информацию о документе несут данные из элемента DOCTYPE.

• HTML – показывает, что для создания документа применяется язык HTML.

• PUBLIC – указывает на то, что при создании документа использована общепринятая версия HTML.

• "-//W3C//DTD HTML 4.01 Transitional//EN" – задает публичное имя спецификации языка, применяемого для разметки документа. Если браузер по этому имени сможет распознать, где находятся правила обработки данного документа, он воспользуется ими, иначе сможет загрузить их по ссылке в следующем атрибуте. В данном случае это язык HTML версии 4.01, новейшей на момент написания книги.

• "http://www.w3.org/TR/html4/loose.dtd" – URL-адрес документа, содержащего *наборы определений типа документа,* подготовленного в соответствии с языком HTML 4.01.

Что такое набор правил определения типа документа (Document Type Definition, DTD), мы обсудим в конце книги, когда приступим к освоению языка XHTML. Сейчас же просто запомните, что это сведения, которые необходимы браузеру или другой программе, предназначенной для работы с данным документом HTML, для его правильной обработки. Для документов HTML 4.01 организация W3C определила три набора таких правил DTD.

• Набор *строгих правил DTD,* которые требуют, чтобы данный HTML-документ точно соответствовал всем требованиям спецификации HTML 4.01. Документы с этим набором правил содержат такое объявление:

<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01//EN» «http://www.w3.org/TR/html4/strict.dtd»>

• Набор *переходных (transitional) правил DTD,* которые допускают использование устаревших, не поддерживаемых в версии HTML 4.01 элементов и атрибутов. Документы с этим набором правил содержат такое объявление:

<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Transitional//EN» «http://www.w3.org/ TR/html4/loose.dtd»>

• Набор *правил DTD для документов HTML 4.01 с фреймами.* Что такое фреймы, вы уже знаете: если веб-страница выглядит в окне браузера как набор нескольких окон со своими полосами прокрутки, значит, это и есть фреймовый HTML-документ. Документы такого типа должны содержать следующее объявление:

<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Frameset//EN» «http://www.w3.org/TR/html4/frameset.dtd»>

В этом объявлении содержится общая служебная информация о данном документе HTML 4.01. Если вы пропустите его при создании HTML-документа, браузер, скорее всего, сам сможет догадаться, как ему отобразить соответствующую страницу, но организация W3C настоятельно рекомендует включать объявления DOCTYPE во все разрабатываемые документы.

Может возникнуть вопрос: а зачем существуют строгие и переходные правила DTD и какие из них следует использовать? Ответ связан с историей развития Интернета в целом и языка HTML в частности. Отказ от устаревших средств языка и использование строгих DTD могут привести к тому, что ваш документ HTML 4.01 не будет корректно воспроизводиться отдельными устаревшими программами, поэтому в настоящее время безопаснее использовать переходный набор DTD.

После того как вы ввели общую информацию о странице, нужно разобраться с ее структурой.

Элемент HTML

Корневым элементом документа HTML 4.01 является <HTML>. Это значит, что все остальные элементы содержатся внутри тегов <HTML> и </HTML>. Тем не менее в документах HTML 4.01 этот элемент не является обязательным, хотя W3C рекомендует включать его. В XHTML-документах наличие HTML обязательно.

В элементе HTML могут применяться следующие атрибуты.

• lang – определяет язык документа.

• dir – задает направление чтения на языке документа (RTL – справа налево, LTR – слева направо).

• version – определяет версию стандарта HTML, использованного при составлении документа. Это устаревший атрибут, и его применение не рекомендовано.

• title – определяет всплывающую подсказку для страницы.

В листинге 1.2 представлен пример использования элемента HTML вместе с атрибутами, указывающими на использование русского языка и направление чтения слева направо.

Листинг 1.2. Элемент HTML с атрибутами

```
<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Transitional//EN»
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="ru" dir="LTR" title="Моя страничка">
<–!Содержимое документа–>
</html>
```

На рис. 1.1 представлен результат использования атрибутов элемента HTML.

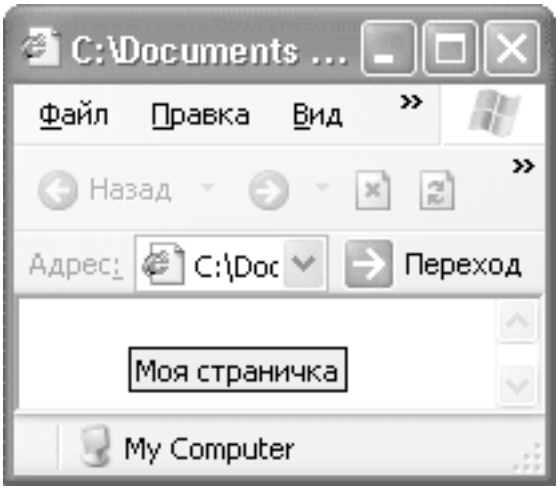

Рис. 1.1. Значение атрибута title отображено в виде экранной подсказки

Следует отметить, что атрибут dir в данном случае не обеспечен достаточной поддержкой известных браузеров даже последних версий. Например, IE 6 и Opera 9 просто выравнивают текст по правому краю, а некоторые устаревшие браузеры этот атрибут просто игнорируют. Так что, если вы намерены использовать текст с альтернативным направлением чтения (иврит, арабский язык), этот атрибут нужно применять с осторожностью.

Элемент HEAD

После того как для документа создана основа, внутри нее нужно создать заглавную область. Информация, вводимая в элемент HEAD, не отображается в окне браузера, а помогает ему в обработке страницы.

В заголовке должны присутствовать как открывающий, так и закрывающий теги <HEAD> и </HEAD>, между которыми располагаются другие элементы, несущие служебную информацию о странице. Элементы, находящиеся внутри элемента HEAD, играют очень важную роль: данные, содержащиеся в них, помогают браузеру в обработке страницы, а поисковым системам – в индексации документа.

Для элемента HEAD определены те же атрибуты, что и для HTML, а также атрибут profil. Он позволяет задать адрес файла с настройками, которые устанавливаются элементами <META> внутри заголовка. С его помощью можно будет избежать многократной записи одних и тех же элементов <META>. Однако этот атрибут пока не введен в действие и рассчитан на будущее развитие языка HTML.

Элементы, которые можно использовать внутри элемента HEAD, представлены в табл. 1.1.

Таблица 1.1. Элементы, используемые внутри элемента HEAD

Элемент	Описание
TITLE	Задает заголовок окна, отображающего документ, требует наличия закрывающего тега. В заголовке может быть только один такой элемент, и его наличие обязательно
META	Определяет различную служебную информацию
SCRIPT	Позволяет добавлять сценарии (см. гл. 12)
LINK	Задает ссылку на таблицы стилей (см. гл. 7–9)
STYLE	Позволяет добавить стили для страницы (см. гл. 7–9)
BASE	Задает базовый адрес документа
OBJECT	Определяет методы обработки нестандартного содержимого (см. гл. 4)

В таблице элементы описаны кратко, дальше мы рассмотрим некоторые из них подробнее. Как можно увидеть из краткого описания в табл. 1.1, они содержат данные, единые для всей страницы.

Начнем с элемента, определяющего заголовок страницы.

Элемент TITLE

Он задает название страницы, которое будет отображаться в строке заголовка окна браузера. Согласно спецификации HTML 4.01 в содержимом элемента HEAD обязательно наличие элемента TITLE, причем в единственном числе.

Элемент требует наличия закрывающего тега </TITLE>. Текст, содержащийся между открывающим и закрывающим тегами, и будет отображаться в строке заголовка окна браузера.

Помимо основной функции – рассказать посетителю, о чем страница, – элемент выполняет несколько косвенных задач. Некоторые поисковые системы используют текст, содержащийся в этом элементе, для поиска и выводят его в качестве заголовка результата поиска, поэтому корректно и качественно составленное заглавие может привлечь посетителей на сайт.

По тексту заголовка пользователь получает дополнительную информацию: что это за сайт и как называется текущая страница. Не стоит думать, что достаточно в документе указать логотип сайта и проигнорировать заголовок. Посетитель может сворачивать окна, и тогда заголовки будут отображаться на кнопках Панели задач – по ним можно будет легко сориентироваться, с каким сайтом работать.

Большинство браузеров поддерживают возможность сохранения веб-страницы на компьютер. В этом случае имя сохраненного файла совпадает с названием заголовка документа. Если в тексте заголовка содержатся символы, недопустимые в имени файла (\ /: *? " < > |), то они будут проигнорированы или заменены другими разрешенными символами.

При сохранении в разделе браузера Избранное в качестве названия ссылки будет использоваться текст, записанный в элементе TITLE. В этом случае адрес текущей страницы с ее заголовком помещается в список ссылок. Поскольку этот список, как правило, хранится в виде отдельных файлов, к их именам также применяется вышеописанное правило.

В листинге 1.3 показан пример использования элемента TITLE.

Листинг 1.3. Использование элемента TITLE

```
<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Transitional//EN»
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Пример создания заголовка страницы</title>
<–!Другое содержимое заголовка–>
</head>
<–!Содержимое документа–>
</html>
```

Название страницы нужно придумывать, логически исходя из ее содержимого.

На рис. 1.2 представлен результат работы листинга 1.3. Видно, что текст заголовка отображается в строке заголовка окна.

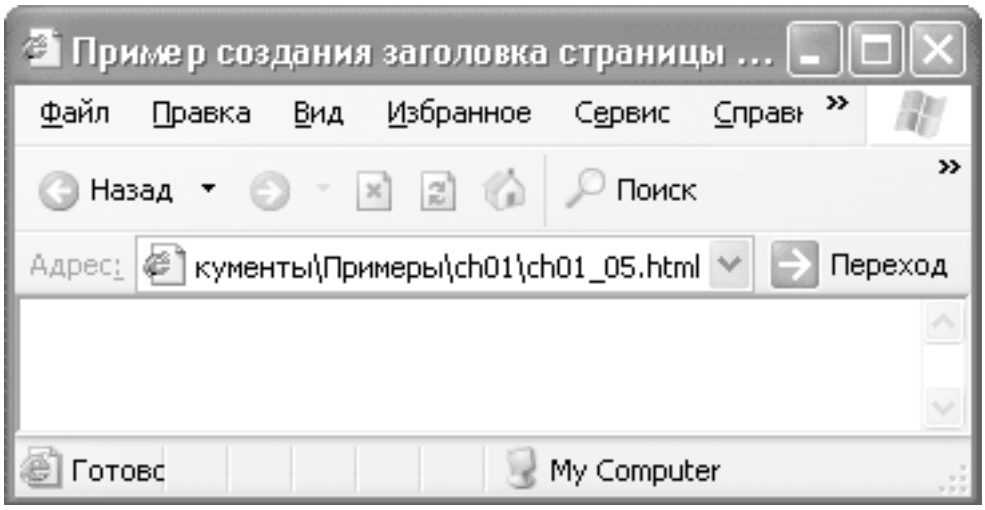

Рис. 1.2. Вид заголовка страницы

Мы определились с заголовком новой страницы, теперь посмотрим, какая служебная информация может содержаться внутри элемента HEAD.

Элемент META

Элемент META используется для хранения дополнительной информации о странице. Эту информацию браузеры применяют для обработки страницы, а поисковые системы – для ее индексации. Например, чтобы указать автора HTML-документа, можно использовать элемент META следующим образом:

```
<META name="Author" content="Вася Пупкин">
```

Здесь значение атрибута name задает имя свойства Author, которому в атрибуте content присваивается имя – «Вася Пупкин». В этом и состоит общее правило применения элементов META: с их помощью вы задаете в атрибуте name имя нового свойства, которому далее в атрибуте content присваиваете значение. Вместо атрибута name можно использовать атрибут http-

equiv, который служит для обмена служебной информацией браузера с веб-сервером. Например, рассмотрим такой элемент:

```
<META http-equiv="Expires" content="Sun, 1 Nov 2009 16:20:47 GMT">
```

Он сообщает браузеру, когда будет исчерпан срок хранения в кэше данной страницы. После этого нужно будет выполнить повторный запрос сервера.

В элементе HEAD может быть несколько элементов META, потому что в зависимости от используемых атрибутов они могут нести разную информацию. В табл. 1.2 представлены возможные значения атрибута http-equiv. Заметьте, спецификация HTML 4.01 не определяет значения этого атрибута, поскольку они устанавливаются протоколом обмена информацией с веб-сервером. Использовать элементы META с такими атрибутами рекомендуется только подготовленным специалистам.

Таблица 1.2. Возможные значения атрибута http-equiv

Значение атрибута	Описание
content-type	Определяет тип содержимого документа. Этот параметр желательно указывать всегда
expires	Задает время действия документа. После даты, указанной для этого свойства, документ будет считаться устаревшим
pragma	Определяет тип кэширования вашего документа, то есть можно запретить браузеру сохранение страницы в кэше
refresh	Позволяет задать параметры автоматической загрузки документа в то же окно браузера, в котором загружен текущий документ

Значение атрибута	Описание
content-language	Задает язык содержимого, аналог атрибута `lang` элемента HTML
content-script-type	Типы сценариев, используемых на сайте
content-style-type	Задает типы таблиц стилей

Рассмотрим подробнее применение описанных выше атрибутов.

В листинге 1.4 приведен пример того, как с помощью атрибута http-equiv задать свойства обработки страницы.

Листинг 1.4. Применение атрибута http-equiv

```
<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Transitional//EN»
"http://www.w3.org/TR/html4/loose.dtd">
<html title="Моя страничка" lang="ru">
<head>
<title>Использование элемента META</title>
<meta http-equiv="Content-Type" content="text/html" charset="windows-1251" />
<meta http-equiv="refresh" content="10" URL="http:\\test.ru" />
<meta http-equiv="pragma" content="no-cache" />
<meta http-equiv="expires" content="Sun, Oct 2010 10:09:00 GMT+3" />
</head>
</html>
```

Как видно из примера, значение атрибута http-equiv указывает на переменную, значение которой определено с помощью атрибута content.

Значение content-type, использованное в примере, будет одинаковым для всех сайтов в кириллической кодировке. Рекомендуем всегда указывать его, иначе браузер может некорректно отображать текст на вашей странице.

Использование параметра refresh полезно, когда страницу перенесли в другое место или когда у нее много адресов. В таком случае вместо того чтобы создавать множество сайтов по разным адресам, можно просто задать возможность переброски посетителя на основной сайт. В примере из листинга 1.4 через 10 секунд после загрузки страницы загрузится сайт test.ru.

Совет

Не все браузеры поддерживают автоматическую пересылку. На всякий случай на странице, с которой идет переадресация, нужно оставлять текстовую ссылку на страницу переадресации.

Рассмотрим возможные значения атрибута content для каждого из представленных свойств (табл. 1.3).

Таблица 1.3. Значения атрибута content для различных значений атрибута http-equiv

Атрибут http-equiv	Значения атрибута content	Описание
content-type	ISO-8859-1	Latin-1
	Windows-1251	Кириллица (Windows)
	KOI8-r	Кириллица (КОИ8-Р)
	cp866	Кириллица (DOS)
	Windows-1252	Западная Европа (Windows)
	Windows-1250	Центральная Европа (Windows)
expires	Sun, Oct 2010 10:09:00 GMT+3	Дата должна указываться в стандарте [RFC850], на английском языке
pragma	no-cache	Не кэшировать. Этот атрибут можно использовать, если страница получается в результате работы сценария, поскольку тогда нет смысла оставлять ее в кэше
refresh	Время в секундах	Время до автоматической перезагрузки страницы
	Время в секундах, URL	Время до автоматической перезагрузки страницы и адрес страницы, которая по истечении этого срока должна быть загружена в текущее окно
content-language	de	Немецкий
	el	Греческий
	en	Английский
	en-GB	Английский, британский
	en-US	Английский, американский
	en-cockney	Английский, кокни
	es	Испанский
	fr	Французский
	it	Итальянский
	i-navajo	Навахо
	ja	Японский
	he	Иврит
	nl	Голландский
	ru	Русский
	pt	Португальский
	zh	Китайский
content-script-type	text/javascript	JavaScript
	text/perlscript	PerlScript
	text/tcl	TCL
	text/vbscript	VBScript
content-style-type	text/css	Язык таблиц стилей CSS

Большинство значений атрибута content, которые вам могут пригодиться, представлены в таблице. Поначалу вы вообще можете ограничиться использованием свойства content-type (оно обязательно), а остальные параметры будете включать при необходимости.

Атрибут name, как и http-equiv, содержит служебную информацию о документе, однако в нем записывается информация другого плана. Например, данные об авторе и содержимом документа. Эти данные не влияют на обработку документа браузером, однако дают информацию для поисковых систем.

В табл. 1.4 представлены возможные значения атрибута name.

Таблица 1.4. Возможные значения атрибута name

Значение атрибута	Описание
keyword	Задает ключевые слова, используемые на странице. По этим словам поисковые системы осуществляют поиск
description	Описание содержимого документа
author	Задает автора документа
document-state	Определяет необходимость переиндексации страницы
resource-type	Задает тип ресурса. Возможные типы мы рассмотрим далее, однако надо учитывать, что если тип документа будет отличным от Document, то поисковые системы не будут его индексировать
revisit	Период переиндексации документа в днях
robots	Указания для роботов и поисковых машин
subject	Задает тематику документа для поисковых машин
URL	Пересылает робота индексации на другую страницу (например, если есть несколько одинаковых сайтов с разными адресами, а хочется, чтобы был проиндексирован только главный)

Как видно из табл. 1.4, большинство свойств отвечает за индексацию страницы в поисковиках. Это очень важный момент, ведь когда вы создаете сайт, то хотите сделать его посещаемым, а в этом помогает элемент META.

Параметры, задаваемые в элементе META, помогают поисковым роботам ассоциировать ваш сайт с определенной тематикой. В результате, когда человек введет в поисковике слово, установленное для вашей страницы в качестве ключевого, среди результатов отобразится ссылка на страницу.

Примечание

Роботы – это специальные программы, которые перемещаются по Интернету и запоминают просмотренные сайты. Результаты поиска заносятся в базы поисковых сайтов, и поиск с самих сайтов осуществляется по этим базам.

У каждого поисковика свои программы-роботы и своя логика поиска и хранения данных в базах, поэтому часто одинаковые запросы в разных поисковых системах дают различные результаты.

В листинге 1.5 приведен пример использования параметров элемента META для управления индексацией страницы.

Листинг 1.5. Параметры элемента META для индексации в поисковиках

```
<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Transitional//EN»
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" title="Моя страничка" lang="ru" xml:lang="ru">
<head>
<title>Использование элемента META</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
<meta http-equiv="pragma" content="no-cache" />
<meta http-equiv="expires" content="Sun, 10 Oct 2010 10:09:00 GMT+3" />
<meta name="author" content="Автор" />
<meta name="description" content="Использование элемента META для индексации сайта в поисковых системах" />
<meta name="document-state" content="Dynamic" />
<meta name="keywords" content="тег, мета, индексация, поиск" lang="ru" />
<meta name="keywords" content="tags, meta, index, search" lang="en" />
<meta name="Resource-Type" content="Document" />
<meta name="Revisit" content="2" />
<meta name="Robots" content="noindex,follow" />
<meta name="URL" content="http:\\test.test" />
</head>
</html>
```

В примере из листинга 1.5 задано много свойств метаданных. Рассмотрим подробнее их действие.

Параметр author задает имя автора. Если владельцем сайта является компания, то вместо author используется параметр copyright, а в качестве значения атрибута content нужно задать имя компании. Можно указать язык, на котором записано значение параметра content. Для этого используется атрибут lang.

Свойство description задает описание документа. Текст, заданный для этого параметра, будет выводиться в качестве описания вашей страницы, отображаемого поисковиком в результатах поиска. Понятно, что этот параметр надо задавать очень аккуратно, потому что именно по этой фразе пользователь будет решать, имеет ли смысл посещение вашей страницы.

Значение dynamic свойства document-state сообщает роботам, что страницу надо будет индексировать при следующем проходе, потому что ее содержимое может поменяться.

Дальше два раза задаются ключевые слова свойства keywords: первый раз на русском языке, второй – на английском. При этом язык, на котором написаны ключевые слова, задается с помощью атрибута lang.

Эффективность сайта напрямую зависит от того, насколько адекватно он отображается в поисковых системах, а это во многом определяется хорошо подобранными ключевыми словами.

В качестве ключевых нужно использовать слова, имеющие прямое отношение к тексту, расположенному на странице. Это значит, что для каждой страницы вашего сайта надо составлять свой список ключевых слов так, чтобы они наилучшим образом отражали ее содержимое.

Не стоит повторять ключевые слова по нескольку раз. Если используете словосочетания, не применяйте в них союзы и предлоги, так как они не несут полезной смысловой нагрузки, а только занимают место. Обычно поисковые роботы распознают первые 200–250 символов, определенных в параметре keyword. Порядок слов зависит от их важности, наиболее важные слова следует располагать в начале списка.

При подборе ключевых слов постарайтесь предположить, какие сочетания и слова будет использовать пользователь при поиске страницы вашей тематики, и используйте в ключевых словах именно их. Для разных страниц сайта, даже если их тематика одинакова, старайтесь

использовать разные сочетания ключевых слов, тогда шанс, что пользователь найдет именно ваш сайт, повысится.

Однако помните, что сейчас поисковые системы ведут поиск не только по ключевым словам – зачастую роботы просматривают всю страницу, чтобы определить ее тематику. Старайтесь создавать страницы, в которых ключевые слова соответствуют содержанию, тогда у вас будет шанс оказаться в первых рядах при поиске.

Параметр resource-type сообщает поисковому роботу тип страницы и применяется для больших сайтов, где много страниц различного назначения. Страница индексируется, только если в качестве значения resource-type задано document.

Рассмотрим параметр revisit. Он говорит, что поисковый робот должен вернуться для переиндексации сайта через два дня. Этот параметр надо задавать, если вы регулярно обновляете содержимое страниц, что поспособствует хранению в поисковых системах актуальной информации о вашем сайте.

Параметр robots дает роботам некоторые управляющие команды. В случае из примера он указывает на то, что текущую страницу индексировать не надо, однако нужно пройти по ссылкам на странице и проиндексировать остальную часть сайта.

В табл. 1.5 представлены команды, которые можно использовать для управления роботами.

Таблица 1.5. Значения атрибута content для свойства robots

Значение	Описание
index	Эта страница должна быть индексирована
noindex	Страница не должна индексироваться
follow	Индексировать страницы по ссылкам с этой
nofollow	Не индексировать страницы по ссылкам с этой
all	Эта страница должна быть индексирована, индексировать страницы по ссылкам с этой
none	Страница не должна индексироваться, не индексировать страницы по ссылкам с этой

Встретив строку URL, робот прекратит индексацию текущей страницы и перейдет на страницу с адресом test.ru.

Из всего, что рассмотрено выше, становится понятна важность элемента META, он отвечает за многие параметры страницы. Далеко не все его возможности будут необходимы постоянно, однако всегда нужно помнить о том, что вы можете сделать с его помощью.

Вы должны знать, что без возможностей элемента META для раскрутки сайта в поисковых системах не обойтись, но они не спасут, если сайт не будет соответствовать ожиданиям посетителей. Поэтому пользуйтесь ими в качестве дополнения к качественно сделанному сайту.

Элемент STYLE

Внутри этого элемента задаются стили, используемые на странице. Таких элементов внутри заголовка страницы может быть несколько. Элемент STYLE поддерживает знакомые вам атрибуты lang и title, а также новые атрибуты type и media. Атрибут type указывает, какой язык задания стилей применяется в данном документе, а атрибут media определяет, на каком устройстве предполагается воспроизводить данный HTML-документ.

Для задания стилей в документе HTML 4.01 применяется язык CSS, который мы будем подробно рассматривать в гл. 7. Здесь же мы немного познакомим вас с возможностями этого языка, чтобы вы поняли их достоинства. Они настолько велики, что консорциум W3C отказался от использования атрибутов форматирования содержимого HTML-документов в элементах, применяемых для разметки текста, например задающих шрифт, его начертание и прочие характеристики. Все эти методы признаны устаревшими, на их смену пришли средства CSS.

Каскадные таблицы стилей (CSS) используют, чтобы максимально отделить HTML-код страницы от ее оформления. Иными словами, внутри таблицы описано оформление различных элементов, а в HTML-коде – только применяемый стиль. Это очень удобный метод – вы можете менять оформление страницы, цвет фона, шрифт, не перебирая огромное количество команд HTML-кода, а просто заменив его в таблице стилей.

Элемент STYLE позволяет реализовать возможности CSS в документе без применения внешних источников. Внутри этого элемента можно записывать код форматирования содержимого странички в формате CSS. Чтобы браузер знал формат кода, атрибуту type элемента STYLE присваивается значение text/css, сообщающее браузеру о применении средств CSS. Вот пример задания CSS-стиля:

```
<style type="text/css">
p.style {
color:#CC0000;
background:#9999CC;
word-spacing:20px;
}
```

Здесь запись p.style указывает, что для элемента P, создающего абзац, определяется стиль под названием style, который определяет для текста внутри абзаца цвет шрифта (color: #CC0000), фона (background: #9999CC) и расстояние между словами (word-spacing:2 0px).

Чтобы подключить к элементу какой-либо стиль, нужно использовать атрибут class и в качестве его значения установить название стиля, который необходимо применить к этому элементу:

```
<p class="style">
Текст со стилем style
</p>
```

Таким образом, используя язык CSS и элемент STYLE, можно создать стили для конкретного элемента или общий стиль, который будет применим к любому элементу, а далее просто ссылаться на этот стиль в процессе разметки документа.

Более того, с помощью элемента STYLE можно задать разные стили для вывода на экран и для вывода на печать. Это полезно, если в качестве фона вы используете темные тона или рисунок. Совершенно не обязательно оставлять подобное декорирование документа при выводе на печать, тем более что это потребует больше краски от принтера. Для этого нужно использовать атрибут media.

Значения атрибута media таковы:

- print – стили для вывода на печать;
- screen – стили для вывода на экран;
- all – стили для вывода на любое устройство.

На примере кода из листинга 1.6 рассмотрим основы синтаксиса CCS.

Листинг 1.6. Использование стилей

```
<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Transitional//EN»
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
<style type="text/css" media="screen" >
p.style {
color:#CC0000;
background:#9999CC;
word-spacing:20px;
}
.style2 {
color:#66FFFF;
background:#990000;
word-spacing:10px;
}
</style>
<style type="text/css" media="print" >
p.style {
color:#000000;
background:#FFFFFF;
word-spacing:20px;
}
.style2 {
color:#000000;
background:#FFFFFF;
word-spacing:10px;
}
</style>
<title>Использование стилей</title>
</head>
<body>
<p class="style">
Текст со стилем style
</p>
<p class="style2">
Текст со стилем style2
</p>
<table border="1" >
<tr>
<td class="style2">
Для ячейки применен стиль style2
</td>
</tr>
</table>
</body>
```

```
</html>
```

В примере создано два стиля: style можно применять только для элемента P (для обозначения этого перед названием стиля стоит название элемента), style2 – для любого объекта. В листинге 1.6 стиль style2 применен еще и для ячейки таблицы.

Результат обработки листинга 1.6 представлен на рис. 1.3.

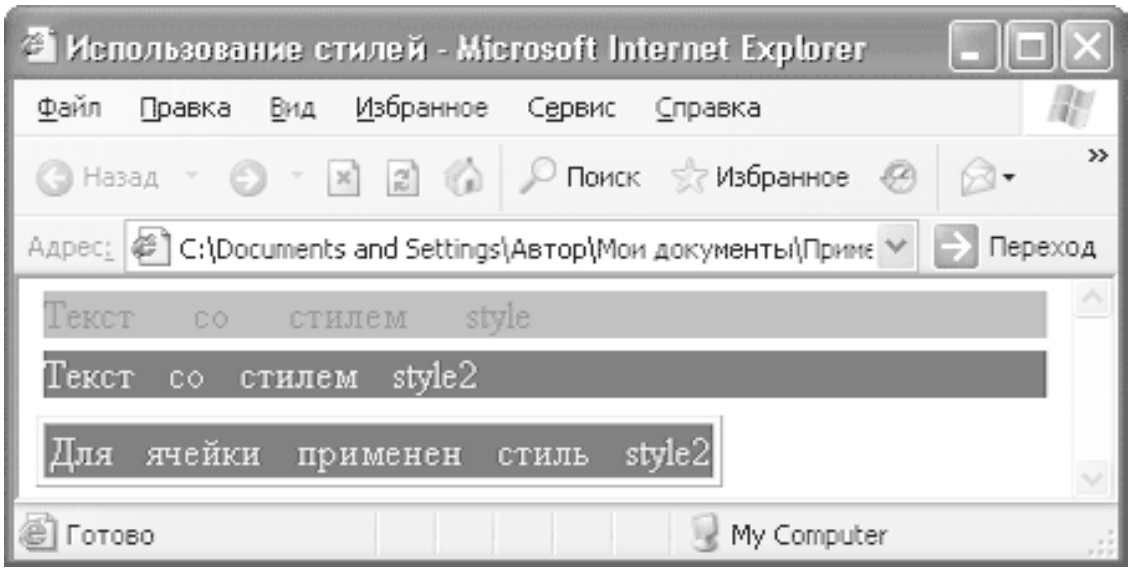

Рис. 1.3. Использование стилей

Для вывода на печать будет использован черный шрифт с белым фоном. Этим мы сэкономим чернила пользователей.

> **Совет**
>
> На странице можно делать ссылки на отображение документа в формате для вывода на печать. Это даст возможность пользователю определить, устраивает ли его установленный формат вывода.

Есть еще один способ применения к элементу стиля: встроить CSS-код непосредственно в элемент разметки в виде значения атрибута style. Такой атрибут доступен для всех элементов HTML. В нем в формате CSS через точку с запятой прописываются значения разных свойств элемента. Вот как это выглядит для элемента P:

```
<p style="color:#CC0000; background:#9999CC; word-spacing:20px;">
```

Как видите, CSS-код совпадает с содержимым элемента STYLE из предыдущего примера. В листинге 1.7 представлен пример задания свойств элементов с помощью встраивания CSS-кода в значение атрибута style.

Листинг 1.7. Использование атрибута style

```
<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Transitional//EN»
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
<title>Использование стилей</title>
```

```
</head>
<body>
<p style="color:#CC0000; background:#9999CC; word-spacing:20px;">
Текст с атрибутом style="color:#CC0000; background:#9999CC; word-spacing:20px;"
</p>
<p style="color:#66FFFF; background:#990000; word-spacing:10px;">
Текст с атрибутом style="color:#66FFFF; background:#990000; word-spacing:10px;"
</p>
<table border="1" >
<tr>
<td style="color:#66FFFF; background:#990000; word-spacing:10px;">
Описание ячейки таблицы с атрибутом style="color:#66FFFF; background: #990000; word-spacing:10px;"
</td>
</tr>
</table>
</body>
</html>
```

Здесь применены такие же стили, как и в предыдущем примере, но параметры заданы с помощью атрибута style.

На рис. 1.4 показан результат обработки кода из листинга 1.7.

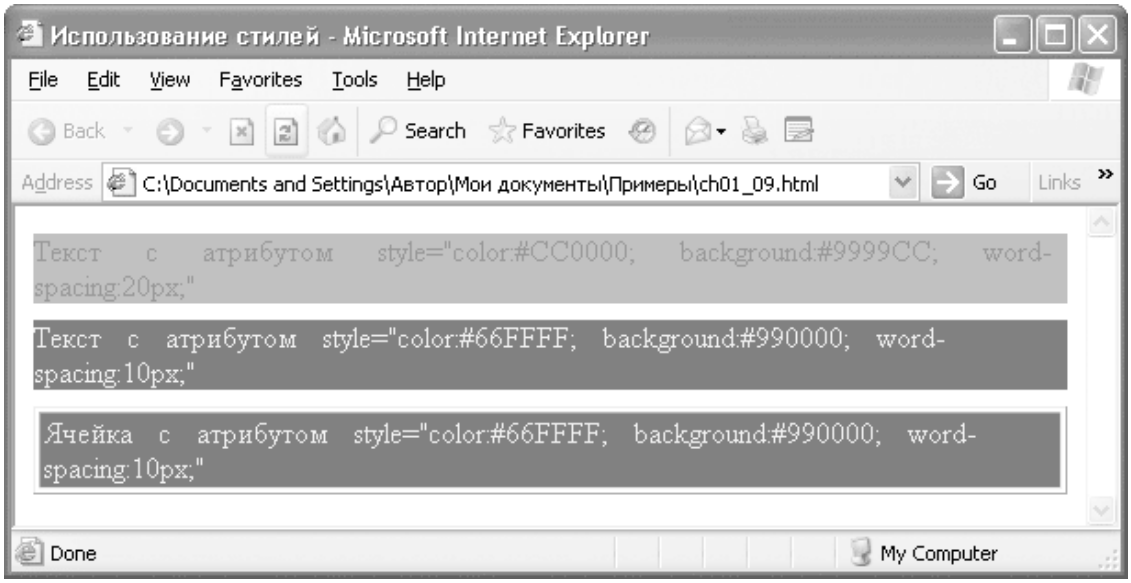

Рис. 1.4. Установка стилей с помощью атрибута style

Как видно из рисунков, результаты обоих листингов одинаковые по стилям. Однако задавать стили в начале документа удобнее, потому что при необходимости их легко будет найти и исправить.

Еще один способ задать стили для документа – записать их в отдельный файл с разрешением CSS. Синтаксис записи в этот файл такой же, как и при использовании элемента STYLE, однако этот способ более универсален.

Подключить файл со стилями к документу можно двумя способами. Первый представлен в листинге 1.8 и использует элемент STYLE. Внутри этого элемента нужно записать такую строку: @import URL("nyTb к файлу с таблицами");.

Листинг 1.8. Импорт файлов с таблицами стилей

```
<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Transitional//EN»
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<style>
@import URL("test.css");
</style>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
<title>Использование стилей</title>
</head>
</html>
```

При такой настройке к элементам нашего документа можно будет применять стили, описанные в файле test.css.

Элемент LINK

Использование элемента LINK – это второй способ подключения файла с таблицами стилей к документу. Элемент не требует наличия закрывающего тега. Его возможности намного шире, нежели просто подключение таблиц стилей. В общем, элемент LINK определяет отношения между текущей страницей и другими документами. Поскольку отношения могут быть разными, то и элементов LINK на странице может быть много.

Основным для этого элемента является атрибут href; его значение – это путь к объекту, для которого описывается тип связи. Этот атрибут является обязательным, что вполне логично – странно было бы устанавливать связи с объектом, не указав, где он расположен.

Другим атрибутом является type, он задает параметры объекта, с которым определяется связь. При связывании таблиц стилей этот атрибут принимает значение text/css.

У элемента LINK есть два взаимодополняющих атрибута: rel и rev. Первый определяет отношение между текущим документом и другим, а второй – отношение другого документа к текущему. Эти атрибуты могут принимать различные фиксированные значения, которые обозначают типы отношений.

В листинге 1.9 представлены примеры использования элемента LINK.

Листинг 1.9. Использование элемента LINK

```
<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Transitional//EN»
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<link href="test.css" rel="stylesheet" type="text/css" />
<link href="copyrihgt.html" rel="copyright" />
<link href="help.html" rev="help" />
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
<title>Использование элемента LINK</title>
</head>
</html>
```

В примере из листинга 1.9 элемент LINK использован для прикрепления к текущему документу таблиц стилей, для указания файла, содержащего информацию об авторском праве на данный документ, и для определения файла, в котором хранится справочная информация.

В табл. 1.6 приведены некоторые возможные значения атрибута rel. За более подробной информацией стоит обратиться к специальной литературе или к спецификации на сайте W3C.

Таблица 1.6. Некоторые значения атрибута rel

Значение	Описание
contents	Ссылка на оглавление данного документа
index	Документ, который можно использовать для индексного поиска по текущему документу
glossary	Словарь терминов данного документа
copyright	Авторские права
next	Следующий документ в очереди просмотра
prev	Предшествующий документ в очереди просмотра
help	Ссылка указывает на документ, предоставляющий справочную информацию

Совет

Использование значения alternate атрибута rel в сочетании с другими атрибутами hreflang и charset, которые задают в элементе LINK язык и кодировку документа, подскажет браузеру, где искать альтернативные версии текущего документа, написанные на других языках.

Мы разобрались с элементом LINK и выяснили его возможности относительно работы с различными документами и в частности с таблицами стилей. Теперь рассмотрим элемент, который пригодится при работе с другими важными объектами создания сайтов.

Элемент SCRIPT

Элемент SCRIPT позволяет присоединять к документу сценарии. Он требует наличия закрывающего тега, при этом текст сценария может располагаться либо в этом элементе, либо во внешнем файле. Если текст сценария расположен во внешнем файле, то он подключается с помощью атрибутов элемента SCRIPT.

Рассмотрим атрибуты этого элемента. Основным можно считать атрибут type, который задает язык сценария, присоединяемого к странице. Если этот атрибут задан, он отменяет установленный по умолчанию язык сценариев. На случай подключения внешнего файла со сценариями у элемента SCRIPT есть атрибут src, в качестве значения которого используют место расположения файла со сценариями. Это удобно, если для различных страниц нужно использовать одинаковые сценарии.

У элемента SCRIPT есть еще один вспомогательный атрибут – defer, который запрещает загружать сценарий до окончания полной загрузки страницы.

В листинге 1.10 показаны разные примеры подключения сценариев к HTML-документу.

Листинг 1.10. Использование элемента SCRIPT

```
<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Transitional//EN»
```

```
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<script defer="defer" type="text/javascript">
<!–текст сценария–>
</script>
<script type="text/javascript" src="test.js">
</script>
<title>Использование элемента SCRIPT</title>
</head>
</html>
```

Количество подключаемых сценариев не ограничено, однако помните, что на их обработку требуется время, так что не стоит загружать браузер больше необходимого.

Элемент BASE

Если внутри документа создаются относительные ссылки на файлы, то может возникнуть ситуация, когда текущий документ перемещен и файлы становятся недоступны. Но в языке HTML есть инструменты для решения этой проблемы.

Элемент BASE служит для того, чтобы задать базовый адрес, относительно которого вычисляются все относительные адреса. Это поможет избежать проблем в случае переноса вашей страницы в другое место. Иначе говоря, все ссылки будут работать, как и прежде.

> **Примечание**
>
> Абсолютный адрес документа использует полный путь к нему, начиная с корневого каталога. Например, C:\Test\test.jpg для файлов на вашем компьютере или http:// www.test.test/mytests/test.html для документов, расположенных в Интернете. Если путь к файлу очень длинный и неудобно каждый раз вводить его, то можно задать относительный адрес. Например, чтобы из документа, находящегося по адресу C:\Test\MyTests\test.html, получить доступ к документу C:\Test\test.jpg, достаточно в нем набрать. \test.jpg, при этом знак «.» означает переход на более высокий уровень иерархии каталогов.

Основным атрибутом элемента BASE является href. В качестве его значения используется адрес базовой папки, относительно которой и будут вычисляться относительные адреса.

Еще одним полезным атрибутом этого тега является target. Он предназначен для работы с фреймами, то есть с HTML-документами, представляющими собой набор окон, каждое из которых является дочерним к основному, родительскому окну документа. Атрибут target определяет, в какое окно будут загружаться страницы по ссылкам, встречающимся в документе. Этот атрибут может принимать четыре значения:

- _top – отменяет все фреймы и загружает страницу в полное окно браузера;
- _blanc – загружает страницу в новое окно;
- _self – загружает страницу в текущее окно;
- _parent – загружает страницу во фрейм-родитель.

При этом можно использовать элемент BASE с указанием только одного из атрибутов.

В листинге 1.11 представлен пример использования элемента BASE.

Листинг 1.11. Использование элемента BASE

```
<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Transitional//EN»
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<base href="http://www.test.test/" target="_blank" />
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
<title>Использование элемента BASE</title>
</head>
</html>
```

В примере из листинга 1.11 все относительные адреса будут браться от адреса http://www.test.test/, то есть если в тексте документа будет строка с адресом ../IMG/test.jpg, то браузер будет ее воспринимать как http://www.test.test/IMG/test.jpg. При этом все ссылки будут открываться в новых окнах, если среди параметров самой ссылки не указать другой вариант.

Мы разобрались со всеми элементами, которые можно использовать внутри элемента HEAD, и от заголовка документа пора перейти к его телу.

Элемент BODY

Внутри элемента BODY располагается сам документ: весь текст, находящийся между открывающим тегом <BODY> и закрывающим тегом </BODY>, будет отображаться браузером. Все элементы, отвечающие за форматирование документа, помещают внутрь элемента BODY.

Атрибуты элемента BODY применяются для того, чтобы установить общие для всего документа свойства, и в этом отношении возможности данного элемента достаточно большие: можно задать цвет ссылок, параметры фона и т. п.

Сначала рассмотрим атрибуты элемента BODY, управляющие отображением ссылок. Для удобства посетителей страницы надо задавать разные цвета для посещенных, непосещенных и активных ссылок, при этом нужно следить, чтобы они не сливались с цветом фона страницы, потому что это будет неудобно посетителям.

За цвет ссылок отвечают следующие атрибуты элемента BODY:

- alink – задает цвет активной ссылки;
- vlink – определяет цвет посещенной ссылки;
- link – устанавливает цвет непосещенной ссылки.

Цвета ссылок можно задать в HEX-формате или ключевыми словами. Ключевые слова имеет смысл использовать, если вы хотите применить стандартный цвет. Ну а если вы собираетесь устанавливать нестандартные цвета, придется воспользоваться HEX-форматом.

В примере из листинга 1.12 представлены оба варианта записи.

Листинг 1.12. Цвет ссылок

```
<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Transitional//EN»
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<base href="http://www.test.test/" target="_blank" />
```

```
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
<title>Использование элемента BODY</title>
</head>
<body alink="#00FF00" vlink="red" link="#330000">
</body>
</html>
```

В примере для активной и непосещенной ссылки цвет задан с помощью цифр и букв: они определяют код цвета в шестнадцатеричном формате и записываются в качестве значения соответствующего атрибута после знака #. Для посещенной ссылки цвет задан ключевым словом. Использовать ключевые слова просто. В качестве обозначения цвета с их помощью применяются стандартные названия цветов на английском языке. Указывайте простые названия цветов вроде blue, red, black и не ошибетесь.

Что касается записи цвета в цифрах, то код для нужного цвета можно определить, используя любой графический редактор.

Элемент BODY предоставляет большие возможности для управления фоном страницы. Рассмотрим атрибуты, отвечающие за это.

Для начала надо задать фоновый цвет страницы. Это делается с помощью атрибута bgcolor, значение которого определяет цвет фона. Потом нужно указать фоновый рисунок страницы с помощью атрибута background, значение которого представляет собой адрес файла с рисунком для фона. При этом старайтесь ставить цвет фона, близкий к цветовой гамме фонового рисунка. Это пригодится, если у пользователя не загрузится фоновый рисунок. Когда цвет фона близок к цвету рисунка, не будет дисбаланса в общем виде страницы.

Следующий атрибут управляет прокруткой фона и называется bgproperties. Он определяет, будет ли фон прокручиваться вместе с текстом. Этот атрибут принимает всего одно значение fixed, позволяющее зафиксировать фон документа и не прокручивать его вместе с текстом. В противоположном случае нужно просто удалить этот атрибут. При выборе параметров прокрутки фона надо действовать очень осторожно и следить за тем, чтобы при прокрутке не терялась читаемость текста.

В листинге 1.13 представлен пример использования атрибутов элемента BODY, отвечающих за фон документа.

Листинг 1.13. Параметры фона

```
<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Transitional//EN»
"http://www.w3.org/TR/html4/loose.dtd">
<head>
<base href="http://www.test.test/" target="_blank" />
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
<title>Использование элемента BODY</title>
</head>
<body background="test.jpg" bgcolor="#0033CC" bgproperties="fixed">
</body>
</html>
```

В примере из листинга 1.13 файл test. jpg оформлен в синих тонах, поэтому и цвет фона взят из синей гаммы. Прокрутка фона вместе с текстом запрещена.

После того как мы разобрались с фоном, можно задать положение содержимого страницы относительно границ окна браузера.

За отступ от границ окна отвечают следующие атрибуты элемента BODY:

• bottommargin – определяет расстояние от нижнего края окна браузера до содержимого страницы;

• leftmargin – указывает расстояние от левого края окна браузера до содержимого страницы;

• topmargin – задает расстояние от верхнего края окна браузера до содержимого страницы.

Расстояние для всех атрибутов устанавливается в пикселах.

В листинге 1.14 приведен пример указания расстояния до содержимого страницы с помощью атрибутов элемента BODY.

Листинг 1.14. Установка расстояния от границ окна до содержимого страницы

```
<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Transitional//EN»
"http://www.w3.org/TR/html4/loose.dtd">
<head>
<base href="http://www.test.test/" target="_blank" />
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
<title>Использование элемента BODY</title>
</head>
<body topmargin="10" leftmargin="20" bottommargin="10">
</body>
</html>
```

В примере расстояние сверху и снизу будет по 10 пикселов, а слева – 20 пикселов.

У элемента BODY есть еще два атрибута, которые могут нам пригодиться. Атрибут text отвечает за цвет текста в документе. Когда задаете цвет текста, следите, чтобы он не сливался с фоном страницы.

Атрибут scroll определяет, будет ли в окне документа вертикальная полоса прокрутки. У него всего два значения: yes и no, обозначающие разрешить и запретить прокрутку соответственно. На самом деле запрещать прокрутку не стоит, потому что документ, который у вас на экране виден полностью без проблем, у пользователя с низким разрешением монитора может не поместиться на экране. В итоге без полосы прокрутки он просто не увидит часть документа.

Мы рассмотрели элемент BODY, его возможности и функции. Все параметры, определяемые в этом элементе, влияют на общий вид документа.

Когда вы создаете большие сайты, то со временем можете забыть, что именно делает та или иная часть кода страницы. Комментарии помогут избежать этого.

Комментарии

Комментарии могут располагаться в любом месте страницы, потому что не отображаются браузером. В качестве текста комментариев можно задавать пояснения к коду страницы. Это удобно, если код большой или с текстом работают несколько человек.

Комментарии заключаются в тег: <! – текст комментария – >. Текст, расположенный внутри этого тега, не будет отображаться.

Резюме

В этой главе вы ознакомились с основами построения HTML-документа и центральными понятиями, используемыми в языке HTML. Большое внимание было уделено метаданным, потому что именно они отвечают за распознавание кодировки страницы и ее индексацию в поисковых системах. Вы также познакомились с основами таблиц стилей и вариантами их подключения к странице.

Большая часть главы была посвящена вспомогательным данным, которые позволят сделать ваш сайт доступным для потенциальных пользователей и дадут возможность браузерам корректно его обрабатывать.

Глава 2
Ввод и оформление текста

Ввод текстовой информации на сайт осуществляется внутри элемента BODY. Однако чаще всего простое расположение текста внутри элемента BODY неприемлемо, необходимо его оформлять, например разделять на абзацы или создавать заголовки. Для оформления текста HTML предоставляет много возможностей: для любого абзаца или заголовка можно задать уникальный внешний вид. Язык HTML также позволяет создавать списки с маркерами любого типа. Можно менять цвет, размер и шрифт текста. В общем, у создателя сайта есть возможность оформить свой текст очень красочно.

Сначала рассмотрим общие принципы структурного форматирования текста, а затем перейдем непосредственно к управлению его внешним видом. При изучении материала этой главы следует помнить, что в новейшей версии языка XHTML форматировать текстовую информацию предлагается с помощью языка CSS, который мы будем изучать в гл. 7–9. Тем не менее содержимое главы весьма важно для практической работы, поскольку вам еще не раз придется столкнуться с многочисленными веб-страницами, оформленными согласно устаревшим средствам, ведь они применялись много лет и на их основе было создано огромное количество документов.

2.1. Создание заголовков

Заголовки – важный элемент сайта, они помогают систематизировать текст. В HTML доступно создание заголовков разных уровней, поэтому очень легко выделять смысловые темы и подтемы. Текст, находящийся в заголовках, влияет на индексацию сайта поисковыми системами, так как многие роботы при поиске обращают внимание на содержимое заголовков, имеющихся на сайте.

В HTML можно создавать заголовки шести уровней. Самым важным считается заголовок первого уровня, а самым малозначимым – шестого.

Заголовки создаются с помощью элементов H1, H2, H3, H4, H5, H6. По умолчанию заголовок самого верхнего уровня выделяется самым крупным шрифтом, и чем ниже уровень заголовка, тем меньше шрифт.

Для заголовка любого уровня можно задать выравнивание по горизонтали. Это делается с помощью атрибута align.

Значения атрибута align:

- left – по левому краю;
- right – по правому краю;
- center – по центру;
- justify – по ширине (только для заголовков длиннее строки).

Для заголовков можно задать атрибут title, значение которого – текст всплывающей подсказки, появляющейся при наведении указателя мыши на заголовок.

В листинге 2.1 представлен пример кода для создания разных типов заголовков и для их различного выравнивания.

Листинг 2.1. Заголовки

```
<html>
<head>
<title>Создание заголовков</title>
</head>
<body>
<h1 align="center">Заголовок 1 уровня</h1>
<h2 align="justify">Заголовок 2 уровня</h2>
<h3 align="left">Заголовок 3 уровня</h3>
<h4 align="right">Заголовок 4 уровня</h4>
<h5>Заголовок 5 уровня</h5>
<h6 title="Заголовок 6 уровня">Заголовок 6 уровня</h6>
</body>
</html>
```

На рис. 2.1 показан результат обработки листинга 2.1 браузером.

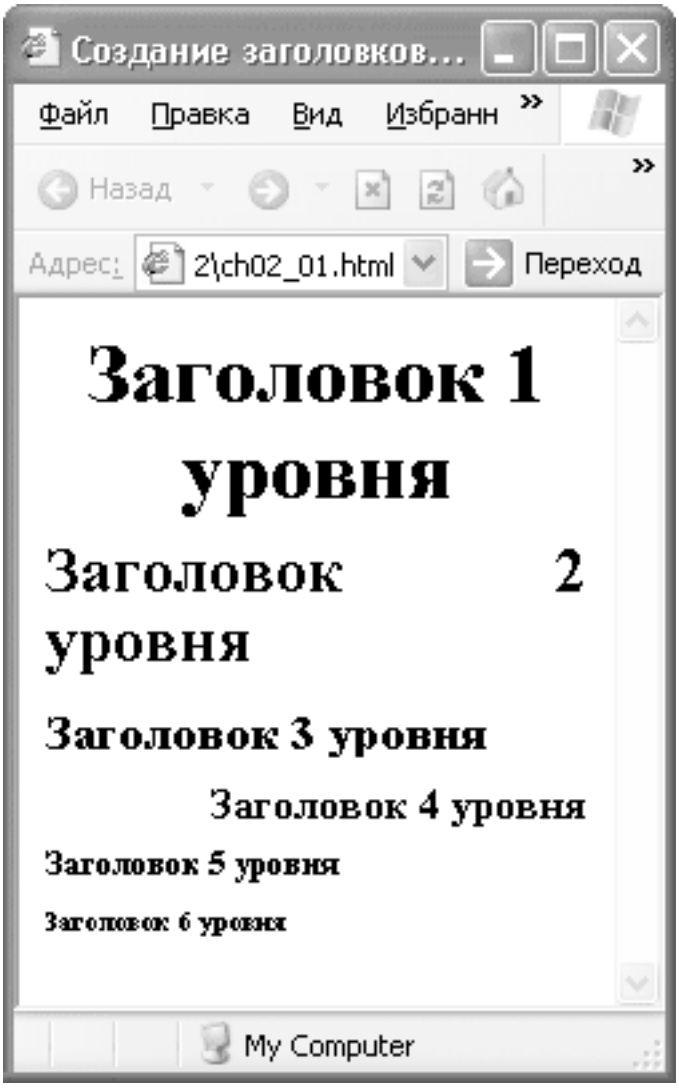

Рис. 2.1. Вид заголовков

На рис. 2.1 видно различие между заголовками разного размера. В принципе, придать тексту подобный вид можно с помощью управления обычным текстом, но, поскольку поисковики воспринимают текст заголовков как важный, имеет смысл для выделения важной информации использовать именно элементы заголовков.

При отображении заголовок всегда начинается с новой строки, а после него всегда идет новая строка – это отделяет заголовок от остального текста.

Для управления другими моментами отображения заголовков, например цветом или шрифтом, используется CSS. Поэтому в элементах H1-H6 разрешено использовать атрибуты style и class, которые подключают стили.

После того как заголовки созданы, можно добавить к ним немного обычного текста, который нужно логически разбить на абзацы.

2.2. Создание абзацев

Абзацы, как известно, делят текст на логические части и на письме выделяются отступом от края листа. В HTML абзацы отделяются друг от друга расстоянием в одну строку.

Для организации абзацев в HTML предусмотрен элемент P, который подразумевает наличие закрывающего тега. Элемент P, наверное, является самым востребованным – наибольшее количество текста, представленного в Интернете, находится внутри элемента P.

Для управления внешним видом текста абзацев в основном используются таблицы стилей, однако небольшое редактирование можно выполнить и с помощью HTML.

Для элемента P можно задать атрибут, определяющий выравнивание. Он называется align и может принимать следующие значения.

• center – выравнивание по центру. При таком виде выравнивания текст прижимается к центру экрана, образуя рваные края. Не стоит использовать подобное выравнивание для текста большого объема, потому что его будет неудобно читать.

• left – выравнивание по левому краю. Текст прижимается к левому краю окна браузера, а справа остаются неровные края. Это самый обычный вид выравнивания, текст такого вида достаточно легок для чтения.

• right – выравнивание по правому краю. Текст прижимается к правому краю экрана, образуя неровные края слева. Такое выравнивание подойдет небольшим эпиграфам. Читать большой текст, выровненный таким образом, будет неудобно.

• justify – выравнивание по ширине. Пробелы между словами автоматически регулируются таким образом, чтобы текст прижимался к левому и правому краям окна. Этот вид выравнивания является наилучшим для больших объемов текста, отсутствие рваных краев слева и справа придает ему аккуратный вид.

Кроме выравнивания, для абзаца можно задать всплывающую подсказку, которая появляется при наведении указателя мыши на текст. Подсказка создается с помощью атрибута title, значением которого является текст подсказки.

В листинге 2.2 приведен пример создания абзацев с разным выравниванием и всплывающими подсказками.

Листинг 2.2. Создание абзацев

```
<html>
<head>
<title>Абзацы</title>
</head>
<body>
<p align="center" title="Абзац, выровненный по центру">Это небольшой заголовочный абзац, выровненный по центру</p>
<p align="right" title="Абзац, выровненный по правому краю">Здесь эпиграф</p>
<p align="left" title="Абзац, выровненный по левому краю">Этот абзац с текстом выровнен по левому краю, поэтому края с правого края будут рваными</p>
<p align="justify" title="Абзац, выровненный по ширине">Этот абзац с основным текстом, выровненный по ширине. Края этого абзаца прилегают к краям окна браузера</p>
</body>
</html>
```

На рис. 2.2 показан результат обработки листинга 2.2, где видно различие между абзацами с разным выравниванием.

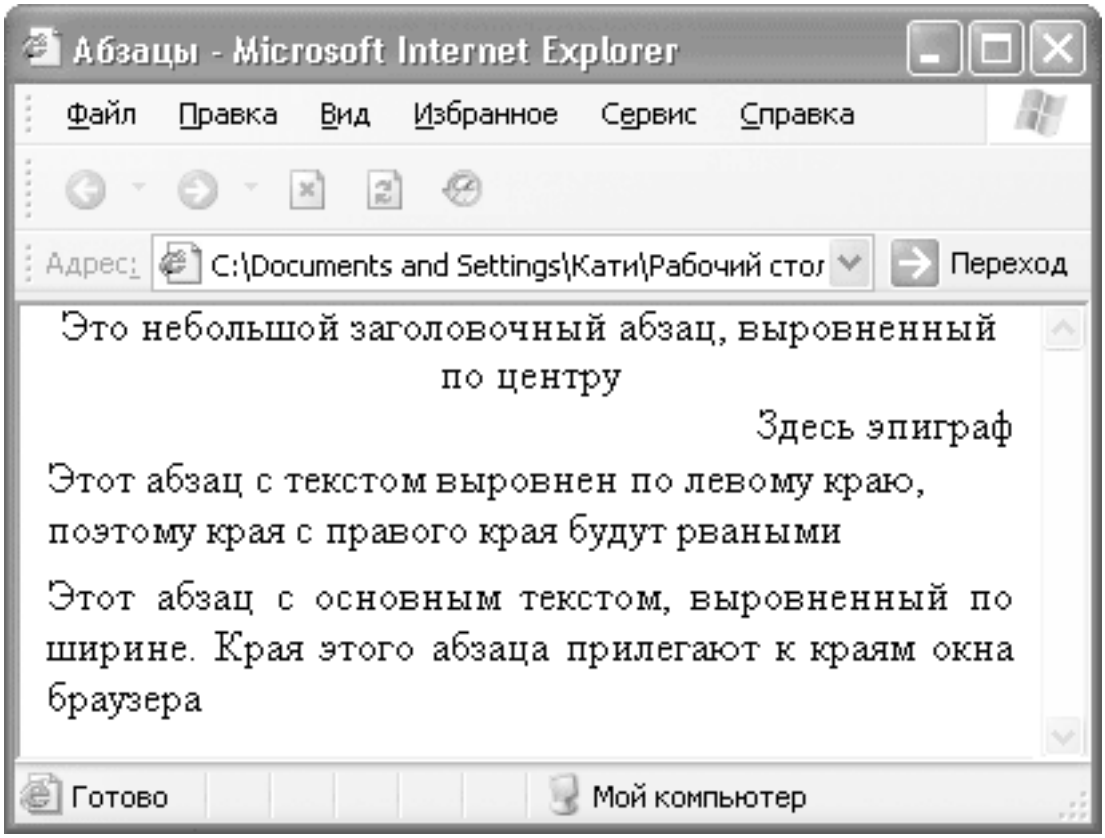

Рис. 2.2. Оформление абзацев

Для дальнейшего оформления текста абзацев используют каскадные таблицы стилей, для этого элемент P допускает использование атрибутов style и class.

2.3. Создание обрывов строк

С оформлением абзаца все понятно. Что же делать, если возникает необходимость оборвать строку, не закрывая абзац, например в том же эпиграфе для записи стихов?

В HTML есть возможность перенести текст на новую строку, не заканчивая абзац. Обычно браузер переносит слова в зависимости от размера окна, и возможность самостоятельно определить место переноса может пригодиться при записи стихов или для отделения различных элементов друг от друга.

Для переноса текста на новую строку служит элемент BR, он не требует закрывающего тега, однако рекомендуется записывать его открывающий тег как <BR />, чтобы все программы отображали его корректно.

В листинге 2.3 представлен пример кода для принудительного переноса строки.

Листинг 2.3. Обрыв строки

```
<html>
<head>
<title>Обрыв строки</title>
</head>
<body>
Наша Таня громко плачет,<br />
Уронила в речку мячик.<br />
</body>
</html>
```

На рис. 2.3 представлен результат обработки браузером кода из листинга 2.3.

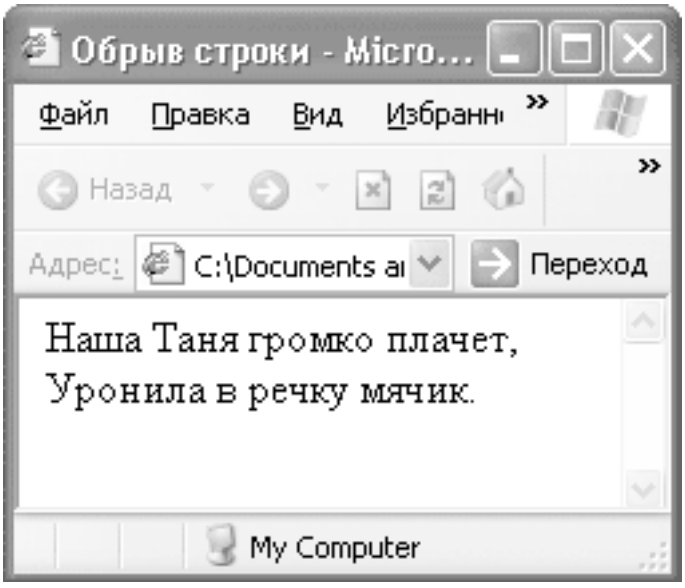

Рис. 2.3. Обрыв строки

Из рис. 2.3 видно, что при использовании элемента BR пустая строка после переноса не добавляется.

Есть еще один вариант применения элемента BR. Его используют, когда нужно задать обтекание текстом таблицы, изображения или любого другого плавающего элемента страницы (то есть элемента, для которого задан атрибут align).

Для этого применяют атрибут clear элемента BR. Атрибут может принимать следующие значения:

• all – запрещает обтекание элемента с двух сторон;

• left – запрещает обтекание с левой стороны плавающего объекта, расположенного после элемента BR;

• right – запрещает обтекание с правой стороны плавающего объекта, расположенного после элемента BR;

• none – отменяет свойство.

Если обтекание запрещено, то текст, следующий за элементом BR, будет отображаться на строке после плавающего объекта.

Кроме обязательного переноса строки, иногда нужно использовать обратное действие, то есть гарантировать, что текст не будет перенесен на новую строку ни в коем случае. Для создания таких неразрывных строк предназначен элемент NOBR, который требует наличия закрывающего тега. Текст, расположенный внутри элемента, будет размещаться в одной строке. При необходимости браузер создаст горизонтальную полосу прокрутки.

Иногда строка может оказаться очень длинной и неудобной для чтения, поэтому внутри элемента NOBR можно использовать элемент WBR, который указывает место для возможного переноса строки.

В листинге 2.4 представлен пример использования элементов NOBR и WBR.

Листинг 2.4. Запрет переноса строки

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
<title>Обрыв строки</title>
</head>
<body>
<nobr>Это очень важная строка, ее ни в коем случае нельзя переносить на другую строку, однако в этом месте <wbr>возможно сделать перенос.</nobr>
</body>
</html>
```

На рис. 2.4 показано отображение в браузере кода из листинга 2.4.

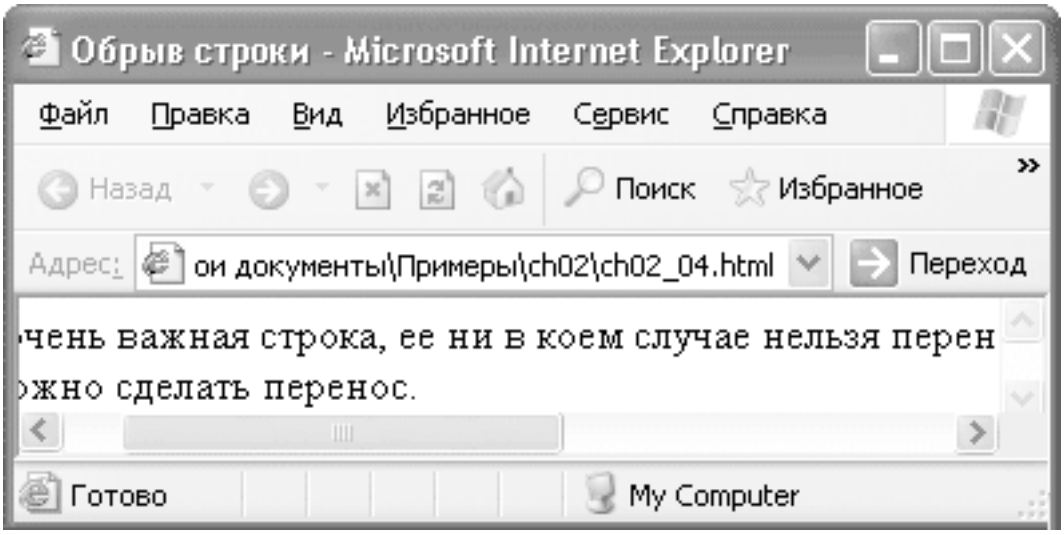

Рис. 2.4. Запрет переносов

Как видно на рис. 2.4, браузер создал горизонтальную полосу прокрутки и перенес строку в том месте, где было разрешено.

Возможность управлять переносами в браузере позволяет контролировать общий вид страницы, но нужно внимательно следить за тем, как браузер организовал переносы.

2.4. Создание списков

Простые списки можно создать с помощью обрывов страниц, но HTML предлагает для этого лучший инструмент.

Списки – важный инструмент, они применяются для организации и группировки данных. Это может пригодиться при создании карты сайта (то есть его оглавления), описания сложных структур и других подобных объектов.

В HTML можно выделить несколько типов списков:

- маркированный;
- нумерованный;
- список определений.

Они отличаются по типам представления информации.

Маркированный список

Маркированные списки – это списки, в которых пункты отмечаются с помощью различных символов. Такие списки еще называют ненумерованными, или неупорядоченными, потому что для элементов данного списка последовательность неважна. Эти списки можно использовать для простого перечисления объектов или их свойств.

Для создания списков в HTML предусмотрен элемент UL, требующий наличия закрывающего тега. Пункты списка находятся внутри элемента UL. Каждый пункт начинается с элемента LI.

У элемента UL есть атрибут type, определяющий вид маркера списка. Он может принимать следующие значения:

- circle – создает маркер в виде круга, белого внутри;
- square – создает маркер в виде квадрата;
- disc – создает маркер в виде круга, закрашенного черным цветом.

Маркер можно выбирать любой, на ваш взгляд наиболее соответствующий виду страницы.

В листинге *2.5* представлен пример создания маркированных списков с разными маркерами.

Листинг 2.5. Маркированные списки

```
<html
<head>
<title>Списки</title>
<body>
<ul type="circle">
<li>Венера</li>
<li>Земля</li>
<li>Юпитер</li>
<li>Марс</li>
</ul>
<ul compact type="square" >
<li>Венера</li>
<li>Земля</li>
<li>Юпитер</li>
```

```
<li>Марс</li>
</ul>
<ul compact type="disc" >
<li>Венера</li>
<li>Земля</li>
<li>Юпитер</li>
<li>Марс</li>
</ul>
</body>
</html>
```

Результат обработки кода из листинга 2.5 показан на рис. 2.5. Здесь видно, как выглядят разные маркеры списков.

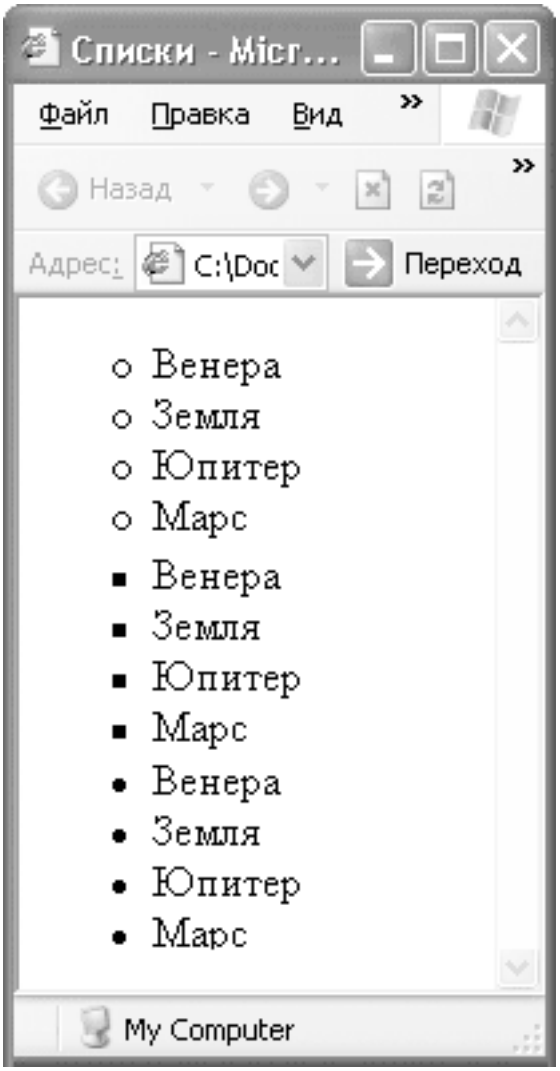

Рис. 2.5. Маркированные списки

При создании маркированных списков с помощью элемента LI можно задать вид маркера отдельно для каждого пункта списка.

В листинге 2.6 приведен пример создания списка с разными маркерами для различных пунктов.

Листинг 2.6. Список с разными маркерами

```
<html>
<head>
<title>Списки</title>
<body>
<ul>
<li type="circle">Земля</li>
<li type="circle">Марс</li>
<li type="square">Солнце</li>
<li type="circle">Венера</li>
<li type="disc">Луна</li>
</ul>
</body>
</html>
```

Результат обработки браузером кода из листинга 2.6 представлен на рис. 2.6.

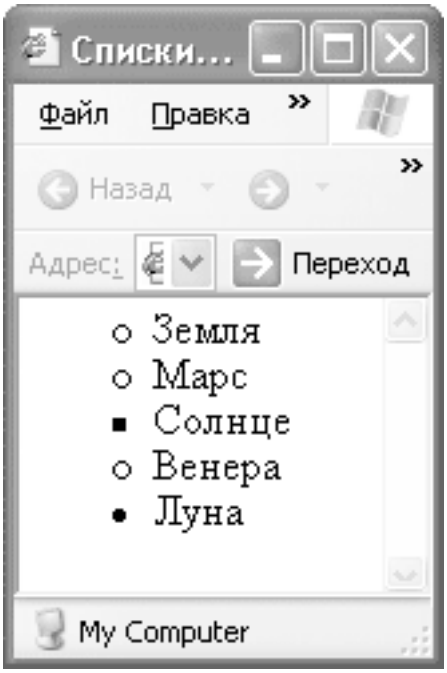

Рис. 2.6. Список с разными маркерами

В примере различные маркеры отмечают объекты разных типов.

Иногда удобнее создавать свои маркеры для списков, в этом случае внешний вид списков будет намного лучше соответствовать стилю вашего сайта и вашим желаниям.

В HTML есть возможность создать список с графическими маркерами. Для этого нужно вместо элемента LI подставить картинку с желаемым изображением. Элемент UL укажет браузеру, что надо сделать отступ, а с помощью элемента BR можно перенести строку.

В листинге 2.7 представлен пример создания списка с графическими маркерами.

Листинг 2.7. Список с графическими маркерами

```
<html>
<head>
<title>Списки</title>
<body>
<ul>
<img src="marker.jpg" />Венера<br />
<img src="marker.jpg" />Марс<br />
<img src="marker.jpg" />Земля<br />
</ul>
</body>
</html>
```

Результат обработки кода из листинга 2.7 показан на рис. 2.7.

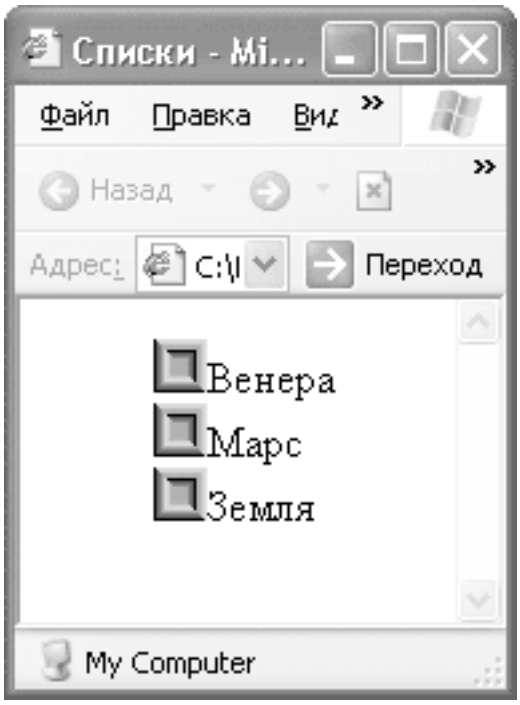

Рис. 2.7. Список с графическими маркерами

Теперь можно использовать картинки для создания симпатичных списков, соответствующих стилю вашего сайта.

Нумерованный список

Однако использование неупорядоченных списков не всегда допустимо. Что делать, если нужно расписать порядок действий, например рецепт приготовления блюда? Для перечисления ингредиентов можно использовать неупорядоченный список, а для описания порядка действий понадобится пронумерованный список.

Нумерованные списки применяются, когда порядок следования пунктов списка имеет большое значение, например при описании алгоритмов или других пошаговых действий. Особенностью списков этого типа является то, что все их элементы упорядочены.

Для создания упорядоченных списков применяется элемент OL, которому требуется наличие закрывающего тега, а все пункты списка находятся внутри этого элемента.

У элемента OL есть атрибут type, который задает формат символов, используемых для нумерации.

Следующие значения атрибута type указывают, что пункты будут нумероваться с помощью:

- A – заглавных букв латинского алфавита;
- a – строчных букв латинского алфавита;
- I – заглавных римских цифр;
- i – строчных римских цифр;
- 1 – арабских цифр.

Вторым атрибутом элемента OL является атрибут start, указывающий, с какого числа начинать нумерацию всего списка.

В листинге 2.8 приведен пример кода для создания упорядоченных списков с разной нумерацией.

Листинг 2.8. Упорядоченные списки

```
<html>
<head>
<title>Списки</title>
<body>
<ol type="1" start="6">
<li>Достать ключ</li>
<li>Вставить ключ в замочную скважину</li>
<li>Повернуть ключ по часовой стрелке на два оборота</li>
<li>Достать ключ из замка</li>
<li>Открыть дверь</li>
</ol>
<ol type="A">
<li>Достать ключ</li>
<li>Вставить ключ в замочную скважину</li>
<li>Повернуть ключ по часовой стрелке на два оборота</li>
<li>Достать ключ из замка</li>
<li>Открыть дверь</li>
</ol>
<ol type="a">
<li>Достать ключ</li>
<li>Вставить ключ в замочную скважину</li>
<li>Повернуть ключ по часовой стрелке на два оборота</li>
<li>Достать ключ из замка</li>
<li>Открыть дверь</li>
</ol>
<ol type="I">
<li>Достать ключ</li>
<li>Вставить ключ в замочную скважину</li>
<li>Повернуть ключ по часовой стрелке на два оборота</li>
<li>Достать ключ из замка</li>
<li>Открыть дверь</li>
```

```
</ol>
<ol type="i">
<li>Достать ключ</li>
<li>Вставить ключ в замочную скважину</li>
<li>Повернуть ключ по часовой стрелке на два оборота</li>
<li>Достать ключ из замка</li>
<li>Открыть дверь</li>
</ol>
</body>
</html>
```

Фрагмент отображения в браузере кода из листинга 2.8 показан на рис. 2.8.

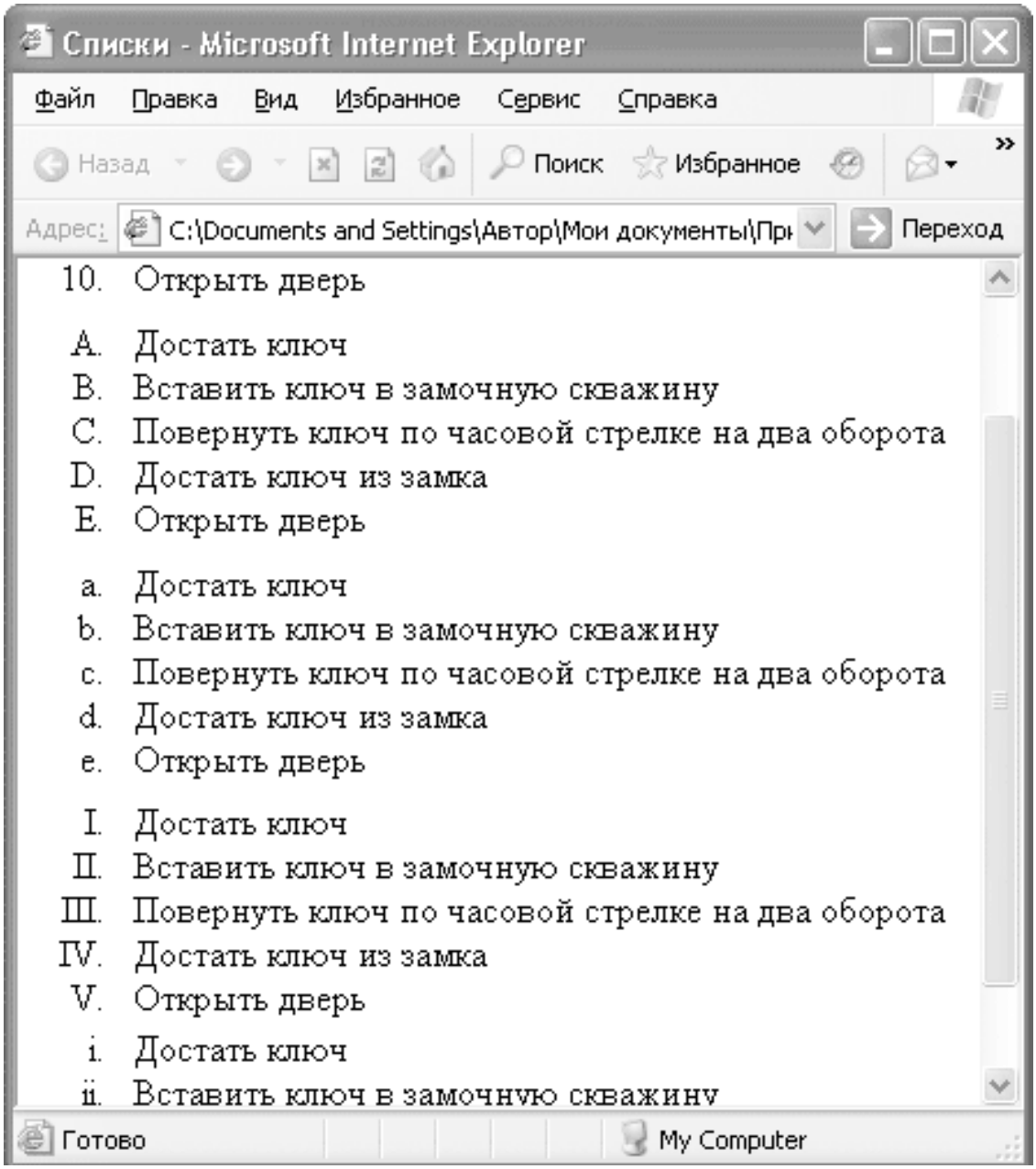

Рис. 2.8. Упорядоченные списки

В примере созданы списки с различными типами нумерации, для списка с арабской нумерацией задан стартовый номер 6.

При создании упорядоченных списков на элемент LI можно возложить дополнительные функции. Как и в примере с маркированными списками, в элементе LI можно задать вид нумерации конкретного пункта с помощью атрибута type. Помимо этого, с помощью атрибута value можно задать номер, с которого будет продолжена нумерация списка.

В примере из листинга 2.9 представлен код для создания списка с разными типами нумерации и различным порядком следования элементов.

Листинг 2.9. Список с разными типами нумерации

```
<html>
<head>
<title>Списки</title>
<body>
<ol type="1" >
<li type="1" value="10">Закрой дверь</li>
<li>Сходи в магазин</li>
<li value="1">Открой дверь</li>
<li>Возьми деньги</li>
<li type="I">Иди налево</li>
<li type="I">Потом поверни направо</li>
<li>Увидишь магазин</li>
</ol>
</body>
</html>
```

Результат обработки листинга 2.9 представлен на рис. 2.9.

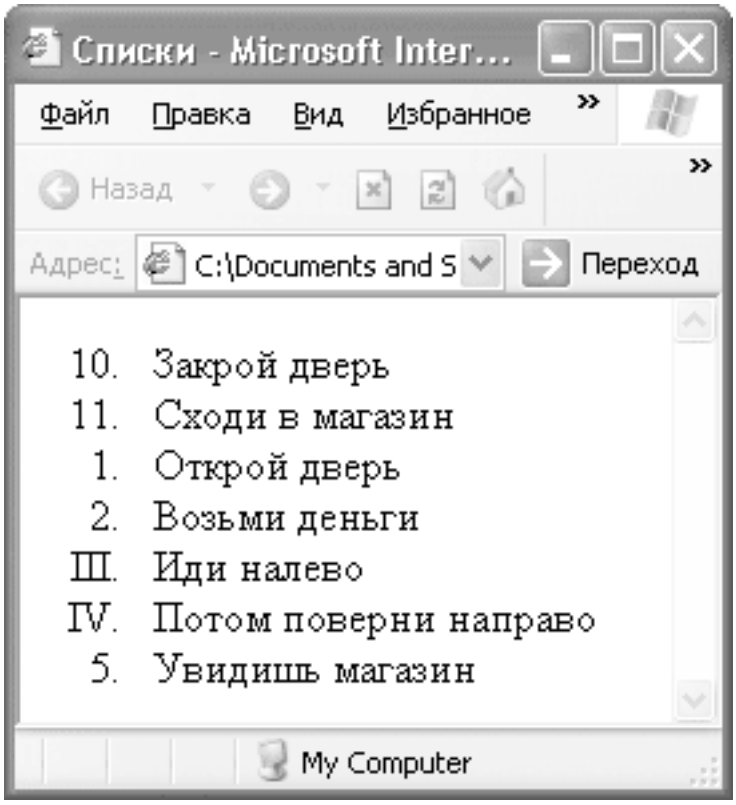

Рис. 2.9. Упорядоченный список с разной нумерацией

Как видно из примера, порядок нумерации и тип ее отображения отделены друг от друга, изменение типа чисел не влияет на числовое обозначение пунктов.

Список определений

Бывает, что на сайте нужно создать список терминов или словарь. Это особенно актуально для сайтов узкой направленности. Для создания подобных конструкций служит список определений.

Список определений – это особый вид списка, который применяется для форматирования словарей или когда необходимо пояснять значения терминов.

Особенность списка определений следует из его функций: элемент такого списка всегда состоит из двух частей. Первая часть задает определяемое слово, а вторая – описание или расшифровку термина. При этом форматирование производится таким образом, что описание термина отображается с отступом от края экрана и, возможно, с пропуском строки от определения.

Для организации списков определений служит элемент DL – внутри него будут находиться определение и описание термина. У этого элемента нет атрибутов, кроме стандартных style и class, с помощью которых к данному определению можно подключить стили.

Чтобы внести информацию внутрь элемента DL, нужно задать элементы DT и DD. Первый используется для того, чтобы задать определение; у него нет никаких особенных атрибутов.

Второй применяется для описания термина из элемента DT. Особых атрибутов в нем также не предусмотрено. Чтобы изменить вид информации, представленной в этом элементе, нужно использовать таблицы стилей.

В листинге 2.10 представлен пример создания списков определений.

Листинг 2.10. Списки определений

```
<html>
<head>
<title>Списки</title>
<body>
<dl>
<dt>Земля</dt>
<dd>Третья планета Солнечной системы</dd>
<dt>Марс</dt>
<dd>Четвертая планета Солнечной системы</dd>
</dl>
</body>
</html>
```

Результат работы кода из листинга 2.10 показан на рис. 2.10.

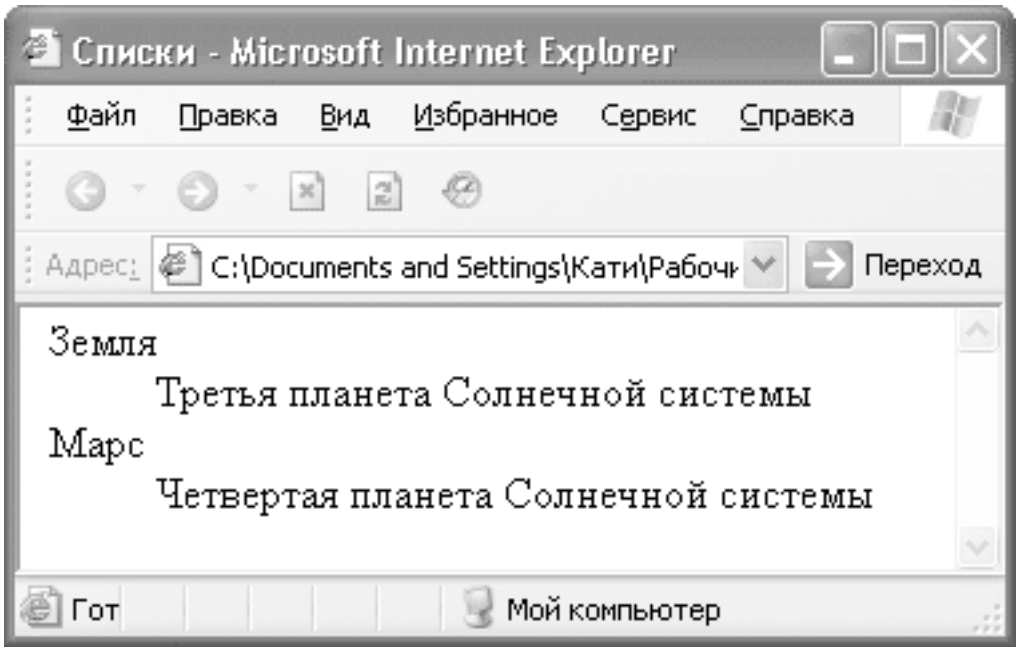

Рис. 2.10. Списки определений

На рис. 2.10 видны особенности форматирования списков определений; информацию в таком виде гораздо проще воспринимать.

Создание вложенных списков

Возможностей простых списков часто не хватает. Например, при создании оглавлений не обойтись без вложенных пунктов. Поэтому рассмотрим создание вложенных списков.

В HTML можно комбинировать и вкладывать друг в друга списки разных типов, но при этом нужно внимательно следить за тем, где заканчивается вложенный список, а где список верхнего уровня, иначе будут проблемы с отображением информации.

В листинге 2.11 представлен код для создания вложенного списка.

Листинг 2.11. Вложенные списки

```
<html>
<head>
<title>Списки</title>
<body>
<ol>
<li>
Открой дверь
<ul>
<li>Ключом</li>
<li>Отмычкой</li>
</ul>
</li>
<li>
Зайди в квартиру
<ul>
<li>Тихо шагая</li>
<li>Громко топая</li>
</ul>
</li>
<li>Ложись спать</li>
</ol>
</body>
</html>
```

На рис. 2.11 можно увидеть, как выглядит вложенный список.

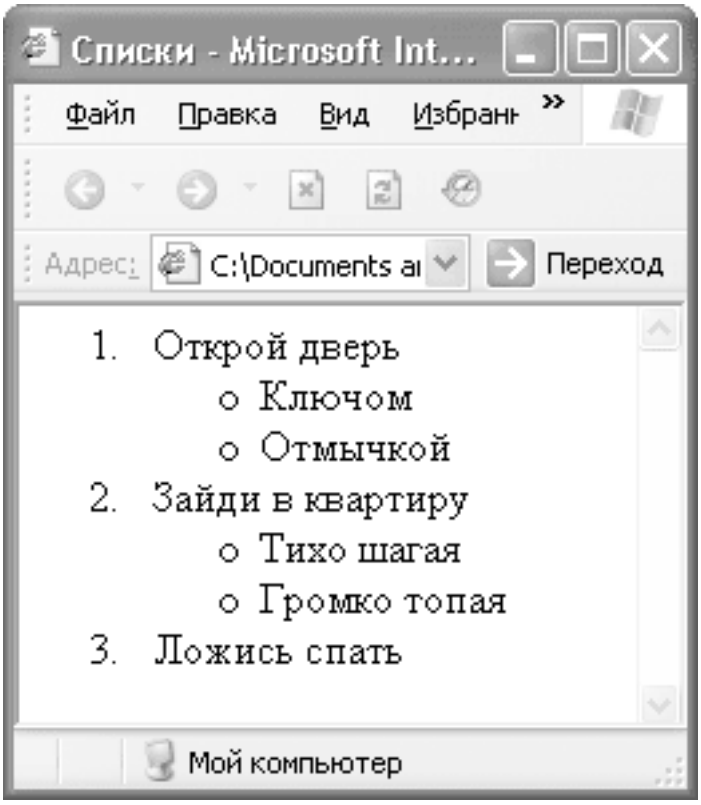

Рис. 2.11. Вложенный список

Видно, что элементы внутренних списков отступают от элементов списка более высокого уровня.

Мы разобрались с большей частью оформления текста, правда, осталось самое главное – то, на чем основан Интернет, – ссылки.

2.5. Ссылки

По своей сути Интернет – это текст и ссылки. Ссылки связывают документы, разбросанные по всему Интернету, в одну сеть. Ваш сайт может находиться на разных компьютерах, но для посетителя он будет казаться единым целым, и все это благодаря ссылкам.

Можно выделить два типа ссылок: внешние и внутренние. Первые связывают страницы в один сайт и помогают передвигаться по нему. Вторые помогают передвигаться в рамках одной страницы.

Внешние ссылки

Внешними называют ссылки на объекты, расположенные вне текущей страницы. Это могут быть картинки, другие страницы сайта, мультимедийные приложения.

Основой внешних ссылок является URL-адрес объекта, на который вы собираетесь сослаться.

Для создания гиперссылок в HTML служит элемент A, который требует наличия закрывающего тега. Внутри элемента располагается текст, который будет выделен как ссылка. На самом деле, чтобы сообщить человеку, что в каком-то месте сайта у вас расположена ссылка, совершенно не обязательно писать прямым текстом адрес следующей страницы. Язык HTML дает возможность «замаскировать» адрес под текст ссылки. Получается, что у вас отдельно есть текст ссылки, который должен внятно описывать, что пользователь увидит, перейдя по ней, и отдельно находится адрес страницы, на которую будет сделан переход при щелчке кнопкой мыши на ссылке. Между тегами <A> и </A> располагается именно текст ссылки.

Адрес документа для перехода записывается в элементе A в качестве значения атрибута href. Таким образом, перемещение по сайту становится удобным, легким и прозрачным для пользователя. Ему абсолютно все равно, где находится документ, на который указывает ссылка, он видит только текст описания.

В листинге 2.12 представлен пример создания ссылки на HTML-страницу и рисунок.

Листинг 2.12. Создание ссылок

```
<html>
<head>
<title>Ссылки</title>
<body>
<a href="aboutme.html">Обо мне</a><br />
<a href="myfoto.jpg">Мое фото</a>
</body>
</html>
```

Пример отображения ссылок в браузере показан на рис. 2.12.

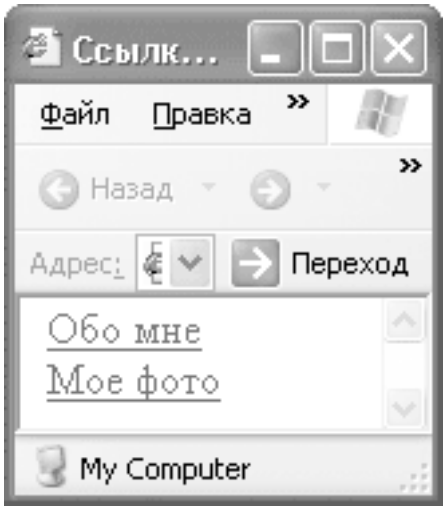

Рис. 2.12. Ссылки

Текст ссылки отображается подчеркнутым, а указатель мыши меняет вид при наведении на ссылку.

Когда посетитель сайта щелкнет кнопкой мыши на ссылке, он перейдет на страницу, которая указана в качестве адреса.

Что же делать, если нужно организовать быстрое перемещение в рамках одной страницы? Здесь тоже помогут ссылки.

Внутренние ссылки

Внутренние ссылки организуют переходы внутри одного HTML-документа. Они применяются, когда на одной странице много текста. Для простоты навигации можно создать ссылки, при щелчке кнопкой мыши на которых пользователь автоматически перейдет к нужной части документа.

Чтобы создать такую ссылку, сначала нужно определить место, к которому ссылка приводит. Это делается с помощью атрибута name элемента A. Необходимый кусок текста заключается в элемент A. Хотя совершенно не обязательно помещать туда текст, можно просто установить теги этого элемента в месте, к которому браузер должен переходить при щелчке кнопкой мыши на ссылке.

В качестве значения атрибута name можно взять любое имя, желательно, чтобы оно характеризовало текущее место, так вам самим будет проще пользоваться метками.

Затем нужно создать ссылку на эту метку. Ссылка на внутреннюю метку создается так же, как и ссылка на внешний документ, только вместо URL-адреса желаемой страницы надо ввести адрес метки в виде #met1. При этом metl – имя вашей метки.

Теперь при щелчке кнопкой мыши на ссылке браузер автоматически перейдет к месту, указанному меткой.

В листинге 2.13 показан пример создания внутренних ссылок.

Листинг 2.13. Внутренние ссылки

```
<html>
<head>
<title>Ссылки</title>
<body>
```

```
<a name="met2">Это метка для перехода 2</a><br />
<a href="#met1">Щелкните кнопкой мыши для перехода к метке 1</a><br />
<a href="#met2">Щелкните кнопкой мыши для перехода к метке 2</a><br />
<a name="met1">Это метка для перехода 1</a>
</body>
</html>
```

Представление внутренних ссылок в браузере показано на рис. 2.13.

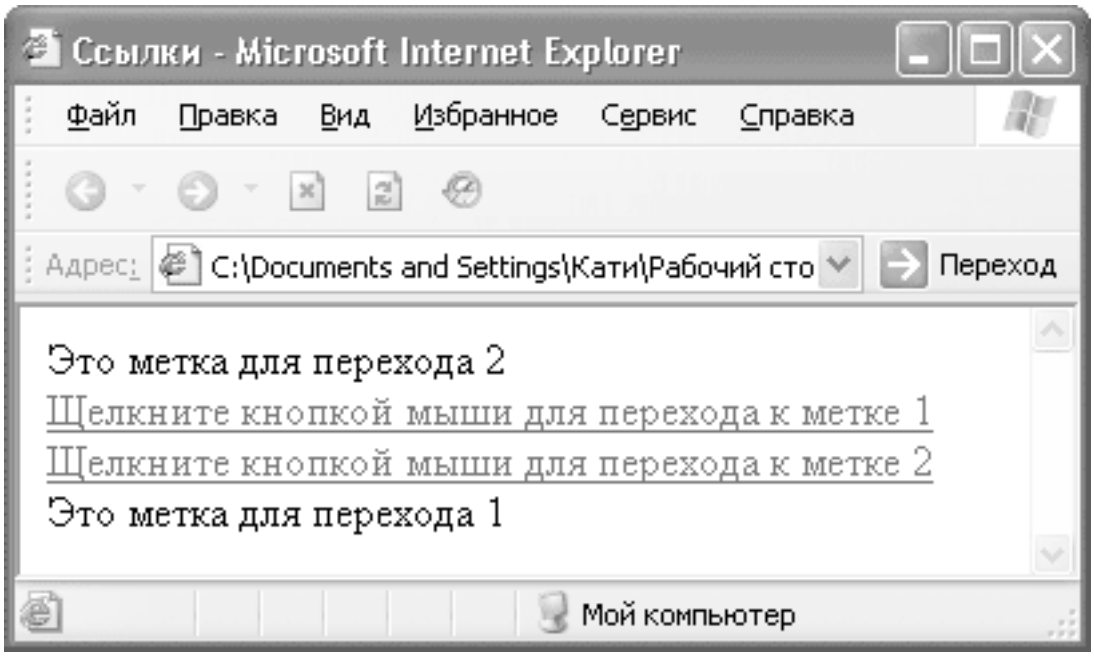

Рис. 2.13. Внутренние ссылки

Как видно, внутренние ссылки при отображении ничем не отличаются от внешних, а текст, отмеченный как метка, никак не выделяется.

Внутренние ссылки по своему синтаксису такие же, как и внешние, поэтому атрибуты элемента A применимы для обоих типов.

Общие моменты

Все немногочисленные атрибуты элемента A можно применять при создании как внутренних, так и внешних ссылок, их действие в зависимости от этого не меняется.

У элемента A есть два вспомогательных атрибута. Первый – target – указывает на то, в каком окне должен открываться документ, отображающийся при выборе ссылки.

Следующие значения атрибута target указывают, что страница загружается:

- _blanc – в новое окно браузера;
- _parent – во фрейм-родитель;
- _self – в текущее окно;
- _top – в полное окно браузера.

Этот атрибут очень полезен, потому что иногда бывает удобно открыть ссылку в новом окне. Например, когда ссылка уводит на другой сайт, а вы не хотите, чтобы пользователь забыл о вашем. Открытие страницы в новом окне – гарантия того, что пользователь снова взглянет на вашу страницу.

Вторым вспомогательным атрибутом является title, он позволяет создать всплывающую подсказку для вашей ссылки, что бывает очень удобно и дает пользователю дополнительную

информацию о том, что его ждет под ссылкой. Значение атрибута – текст всплывающей подсказки.

В листинге 2.14 представлен пример создания ссылок с рассмотренными выше атрибутами.

Листинг 2.14. Ссылки с дополнительными атрибутами

```
<html>
<head>
<title>Ссылки</title>
<body>
<a href="aboutme.html" target="_blank" title="Перейдя по ссылке, вы сможете узнать обо мне">Обо мне</a>
</body>
</html>
```

На рис. 2.14 показан пример отображения ссылки со всплывающей подсказкой.

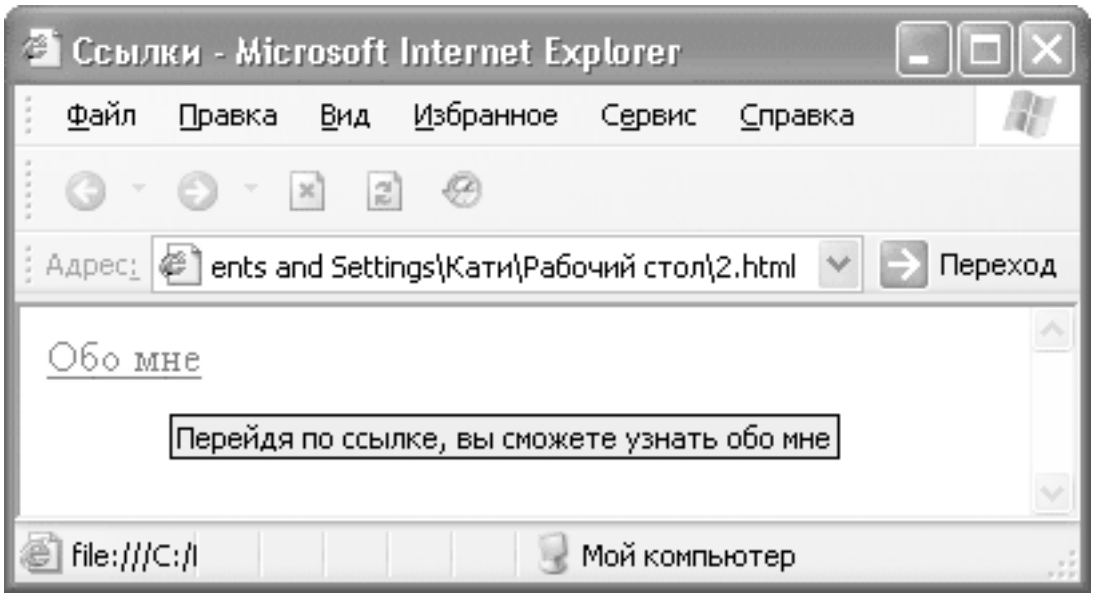

Рис. 2.14. Ссылки с дополнительными атрибутами

Видно, что всплывающая подсказка может помочь посетителю страницы.

Помимо этого, любой тип ссылки можно реализовать в виде изображения, то есть новая страница будет открываться при щелчке кнопкой мыши на рисунке. Можно, например, создать миниатюрные копии фотографий, при щелчке кнопкой мыши на которых будет открываться фото большего размера.

Для создания ссылки-изображения достаточно поместить рисунок внутри элемента A.

В листинге 2.15 показан пример создания ссылки-изображения.

Листинг 2.15. Ссылки-изображения

```
<html>
<head>
<title>Ссылки</title>
```

```
<body>
<a href="mypetfoto.html" target="_blank" title="Пройдя по ссылке, вы сможете увидеть фото моего зверя"><img src="mypetfotomini.jpg" /></a>
</body>
</html>
```

На рис. 2.15 показано, как браузер отображает ссылки-изображения.

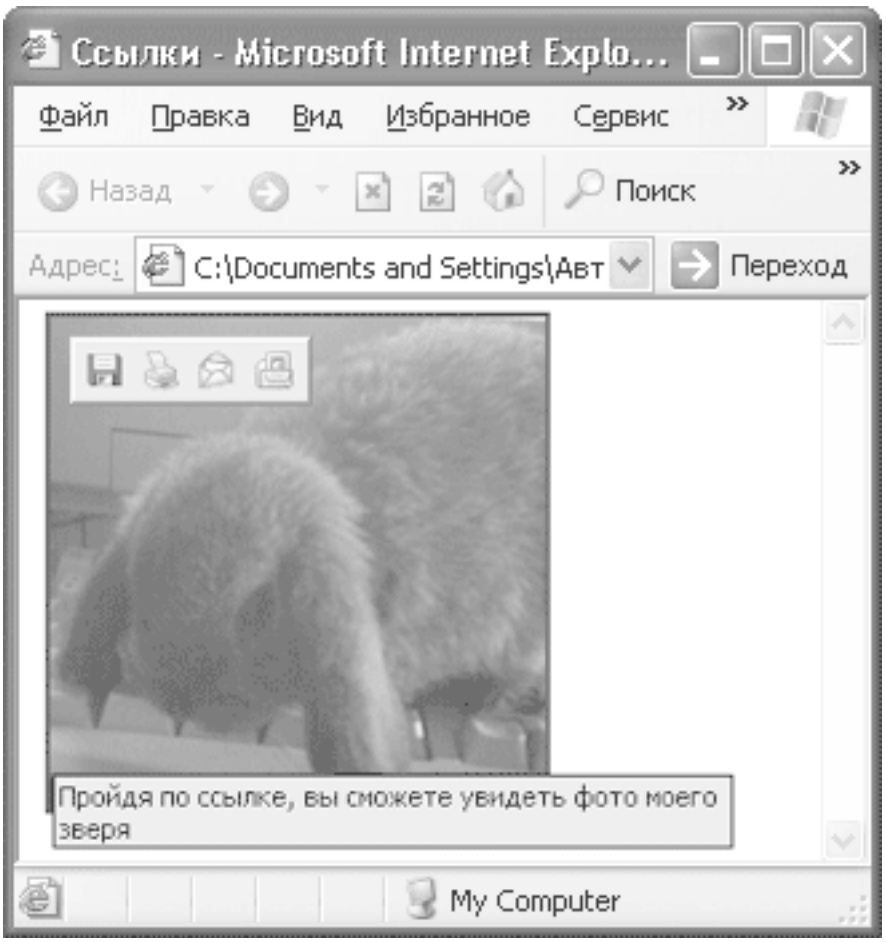

Рис. 2.15. Ссылка-изображение

Вокруг рисунка создается рамка, которая помогает посетителю понять, что перед ним не просто изображение, а ссылка.

Теперь вы можете создать ссылку любого вида на любой объект. Следите за тем, чтобы ссылки имели понятные названия, соответствующие содержанию документов, расположенных за ними, тогда посетители сайта будут вашими постоянными гостями.

Теперь рассмотрим, как можно влиять на внешний вид текста на странице.

2.6. Форматирование текста

Для чего форматировать текст? Странный вопрос, ведь при создании сайта хочется, чтобы он был выдержан в одном стиле, а цвет и вид текста, принятые по умолчанию, не отвечают стилистике большинства сайтов. Поэтому приходится форматировать текст самостоятельно, благо HTML предоставляет для этого огромные возможности.

Есть много элементов для форматирования текста, и все они делятся на две группы: логические и физические. Друг от друга группы отличаются принципиально.

Логические элементы сообщают браузеру о том, какой тип информации в них содержится, например важный текст или цитата. Браузер сам решает, как отобразить такой текст. Конечно, есть принятые стандарты для отображения определенных элементов, но в таких элементах главное – смысл. По сути они разбивают документ на логические части и при этом не обязывают браузер отображать текст, расположенный внутри элемента, каким-либо конкретным образом.

Физические элементы просто говорят браузеру, как должен выглядеть тот или иной блок текста, не уточняя никак смысл и значимость его содержимого. Другими словами, такой элемент заставляет браузер нарисовать букву красной, полужирной или курсивом, не уточняя, почему буква должна выглядеть именно так.

В некотором роде действие на внешний вид текста у многих элементов одинаково: для выделения текста полужирным шрифтом, например, можно использовать как логические, так и физические элементы.

Начнем рассмотрение способов форматирования текста с использования логических элементов.

Логические элементы для форматирования

Как уже говорилось, логические элементы для форматирования определяют не внешний вид текста, а его тип, и в зависимости от которого браузер применяет тот или иной вид внешнего форматирования. Все элементы, рассматриваемые ниже, являются контейнерами и требуют наличия закрывающего тега.

Некоторые из этих элементов могут вообще не изменять отображение текста, поэтому при их рассмотрении будем делать упор на то, как они определяют значение текста, а не на то, как они его форматируют.

Элемент ABBR

Элемент ABBR определяет текст как аббревиатуру. С помощью атрибута title можно задать всплывающую подсказку с расшифровкой аббревиатуры. При этом поисковые роботы индексируют именно полный вариант расшифровки, определенный в атрибуте title.

Пример:

```
<abbr title="Научно-исследовательский институт">НИИ</abbr>
```

Элемент ACRONYM

Элемент ACRONYM указывает, что текст является акронимом.

Примечание

Акронимы – это некие устоявшиеся сокращения, например СНГ, США и т. п.

Атрибут title позволяет задать расшифровку акронима. Пример:
<acronym>СНГ</acronym>

Элемент CITE

Элемент CITE отмечает небольшую цитату или сноску, взятую из другого источника. Такой текст обычно отображается курсивом.

Пример:
<cite>Здесь указан источник информации</cite>

Элемент CODE

Этот элемент указывает на программный код, который может содержать, например, переменные, функции, небольшие куски программы. Такой текст обычно выводится моноширинным шрифтом.

Пример:
Зададим функцию <code> func(int a, char b);</code>

Элемент DEL

Элемент DEL помечает текст как удаленный и может использоваться при внесении изменений в документы. У этого элемента есть два атрибута: cite должен содержать URL документа, в котором описаны причины удаления фрагмента, а datetime – дату и время удаления фрагмента в формате ГГГГ-ММ-ДДТчч: мм: ссTZD (аббревиатура от Time Zone region with Daylight Saving Time – регион часового пояса с летним временем). Браузеры такой текст отображают как зачеркнутый.

Пример:
<del cite="whydel.html" datetime="2007-10-06T10:11:45+3.00"> Неактуальный фрагмент</del>

Элемент DFN

Этот элемент выделяет текст как определение. Элемент можно использовать, если новый термин встречается в тексте впервые и тут же дается его определение. Браузер отображает такой текст курсивом.

Пример:
<dfn>Определение</dfn>-описание

Элемент EM

Элемент EM выделяет важные фрагменты текста. Браузер отображает такой текст курсивом.

Пример:
Важно

Элемент INS

Элемент INS отмечает текст как вставку и применяется для выделения изменений, вносимых в документ. У этого элемента есть два атрибута: cite должен содержать URL документа, в котором описаны причины добавления фрагмента, а datetime должен содержать дату и время добавления в формате ГГГГ-ММ-ДДТчч: мм: ссTZD. Браузеры отображают такой текст как подчеркнутый.

Пример:

<ins cite="whyadd.html" datetime="2007-10-06T10:11:45+3.00">Новый фрагмент</ins>

Элемент KBD

Элементом KBD помечают текст, вводимый пользователем с клавиатуры. Браузеры отображают такой текст моноширинным шрифтом.

Пример:

Введите слово <kbd>дом</kbd>

Элемент Q

Этот элемент обозначает текст как цитату и применяется для добавления коротких высказываний в текст. Обычно отображается как курсив, но некоторые браузеры берут в кавычки текст, отмеченный этим элементом.

Пример:

Цитата: <q>Как сказал поэт</q>

Элемент SAMP

Элемент SAMP определяет текст как пример и обычно используется для выделения результатов работы программы. Браузер выделяет этот текст моноширинным шрифтом.

Пример:

<samp>Образец</samp>

Элемент STRONG

Элемент STRONG предназначен для постановки акцента на тексте. Браузеры выделяют такой текст полужирным шрифтом.

Пример:

Очень важный фрагмент

Элемент VAR

Этот элемент применяется для выделения переменных из программ. Браузер отмечает такой текст курсивом.

Пример:

Введите переменную <var>X</var>

В листинге 2.16 показан код страницы с различным форматированием текста.

Листинг 2.16. Логическое форматирование текста

```
<html>
<head>
<title>Логические элементы для форматирования</title>
<body>
<abbr title="Научно-исследовательский институт">НИИ</abbr><br />
2.6. Форматирование текста
<acronym>СНГ</acronym><br />
<cite>Здесь указан источник информации</cite><br />
Зададим функцию <code>func(int a, char b);</code><br />
<del cite="whydel.html" datetime="2007-10-06T10:11:45+3.00">Неактуальный фрагмент</del><br />
<dfn>Определение</dfn>-описание<br />
<em>Важно</em><br />
<ins cite="whyadd.html" datetime="2007-10-06T10:11:45+3.00">Новый фрагмент</ins><br />
Введите слово <kbd>дом</kbd><br />
Цитата: <q>Как сказал поэт</q><br />
<samp>Образец</samp></br>
<strong>Очень важный фрагмент</strong><br />
Введите переменную <var>X</var><br />
</body>
</html>
```

Результат обработки браузером кода из листинга 2.16 показан на рис. 2.16.

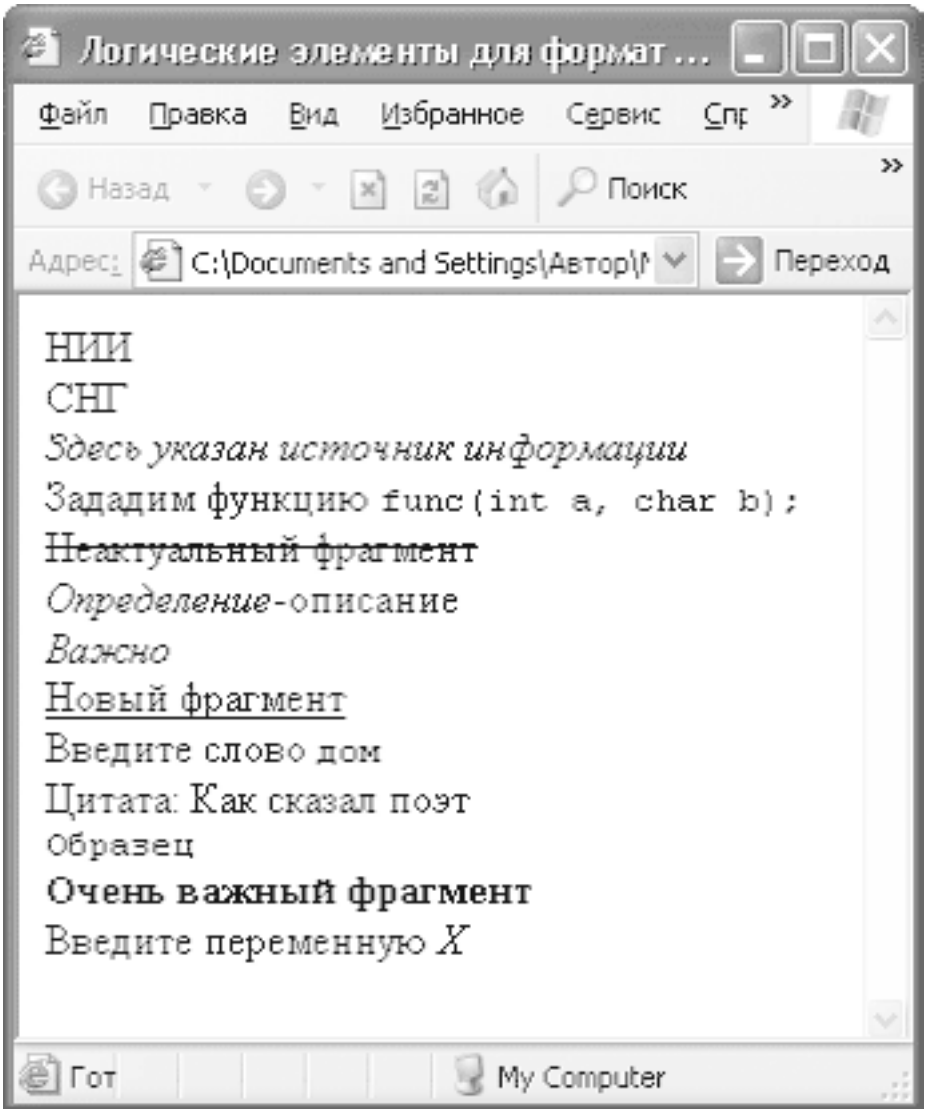

Рис. 2.16. Логическое форматирование текста

На рис. 2.16 видно, что не все логические элементы для форматирования изменяют вид текста, потому что предназначены не для этого. Различные браузеры могут по-разному выводить различные элементы, поэтому перед тем как выложить сайт в Интернете, проверьте его вид в различных программах.

Физические элементы для форматирования

Физические элементы для форматирования сообщают браузеру, как должен выглядеть текст, расположенный внутри элемента.

Элемент B

Элемент B задает полужирное написание шрифта.
Пример:
Полужирный шрифт

Элемент I

Элемент I отображает выделенный текст курсивом.
Пример:
Курсив

Элемент TT

Этот элемент задает моноширинное написание текста.
Пример:
<tt>Моноширинный шрифт</tt>

Элемент U

Элемент U отображает текст подчеркнутым шрифтом.
Пример:
<u>Подчеркнутый</u>

Элемент S

Элемент S зачеркивает текст горизонтальной линией.
Пример:
<s>Зачеркнутый</s>

Элемент STRIKE

Этот элемент также зачеркивает текст горизонтальной линией.
Пример:
<strike>Снова зачеркнутый</strike>

Элемент BIG

Элемент BIG отображает текст, расположенный внутри, шрифтом большего размера, чем остальной текст. Если быть точнее, то размер шрифта увеличивается на единицу.

Примечание

В языке HTML размеры шрифта измеряются в условных единицах от одного до семи. Размером по умолчанию принят третий размер.

При вложении элементов размер шрифта будет увеличиваться на единицу каждый раз.
Пример:
Шрифт <big>побольше <big>Еще больше</big></big>

Элемент SMALL

Элемент SMALL отображает выделенный текст шрифтом на единицу меньшего размера относительно окружающего текста. При вложении элементов шрифт будет уменьшаться на единицу с каждым вложением.

Пример:

Шрифт <small>поменьше</small>

Элемент SUB

Этот элемент задает подстрочное написание символов, то есть текст располагается ниже уровня базовой строки и становится меньшего размера. Удобно при вводе формул.

Пример:

_{Подстрочный} шрифт

Элемент SUP

Элемент SUP задает надстрочное написание символов, то есть текст располагается выше базовой линии и становится меньшего размера. Этот элемент используют при написании формул.

Пример:

^{Надстрочный} шрифт

Элемент SPAN

Элемент SPAN позволяет выделить часть текста и определить для нее особые параметры отображения с помощью таблиц стилей. Он применяется для выделения небольших областей текста.

Пример:

Текст с фоном

В листинге 2.17 представлен код страницы с различным форматированием текста.

Листинг 2.17. Физическое форматирование

```
<html>
<head>
<title>Элементы физического форматирования</title>
<body>
<b>Полужирный шрифт</b><br />
<i>Курсив</i><br />
<tt>Моноширинный шрифт</tt><br />
<u>Подчеркнутый</u><br />
<s>Зачеркнутый</s><br />
<strike>Снова зачеркнутый</strike><br />
Шрифт <big>побольше <big>Еще больше</big></big><br />
Шрифт <small>поменьше</small><br />
<sub>Подстрочный</sub> шрифт<br />
```

```
<sup>Надстрочный</sup> шрифт<br />
<span style="background-color:#00FFFF">Текст с фоном</span>
</body>
</html>
```

Результат обработки браузером кода из листинга 2.17 представлен на рис. 2.17.

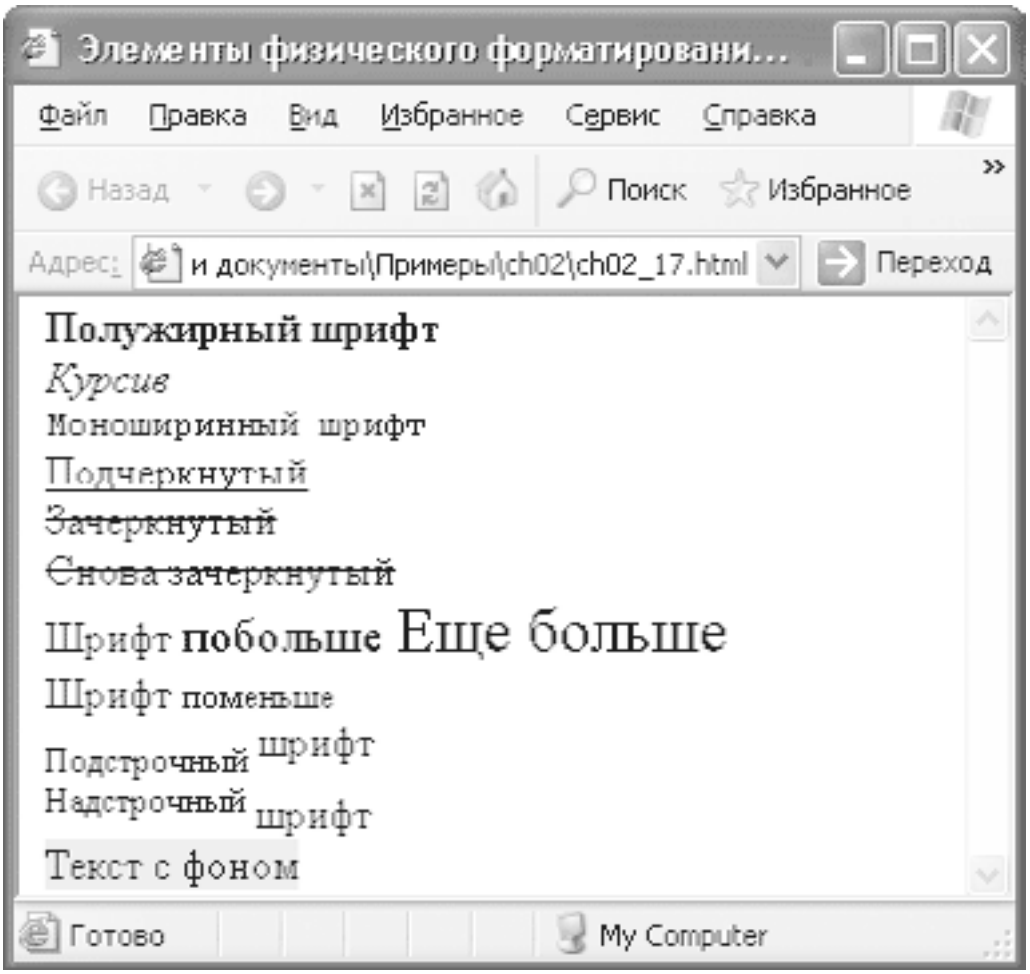

Рис. 2.17. Физическое форматирование

На рис. 2.17 видно, что любой физический элемент форматирования изменяет вид текста, потому что именно это является его функцией.

Как видно из описания, зачастую действие логических и физических элементов дублируется. В принципе, рекомендуется использовать логические элементы форматирования, так как они определяют суть фрагмента текста.

В табл. 2.1 кратко описаны аналоги рассмотренных физических элементов среди логических элементов и показано, какие элементы являются взаимозаменяемыми.

Таблица 2.1. Элементы форматирования текста

Физический элемент	Логический элемент	Действие на текст
B	STRONG	Полужирный
I	EM, DFN, VAR, CITE	Курсив
TT	CODE, SAMP, KBD	Моноширинный
U	INS	Подчеркнутый
S, STRIKE	DEL	Зачеркнутый
BIG	Нет	Увеличение шрифта
SMALL	Нет	Уменьшение шрифта
SUB	Нет	Подстрочный
SUP	Нет	Надстрочный

Как видно из таблицы, для многих физических элементов форматирования можно найти логические аналоги, которые намного лучше опишут смысловую нагрузку выделенного текста.

Все рассмотренные выше элементы применимы для форматирования небольших блоков текста. Дальше мы рассмотрим элементы, которые можно применять для изменения внешнего вида крупных текстовых блоков.

Элементы для форматирования больших блоков текста

Элементы, которые мы рассмотрим в этом подразделе, позволяют форматировать большие блоки текста. Они определяют параметры отображения и расположения текста, заключенного в их блок.

Начнем рассмотрение с элемента, напрямую отвечающего за параметры шрифта.

Элемент FONT

Элемент FONT задает параметры шрифта для текста. Хотя для форматирования предпочтительнее использовать таблицы стилей, некоторые простые документы допускают и такое определение параметров текста.

Параметры текста задаются с помощью атрибутов элемента FONT. Можно определить шрифт, размер и цвет текста, расположенного внутри него.

За шрифт отвечает атрибут face, значением которого должно быть название шрифта. Однако название должно быть знакомо компьютеру пользователя, иначе будет применен шрифт по умолчанию. Для решения проблемы несоответствия или отсутствия шрифтов можно задать несколько допустимых типов, введя их через запятую в качестве значения атрибута face.

За размер шрифта отвечает атрибут size. Значение задается в относительных величинах, то есть 2 или 6. По умолчанию используется размер 3. При этом можно задать размер шрифта относительно остального текста. Для этого нужно сначала указать +, если необходимо, чтобы размер шрифта на данном участке был больше, чем основной текст, или —, если требуется обратное форматирование. После знака надо указать количество пунктов, на которое текст должен быть больше или меньше.

За цвет шрифта отвечает атрибут color, значением которого должно быть либо ключевое слово, обозначающее имя цвета, либо код цвета в формате #RRGGBB.

Совет

Код цвета можно посмотреть в любом графическом редакторе.

В листинге 2.18 показан пример кода для определения параметров текста с помощью элемента FONT.

Листинг 2.18. Использование элемента FONT

```
<html>
<head>
<title>Элемент FONT</title>
<body>
<font color="#FF0000" face="Verdana, Arial, Helvetica, sans-serif" size="+2">
Текст красного цвета с размером +2
</font><br />
<font color="#FF00FF" face="Verdana, Arial, Helvetica, sans-serif" size="-4">
Текст фиолетового цвета с размером –4
</font><br />
</body>
</html>
```

Результат вывода на экран кода из листинга 2.18 показан на рис. 2.18.

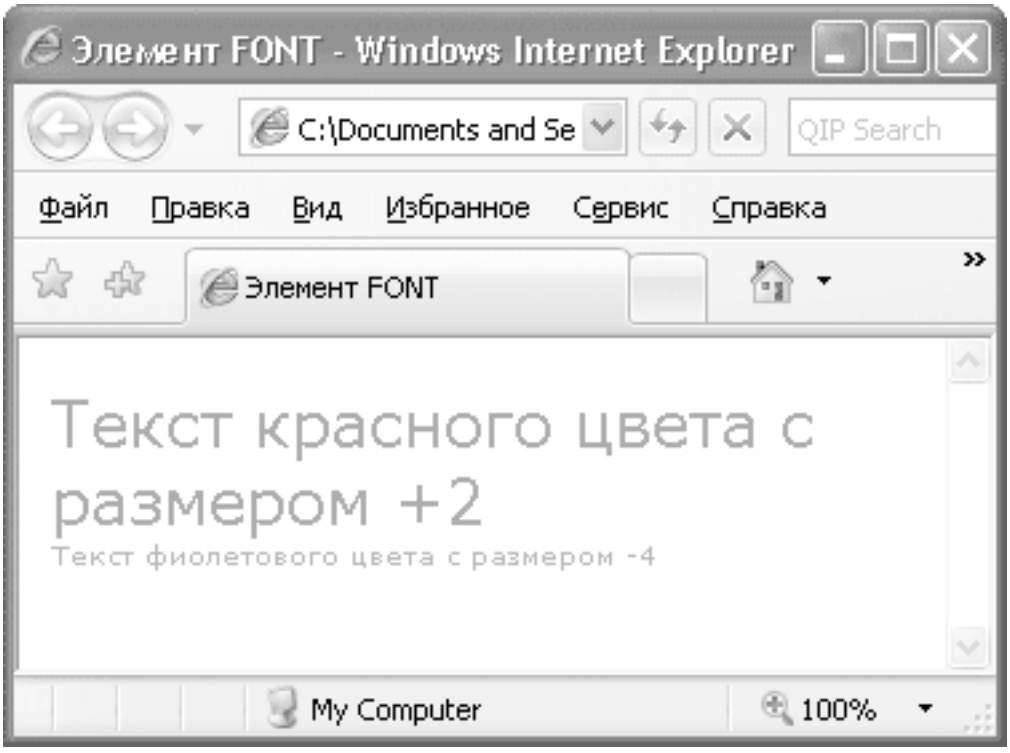

Рис. 2.18. Использование элемента FONT

В примере задан список похожих шрифтов. Браузер будет просматривать список по порядку и выведет текст тем шрифтом, который найдет первым. В качестве последнего варианта указан не шрифт, а семейство шрифтов Sans Serif. Если браузер не найдет ни один из перечисленных шрифтов, он возьмет известный ему шрифт из этого семейства.

Как задать параметры шрифта, мы разобрались. Однако этот метод не лучший. Как уже говорилось, удобнее задавать форматирование с помощью таблиц стилей.

Элемент DIV

Элемент DIV служит для выделения больших блоков текста под форматирование с помощью таблиц стилей. Иными словами, вы помещаете необходимый блок текста между тегами элемента DIV и либо задаете ему параметры в атрибуте style, либо подключаете класс из таблицы стилей с помощью атрибута class.

Единственное, что вы можете сделать с помощью HTML, – выровнять текст в блоке с помощью атрибута align и создать всплывающую подсказку для блока с помощью атрибута title.

В листинге 2.19 представлен пример кода для выделения текста с помощью элемента DIV.

Листинг 2.19. Использование элемента DIV

```
<html>
<head>
<title>Элемент DIV</title>
<body>
<div style="cursor:crosshair" align="center">
Текст внутри этого элемента выровнен по центру, а с помощью стилей задается вид указателя мыши при наведении на этот блок.
</div>
</body>
</html>
```

Результат обработки кода из листинга 2.19 показан на рис. 2.19.

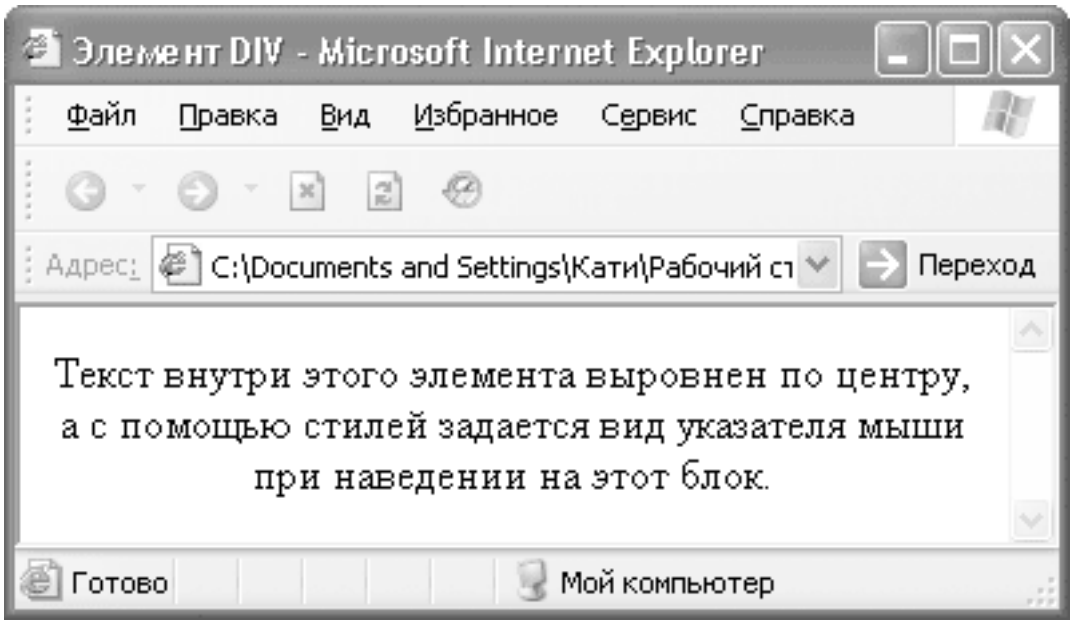

Рис. 2.19. Использование элемента DIV

Теперь рассмотрим ситуацию, когда у вас уже есть отформатированный нужным образом текст и вы не хотите ничего менять.

Элемент PRE

Этот элемент служит для ввода текста без форматирования, то есть с сохранением всех пробелов и переносов строк.

Примечание

Обычно в HTML несколько пробелов подряд воспринимаются как один пробел.

При использовании этого элемента текст выводится моноширинным шрифтом. Элемент PRE часто используют для вывода кодов программ. Внутри этого элемента можно применять большинство элементов форматирования текста.

В листинге 2.20 приведен пример использования элемента PRE.

Листинг 2.20. Использование элемента PRE

```
<html>
<head>
<title>Элемент PRE</title>
<body>
<pre>
Здесь
можно
расположить
код
программы
</pre>
</body>
</html>
```

На рис. 2.20 показано, как текст, расположенный между тегами элемента PRE, выглядит в браузере.

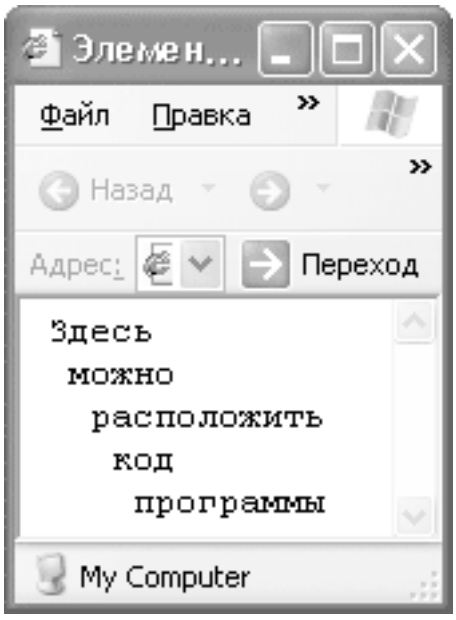

Рис. 2.20. Использование элемента PRE

Далее рассмотрим элемент, предназначенный для форматирования больших объемов текста и отвечающий за цитаты.

Элемент BLOCKQUOTE

Элемент BLOCKQUOTE определяет выделенный текст как цитату и применяется для описания больших высказываний. Он задает для текста отступы сверху, снизу и слева. Внутри этого элемента могут присутствовать элементы форматирования текста.

В листинге 2.21 представлен пример выделения цитаты с помощью элемента BLOCKQUOTE.

Листинг 2.21. Ввод больших цитат

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
<title>Элемент BLOCKQUOTE</title>
<body>
<blockquote>
Здесь можно расположить важную и большую цитату.
</blockquote>
</body>
</html>
```

На рис. 2.21 показано, как в браузере выглядит текст из элемента BLOCKQUOTE.

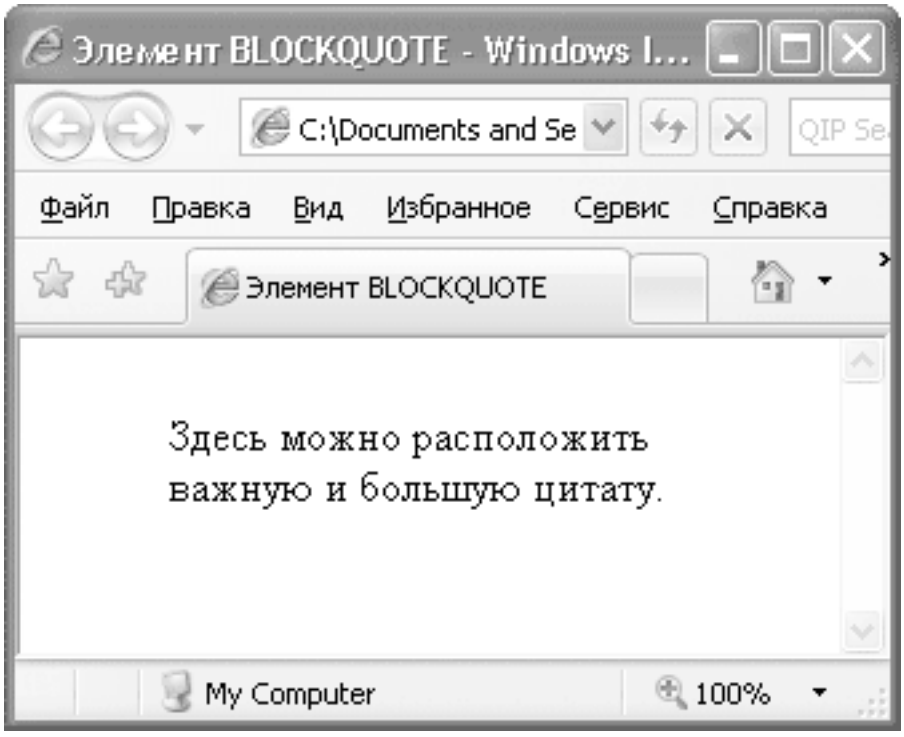

Рис. 2.21. Использование элемента BLOCKQUOTE

Мы рассмотрели, как действует каждый вариант форматирования текста отдельно, но HTML позволяет вкладывать элементы, при этом объединяя их действие.

Вложение элементов

Язык HTML позволяет вкладывать элементы форматирования друг в друга. При этом их действия суммируются. Если вложить в элемент B элемент I, то получится текст, написанный полужирным курсивом. При этом нужно следить за правильным закрытием элементов: тот, что открыт раньше, закрывается позже.

В листинге 2.22 показаны примеры правильного и неправильного вложения элементов.

Листинг 2.22. Вложение элементов

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
<title>Вложение элементов</title>
<body>
<del><b><i>Это неправильное вложение элементов</b></i></del><br />
<ins><b><i>Это правильное вложение элементов</i></b></ins>
</body>
</html>
```

Неправильную запись некоторые браузеры могут отображать некорректно.

Резюме

В данной главе были рассмотрены основные принципы форматирования текста с использованием возможностей HTML. Особенно подробно было рассказано об особенностях структурного форматирования документа и форматировании самого текста, были описаны особенности логического и физического форматирования текста, работа с большими блоками текста и вложением элементов.

Теперь внешний вид текста на вашем сайте полностью подвластен вам. Помните, что ни один символ не может быть написан напрямую в элементе BODY. Текст всегда должен быть включен в элемент, который определяет его назначение и внешний вид.

Глава 3
Создание таблиц

В технической литературе и в различных документах таблицы используют, чтобы расположить информацию в простом и понятном виде. О том, как использовать таблицы в своих сайтах, вы узнаете в этой главе.

В языке HTML таблицы используются в двух случаях: для представления числовых данных, разбитых по строкам и столбцам, или как средство форматирования веб-страниц, задания взаимного расположения элементов страницы. Ячейки таблицы могут содержать любые HTML-элементы, например заголовки, списки, абзацы, фигуры, графику, а также элементы форм. Фактически весь сайт расположен в большой сложной таблице (рис. 3.1).

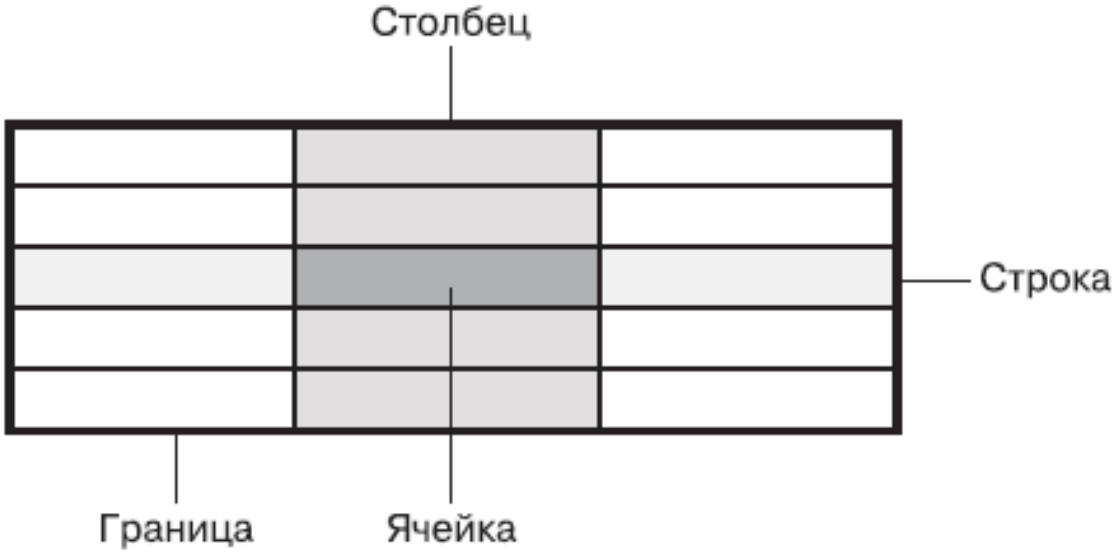

Рис. 3.1. Пример использования HTML-таблицы при создании сайта

Примечание

Как и в прошлой главе, напоминаю, что оформление таблицы согласно требованиям языка XHTML должно выполняться средствами CSS. Все элементы и атрибуты форматирования признаны в спецификации языка HTML 4. 01 нежелательными.

3.1. Что такое таблица

Таблица состоит из ячеек, образующихся при пересечении строк и столбцов (рис. 3.2).

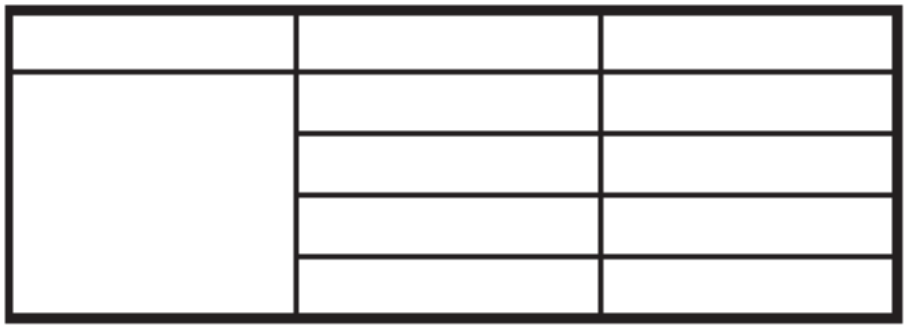

Рис. 3.2. Пример стандартной таблицы

Рассмотрим элементы таблицы.

• Ячейка – это основной элемент таблицы. Она формируется пересечением строки и столбца.

• Строка – это прямая линия ячеек, расположенных слева направо.

• Столбец – это набор ячеек, расположенных одна над другой сверху вниз.

• Граница – это линия, которая окружает каждую ячейку и таблицу в целом.

В стандартной таблице каждая строка и каждый столбец содержат одинаковое количество ячеек. Однако таблицы могут очень сильно отличаться от предложенной стандартной. В таблице, изображенной на рис. 3.3, одна ячейка объединяет в себе четыре строки, а другая – три столбца.

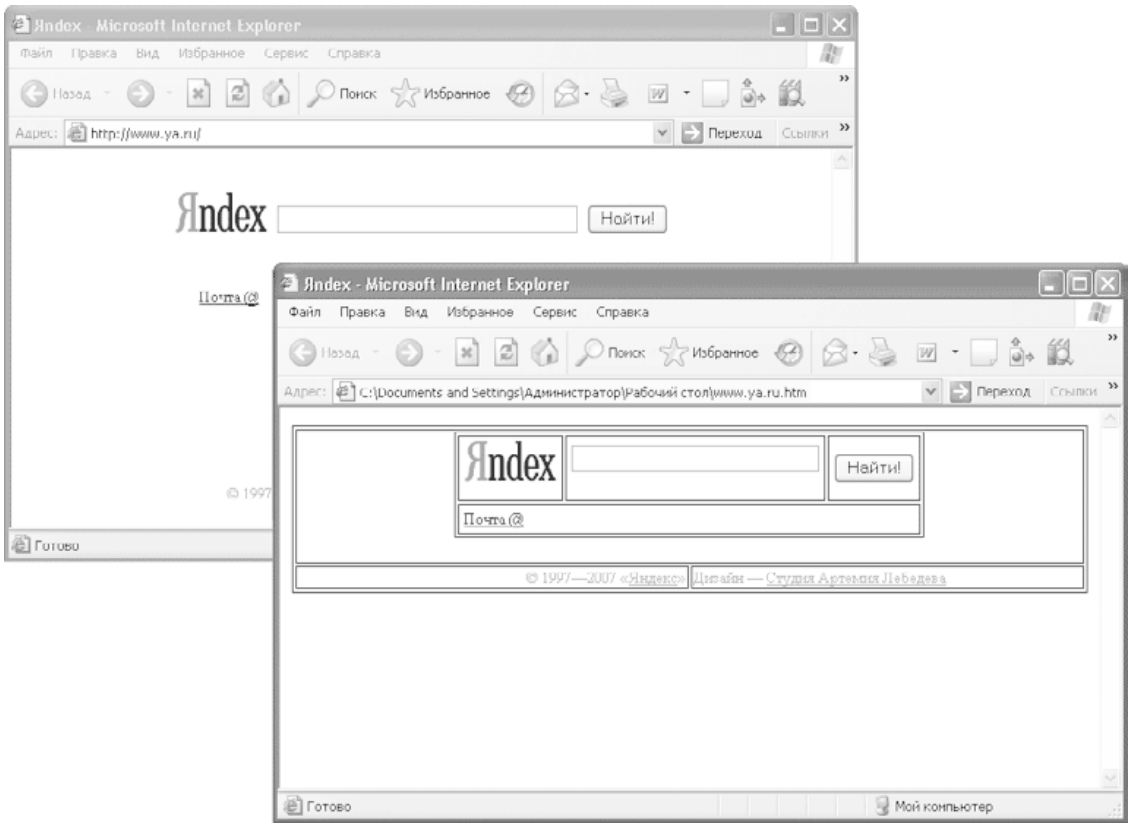

Рис. 3.3. Пример нестандартной таблицы

Кстати, при создании таблицы лучше начать с ее планирования, потом будет легче верстать, так как вы наглядно будете видеть вашу таблицу и будет меньше вероятности ошибиться. Для этого вам следует выбрать оптимальный способ создания таблицы.

Рассмотрим некоторые примеры планирования таблиц.

• Нарисуйте таблицу на бумаге.

• Нарисуйте таблицу в Paint, Adobe Photoshop или в AutoCAD.

• Используйте HTML-совместимые текстовые редакторы, например Microsoft Word. Создайте таблицу и сохраните в редакторе HTML-страницы. Затем откройте страницу в окне браузера и скопируйте исходный текст в ваш HTML-редактор для дальнейшей корректировки и форматирования.

3.2. Создание тела таблицы

В построении HTML-таблиц нет ничего сложного. Описание таблиц должно располагаться внутри элемента BODY. Документ может содержать произвольное количество таблиц, допускается вложение таблиц друг в друга. Каждая таблица должна начинаться тегом <TABLE> и завершаться тегом </TABLE>:

```
<body>
<table>
</table>
</body>
```

Все прочие элементы таблицы должны быть вложенными в элемент TABLE. Наименование таблицы определяется тегами <CAPTION>. </CAPTION>. Выравнивание наименования задается с помощью атрибута align, который может принимать значения top (над таблицей) и bottom (под таблицей). По умолчанию наименование располагается над таблицей.

3.3. Ячейки таблицы

Теперь приступим к созданию ячеек таблицы. Для начала нужно создать необходимое количество строк, затем поделить строки столбцами на ячейки. Количество пар тегов <TR> и </TR> определяет количество горизонтальных строк в вашей таблице. Встречаются случаи, когда строка создана только с помощью тега <TR> без использования закрывающего тега </TR>. Количество пар тегов <TD> и </TD>, расположенных между тегами соответствующей строки, определяет количество ячеек (столбцов) в пределах строки. Встречаются случаи использования элемента TD без закрывающего тега.

```
<body>
<table>
<tr><td> </td><td> </td></tr>
<tr><td> </td><td> </td></tr>
</table>
</body>
```

Чтобы созданная ячейка являлась заголовком в таблице (текст выравнивается по центру и выделяется полужирным шрифтом), нужно воспользоваться парой тегов <TH> и </TH>. Встречаются случаи использования элемента TH без закрывающего тега. Элементы TH, TR и TD без закрывающих тегов корректно интерпретируются только в последней версии браузера Internet Explorer.

Теперь вы можете создать простую HTML-таблицу. Рассмотрим пример таблицы, которая состоит из трех строк и трех столбцов, причем ячейки первой строки будут заголовками соответствующих столбцов (листинг 3.1). На рис. 3.4 показано, как данная таблица выглядит в окне браузера.

Листинг 3.1. Код простой HTML-таблицы, которая состоит из трех столбцов и трех строк

```
<html>
<head>
<title>Простая HTML-таблица</title>
</head>
<body>
<table>
<caption>Это заголовок таблицы</caption>
<tr><th>Заголовок 1</th><th>Заголовок 2</th><th>Заголовок 3</th></tr>
<tr><td>Ячейка 2x1 </td><td>Ячейка 2x2 </td><td>Ячейка 2x3 </td></tr>
<tr><td>Ячейка 3x1 </td><td>Ячейка 3x2 </td><td>Ячейка 3x3 </td></tr>
</table>
</body>
</html>
```

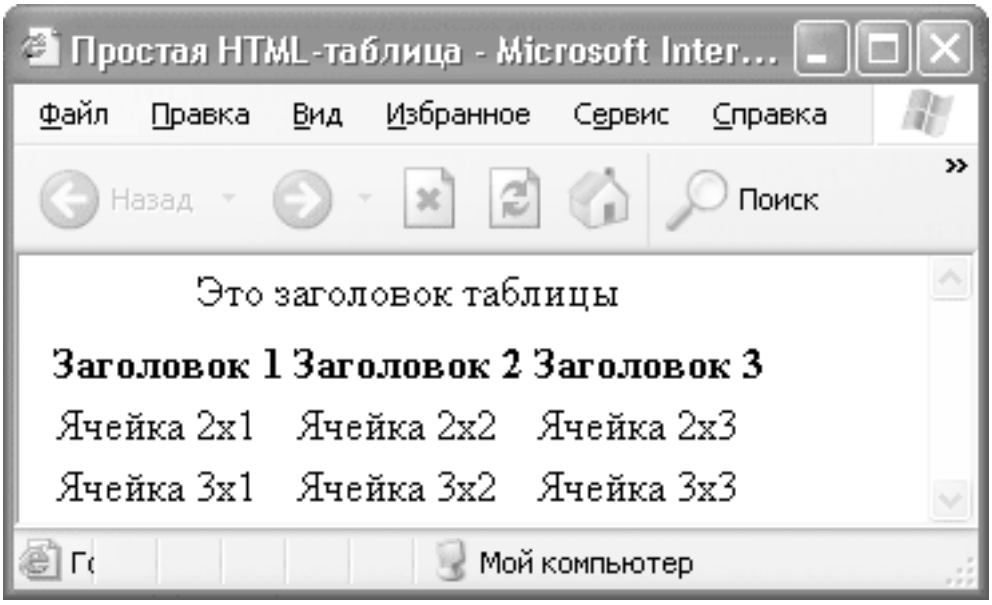

Рис. 3.4. Пример простой HTML-таблицы

Совет

Все браузеры игнорируют любые символы пробела и табуляции и перевод строки вне тегов HTML-документа, поэтому описывать таблицу можно так, чтобы она легко читалась. Я считаю, что использование предложенного варианта описания таблицы позволит вам не упустить ни одного тега HTML-документа.

3.4. Граница таблицы

В рассмотренном выше примере в таблице и ее ячейках отсутствовали границы. Граница таблицы создается с помощью атрибута border элемента TABLE. Ширина границы таблицы указывается в пикселах. Если атрибут border не указан, то таблица выводится без видимой рамки. Благодаря атрибуту bordercolor можно задать цвет границы таблицы, указав код цвета. Добавим к уже созданной таблице черную границу шириной четыре пиксела.

Для этого изменим строку

<table>

на

<table border="4" bordercolor="#000000">

Граница таблицы (и другие элементы сайта) в окне каждого браузера отображается по-разному. На рис. 3.5 показано, как созданная таблица отображается в окне стандартного браузера Internet Explorer, а на рис. 3.6 эта же таблица изображена в браузере Opera. Опытные веб-программисты стараются просматривать созданную страницу под разными типами браузеров, чтобы все везде выглядело одинаково. У каждого браузера свои капризы, так что создать сайт, чтобы все выглядело одинаково в любом браузере, – задача непростая.

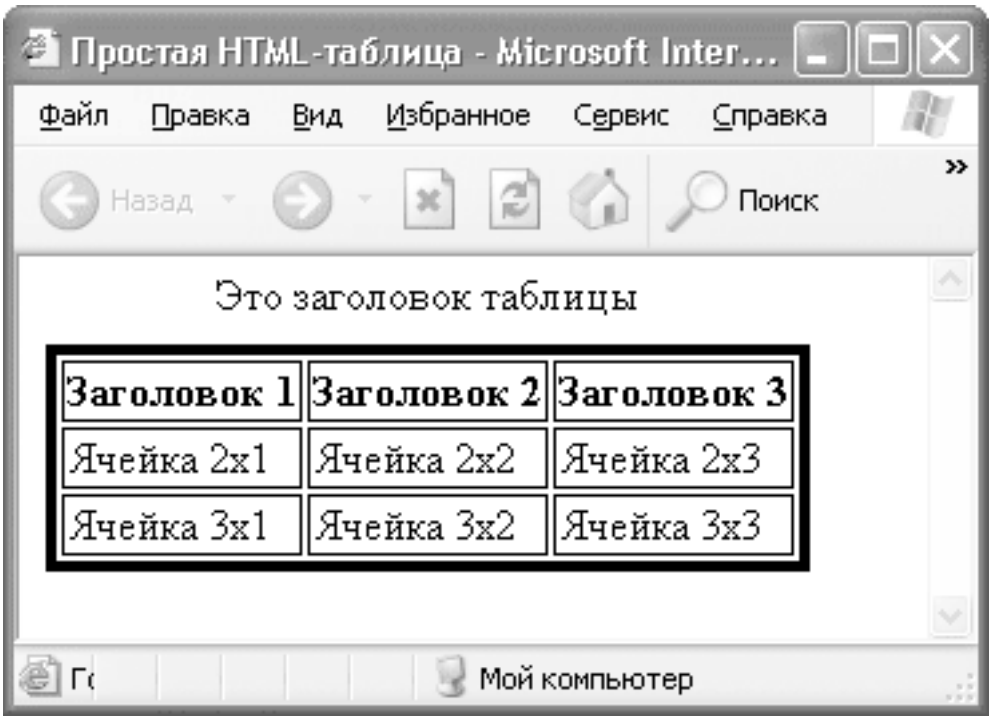

Рис. 3.5. Отображение таблицы в окне Internet Explorer

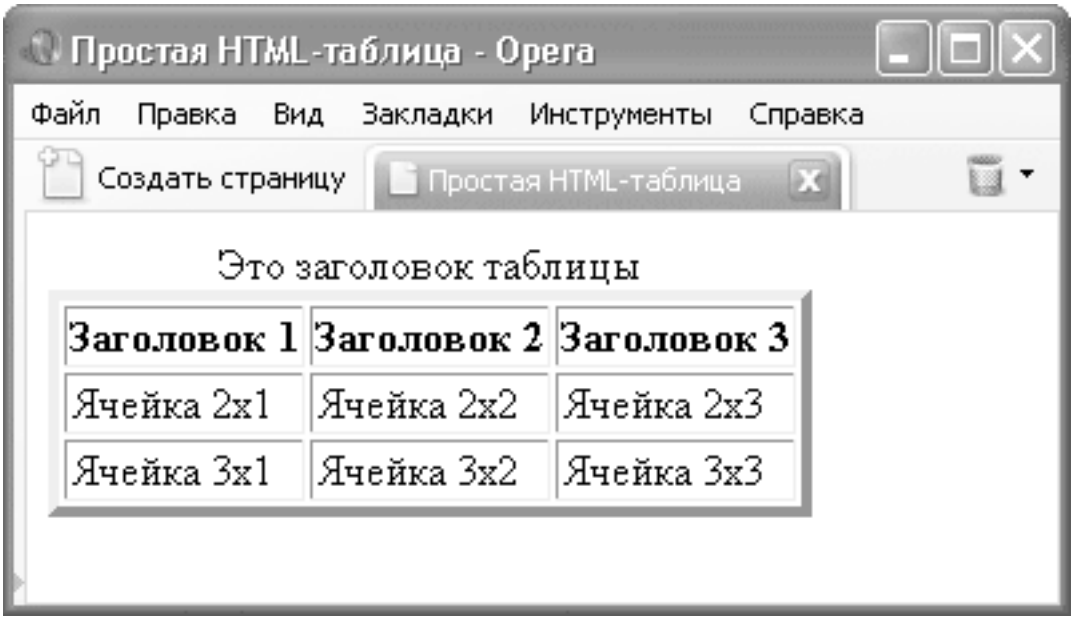

Рис. 3.6. Отображение таблицы в окне Opera

Из рис. 3.5 видно, что между границами ячеек и внешней границей таблицы существует свободное пространство. Это пространство легко регулируется атрибутами cellspacing и cellpadding элемента TABLE. Атрибут cellspacing определяет расстояние между ячейками таблицы в пикселах, а атрибут cellpadding – между содержимым ячейки и ее границей.

В листинге 3.2 приведен пример таблицы, у которой расстояние между содержимым ячеек и границей равно 10 пикселов, а расстояние между ячейками равно нулю (рис. 3.7).

Листинг 3.2. Код HTML-таблицы с нестандартными расстояниями между ячейками и границей

```
<html>
<head>
<title>Простая HTML-таблица</title>
</head>
<body>
<table border="4" bordercolor="#000000" cellspacing="0" cellpadding="10">
<caption>Это заголовок таблицы</caption>
<tr><th>Заголовок 1</th><th>Заголовок 2</th><th>Заголовок 3</th></tr>
<tr><td>Ячейка 2x1 </td><td>Ячейка 2x2 </td><td>Ячейка 2x3 </td></tr>
<tr><td>Ячейка 3x1 </td><td>Ячейка 3x2 </td><td>Ячейка 3x3 </td></tr>
</table>
</body>
</html>
```

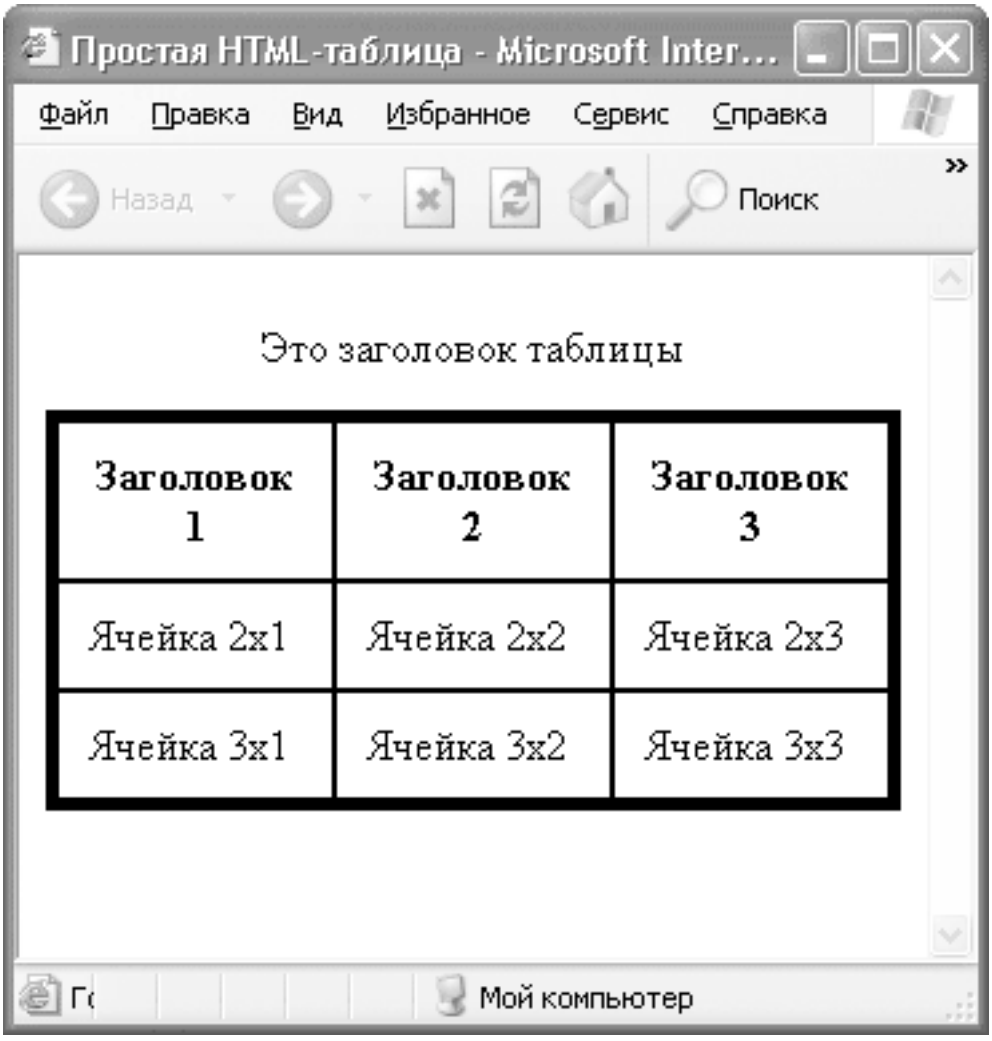

Рис. 3.7. Таблица с нестандартными расстояниями между ячейками и границей

Получив некоторые навыки работы с границей таблицы, вы можете приступать к управлению ее отображением. С помощью атрибута frame будем управлять отображением внешней границы, а с помощью атрибута rules – внутренними границами таблицы. Атрибут frame может принимать следующие значения:

- above – отображается только верхняя линия границы;
- below – видна только нижняя линия границы;
- box – отображается внешняя граница таблицы;
- border – видна внешняя граница таблицы (аналогично box);
- hsides – отображаются только горизонтальные линии границы;
- lhs – видна только левая линия границы;
- rhs – отображается только правая линия границы;
- void – внешняя граница таблицы не отображается;
- vsides – видны только вертикальные линии границы.

Frame и rules – это новые атрибуты в HTML, они поддерживаются только последними версиями браузеров.

Атрибут rules может принимать следующие значения:

- all – граница отображается вокруг каждой ячейки;
- cols – видны только вертикальные границы между столбцами;
- groups – отображаются только вертикальные границы между группами столбцов или горизонтальные границы между группами строк;

• none – границы между ячейками не отображаются;
• rows – видны только горизонтальные границы между группами строк.

В листинге 3.3 приведен пример использования атрибутов frame и rules.

Листинг 3.3. Код HTML-таблицы с частичным отображением линий внешней и внутренней границ

```
<html>
<head>
<title>Простая HTML-таблица</title>
<head>
<body>
<table border="4" bordercolor="#000000" cellspacing="0" cellpadding= "10" frame= Hsides
rules= Cols>
<caption>Склонение слова пиксел по падежам </caption>
<tr><th>Падеж</th><th>Пример </th></tr>
<tr><td>Им. </td><td>пиксел </td></tr>
<tr><td>Рд. </td><td>пиксела </td></tr>
<tr><td>Дт. </td><td>пикселу </td></tr>
<tr><td>Вт. </td><td>пиксел </td></tr>
<tr><td>Тв. </td><td>пикселом </td></tr>
<tr><td>Пр. </td><td>о пикселе </td></tr>
</table>
</body>
</html>
```

На рис. 3.8 показано, как рассмотренная таблица будет отображаться в окне браузера. У таблицы есть горизонтальные линии внешней и вертикальные линии внутренней границы, так как именно такие значения мы задали соответствующим атрибутам rules и frame.

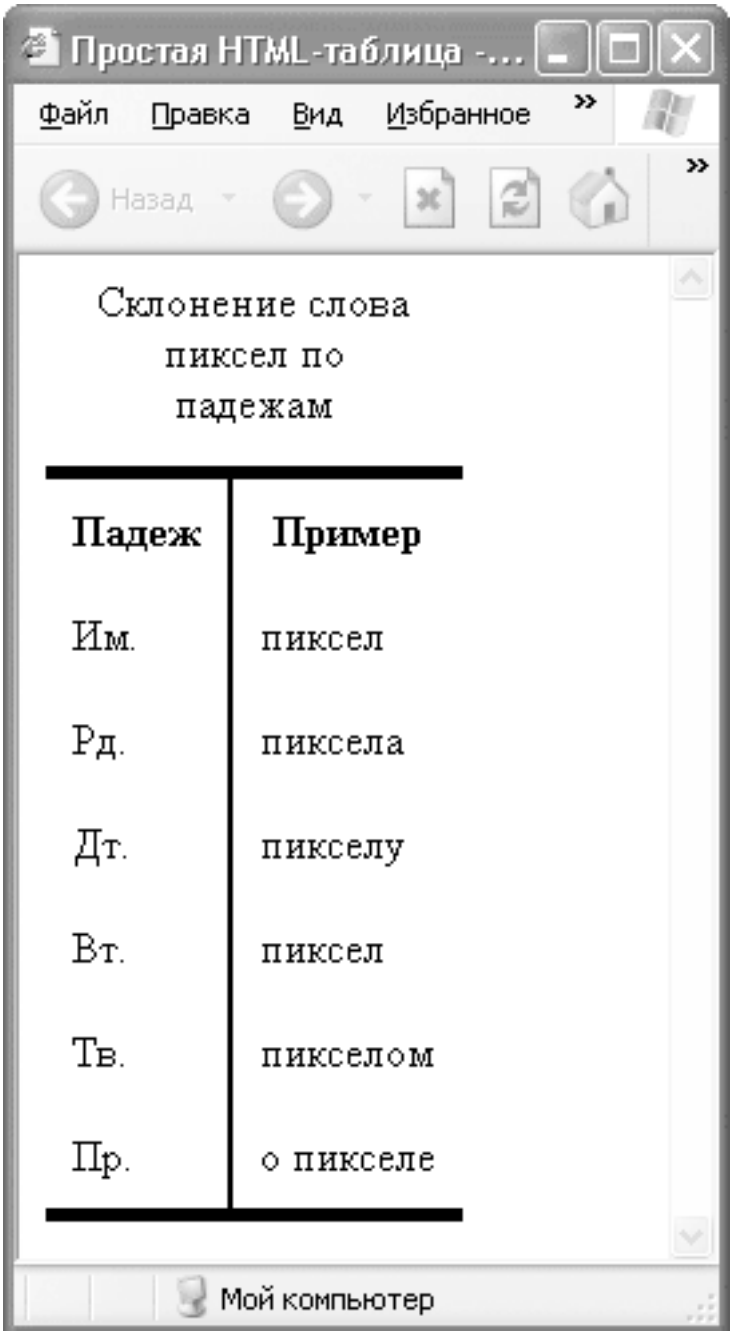

Рис. 3.8.HTML-таблица с частичным отображением линий внешней и внутренней границ

3.5. Ширина и высота таблицы и ячеек

Ширина таблицы задается атрибутом width элемента TABLE. Значение можно задавать как в абсолютных единицах (width="2 5 0"), так и в относительных (width="80 %"). Например, задав значение ширины в 600 пикселов, можно быть уверенным, что таблица поместится в окне браузера при любом разрешении монитора. Если же ширина задается в процентах, то они высчитываются от ширины окна браузера или от ширины ячейки другой таблицы, в которую вставлена данная. То же самое можно делать и с высотой таблицы с помощью атрибута height.

Все вышесказанное относится и к ячейкам таблицы. Просто необходимо использовать элементы с соответствующими атрибутами. При этом вовсе не обязательно задавать размеры каждой отдельной ячейки. При изменении ширины или высоты ячейки все соседние ячейки в пределах столбца будут отображаться с учетом нового значения.

При задании чрезмерно малых величин ширины и высоты таблицы браузер определяет минимальные значения, которые позволяют нормально отображать данные.

На рис. 3.9 и 3.10 изображены две таблицы одинакового содержания, но разной ширины и высоты.

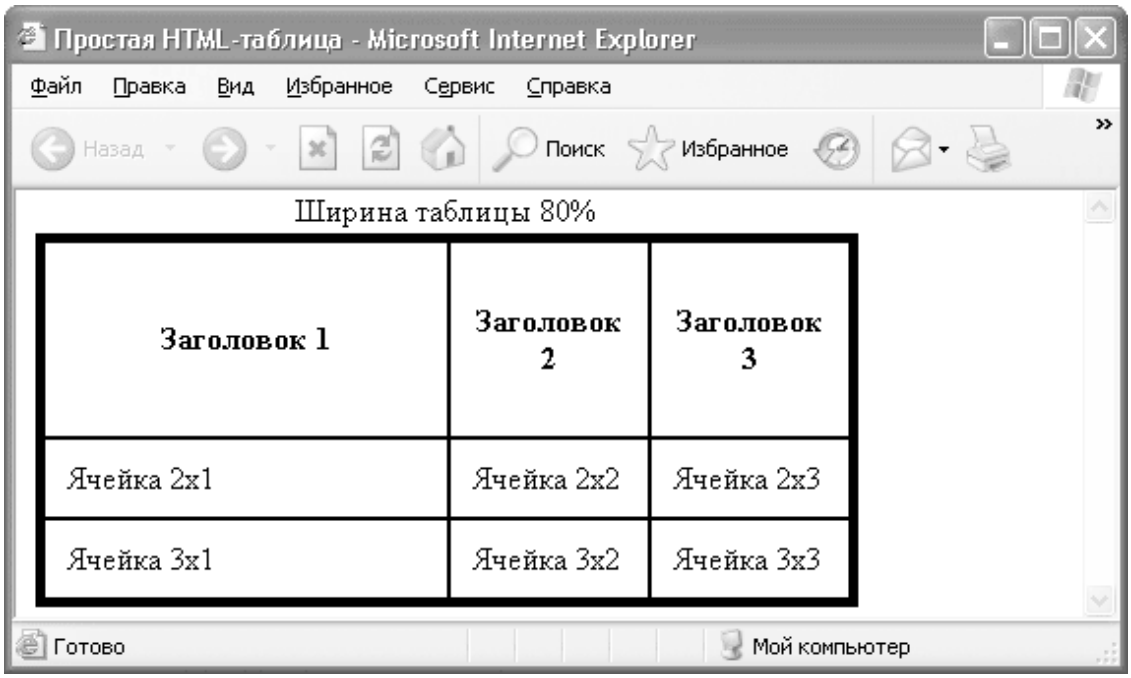

Рис. 3.9. Ширина таблицы равна 80 % от ширины окна браузера

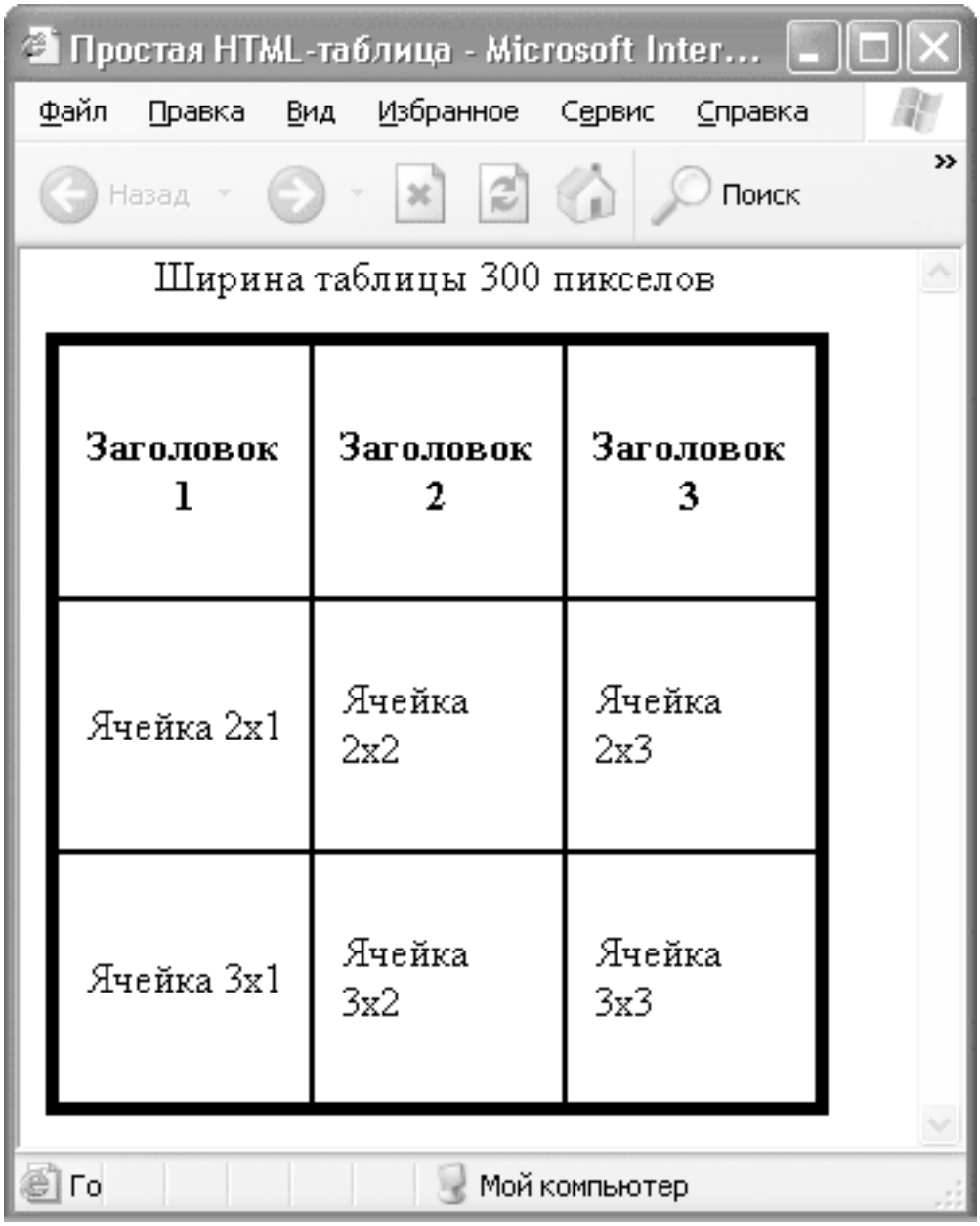

Рис. 3.10. Ширина и высота таблицы равна 300 пикселам

Ширина первой таблицы равна 80 % от размера окна браузера, а первый столбец этой таблицы занимает 50 % от всей ширины таблицы. Высота первой строки равна 100 пикселам.

Вторая таблица квадратная, ширина стороны равна 300 пикселам. В листингах 3.4 и 3.5 приведены коды страниц, содержащих описанные таблицы.

Листинг 3.4. Код таблицы шириной 80 % от ширины окна браузера

```
<html>
<head>
<title>Простая HTML-таблица</title>
</head>
<body>
<table border="4" bordercolor="#000000" cellspacing="0" cellpadding="10" width="80%">
<caption>Ширина таблицы 80%</caption>
```

```
<tr><th height="100" width="50%">Заголовок 1</th>
<th>Заголовок 2</th><th>Заголовок 3</th></tr>
<tr><td>Ячейка 2x1 </td><td>Ячейка 2x2 </td><td>Ячейка 2x3 </td></tr>
<tr><td>Ячейка 3x1 </td><td>Ячейка 3x2 </td><td>Ячейка 3x3 </td></tr>
</table>
</body>
</html>
```

Листинг 3.5. Код таблицы шириной 300 пикселов

```
<html>
<head>
<title>Простая HTML-таблица</title>
</head>
<body>
<table border="4" bordercolor="#000000" cellspacing="0" cellpadding="10" width="300" height="300">
<caption>Ширина таблицы 300 пикселов</caption>
<tr><th>Заголовок 1</th><th>Заголовок 2</th><th>Заголовок 3</th></tr>
<tr><td>Ячейка 2x1 </td><td>Ячейка 2x2 </td><td>Ячейка 2x3 </td></tr>
<tr><td>Ячейка 3x1 </td><td>Ячейка 3x2 </td><td>Ячейка 3x3 </td></tr>
</table>
</body>
</html>
```

3.6. Группировка строк и столбцов

В стандарте HTML 4 появились новые элементы для группировки (не объединения, а именно группировки) строк и столбцов таблицы в группы с общими свойствами.

Для группировки столбцов таблицы служат элементы COLGROUP и COL. Элемент COLGROUP создает структурную группу столбцов, которая выделяет множество логически однородных ячеек. Так, одна структурная группа может охватывать ячейки заголовков столбцов, а другая – ячейки, содержащие данные. Элемент COL предназначен для формирования неструктурных групп столбцов, которые делят таблицу на разделы, не имеющие отношения к структуре. Это удобно в том случае, когда не все столбцы содержат информацию одного типа.

Полезным атрибутом элементов COLGROUP и COL является атрибут span со значением n. Атрибут распространяет свойства, заданные этими элементами на n столбцов в группе.

```
<table>
<col span=2 style="color:red">
<tr><td> Этот текст будет красным цветом </td>
<td> И этот текст будет красным цветом </td>
<td> А этот текст будет черным цветом </td></tr>
</table>
```

Для группировки строк таблицы служат элементы THEAD, TBODY и TFOOT. Их использование существенно облегчает компоновку и форматирование таблиц.

Для создания группы заголовков для столбцов таблицы используют элемент THEAD. Его допускается использовать в пределах таблицы только один раз. Для создания одной или нескольких групп строк таблицы, содержащих основные данные, применяется элемент TBODY. Элемент TFOOT позволяет создать группу строк для представления информации о суммах или итогах в нижней части таблицы. Этот элемент допускается использовать в пределах таблицы только один раз. Вовсе не обязательно создавать группы строк таблицы всех трех типов.

```
<thead>
<tr><th> </th><th> </th><th> </th></tr>
</thead>
</tbody>
<tr><td> </td><td> </td><td> </td></tr>
<tr><td> </td><td> </td><td> </td></tr>
<tbody>
```

Предположим, нужно создать таблицу, в которой внутренние линии не отображаются, а отображается только линия, отделяющая заголовки столбцов от основного текста (тела таблицы). Чтобы создать такую таблицу, необходимо выполнить следующее.

1. Сгруппировать строки нужным образом.
2. Указать видимость границы между группами строк.

3.7. Выравнивание таблицы и содержимого ячеек

Для выравнивания элементов таблиц по горизонтали и вертикали в элементах TABLE, TR, TH и TD используют атрибуты align и valign.

Атрибут align применяется ко всем элементам таблицы и определяет общее горизонтальное выравнивание:

• <table align=left/right/center> – таблицы на странице по левому краю/правому краю/по центру;

• <tr align=left/right/center> – элементов строки по левому краю/правому краю/по центру;

• <th align=left/right/center> – заголовка таблицы по левому краю/правому краю/по центру (по умолчанию по центру);

• <td align=left/right/center/char> – данных в ячейке по левому краю/ правому краю/по центру/по заданному символу (по умолчанию по левому краю).

Атрибут valign также применяется ко всем элементам таблицы и определяет общее вертикальное выравнивание:

• <table valign=bottom/middle/top> – элементов таблицы внизу/по центру/вверху (по умолчанию по центру);

• <tr valign=bottom/middle/top> – элементов строки внизу/по центру/ вверху;

• <th valign=bottom/middle/top> – заголовка таблицы внизу/по центру/ вверху;

• <td valign=bottom/middle/top> – данных в ячейке внизу/по центру/ вверху.

В листинге 3.6 приведен пример использования в различных комбинациях описанных выше атрибутов форматирования таблицы и содержимого ячеек (рис. 3.11).

Листинг 3.6. Пример использования выравнивания таблицы и содержимого ячеек

```
<html>
<head>
<title>Простая HTML-таблица</title>
</head>
<body>
<table border="4" bordercolor="#000000" cellspacing="0" cellpadding="0"width= "400" height="150" align=center>
<caption>Наименование товара</caption>
<tr><th>Товар </th><th>Код</th><th>Количество</th><th>Цена </th></tr>
<tr valign=bottom align=center>
<td>Клей</td><td>028</td><td>190 шт </td><td>12,2 руб</td></tr>
<tr valign=bottom align=center>
<td>Скотч</td><td>058</td><td>120 шт </td><td>4,6 руб </td></tr>
<tr valign=bottom align=center>
<td>Ластик</td><td>986</td><td>100 шт </td><td>2,3 руб </td></tr>
</table>
</body>
</html>
```

Наименование товара

Товар	Код	Количество	Цена
Клей	028	190 шт	12,2 руб
Скотч	058	120 шт	4,6 руб
Ластик	986	100 шт	2,3 руб

Рис. 3.11. Пример использования выравнивания таблицы и содержимого ячеек

3.8. Объединение ячеек таблицы

На практике встречается большое количество таблиц, в которых одна ячейка объединяет в себе несколько ячеек по высоте и ширине (см. рис. 3.2). В HTML ячейки объединяют с помощью атрибутов colspan и rowspan. Атрибут colspan определяет количество ячеек, на которые простирается данная ячейка по горизонтали, а rowspan – по вертикали.

На рис. 3.12 изображена таблица с объединенными ячейками. Заголовок таблицы находится в ячейке, объединяющей все три ячейки строки. Таблица содержит еще две ячейки, каждая из которых объединяет две ячейки по вертикали. Код такой таблицы приведен в листинге 3.7.

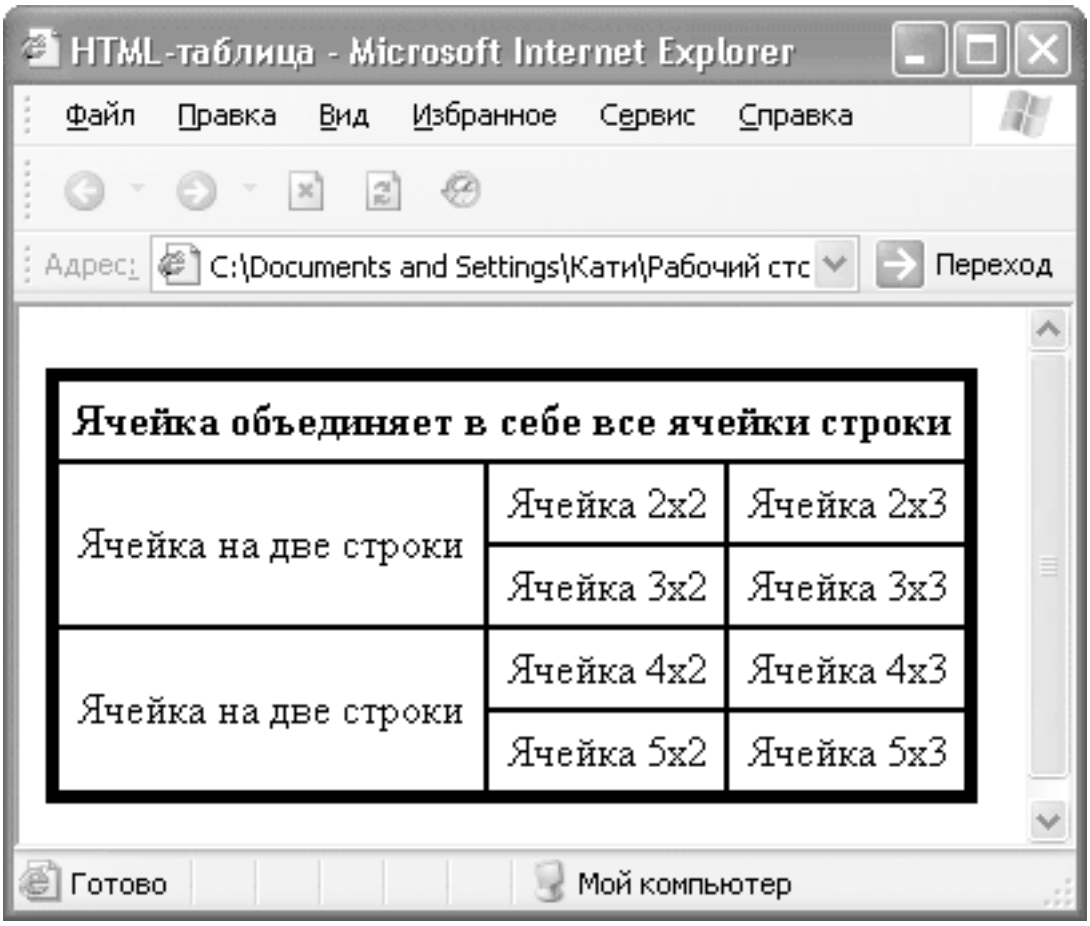

Рис. 3.12. Таблица с объединенными ячейками

Листинг 3.7. Пример использования атрибутов объединения ячеек

```
<html>
<head>
<title>HTML-таблица</title>
<head>
<body>
<table border="4" bordercolor="#000000" cellspacing="0" cellpadding="5" >
<tr align=center><th colspan=3>Ячейка объединяет в себе все ячейки строки </th></tr>
<tr align=center><td rowspan=2>Ячейка на две строки </td>
```

```
<td>Ячейка 2x2</td><td>Ячейка 2x3</td></tr>
<tr align=center><td>Ячейка 3x2</td><td>Ячейка 3x3</td></tr>
<tr align=center><td rowspan=2>Ячейка на две строки</td>
<td>Ячейка 4x2</td><td>Ячейка 4x3</td></tr>
<tr align=center><td>Ячейка 5x2</td><td>Ячейка 5x3</td></tr>
</table>
</body>
</html>
```

3.9. Установка фонового цвета или рисунка ячейки

В HTML возможны различные варианты установки фонового цвета или рисунка. Благодаря атрибуту bgcolor можно изменять цвет содержимого ячейки, строки, группы столбцов, группы строк, таблицы целиком.

Используя конструкцию <td bgcolor= "#000000">, чередующуюся с конструкцией <td></td> (со стандартным фоновым цветом ячеек), можно создать таблицу, изображенную на рис. 3.13.

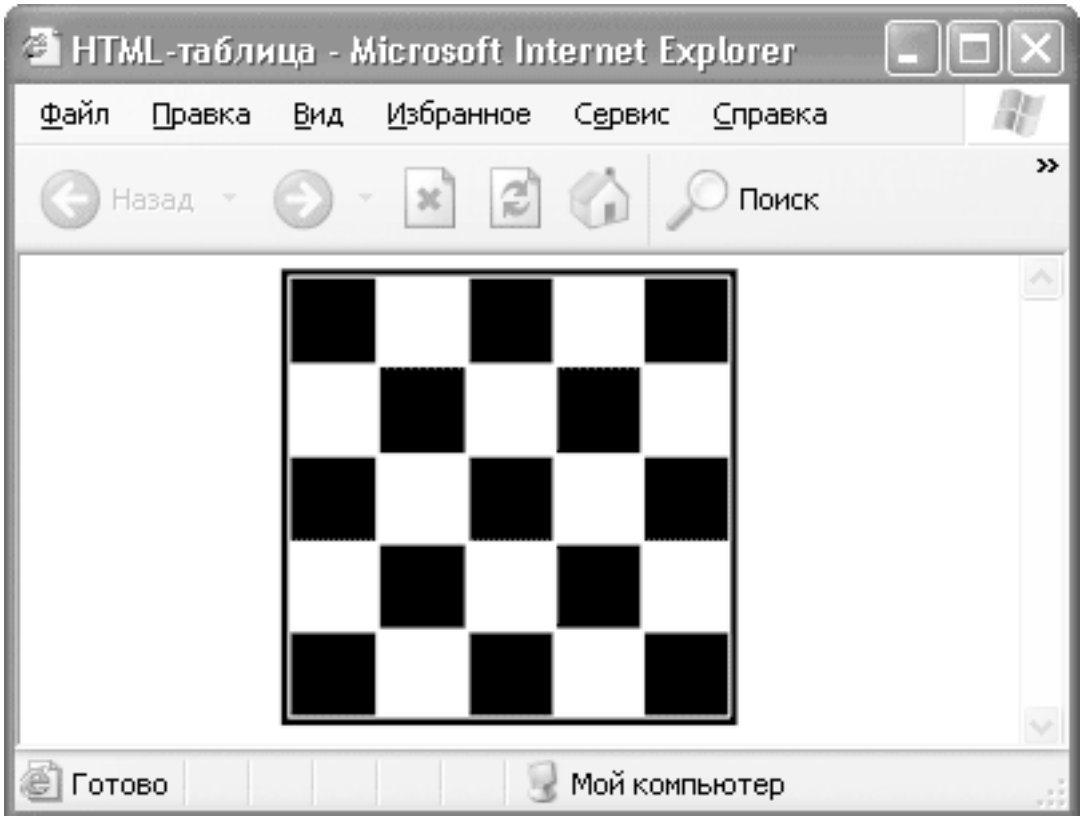

Рис. 3.13. Пример использования атрибута bgcolor

С помощью атрибута background можно задать графический фон ячейки или таблицы целиком, указав путь к изображению. Например, <td background="img\fon.gif">.

Если размеры изображения превышают габариты ячейки или таблицы, для которых оно предназначено, браузер обрезает его таким образом, чтобы оно уместилось в пределах соответствующего объекта.

3.10. Создание вложенных таблиц

Существуют такие моменты, когда необходимо создать ячейки, границы которых не должны совпадать (рис. 3.14). Здесь атрибут colspan бессилен.

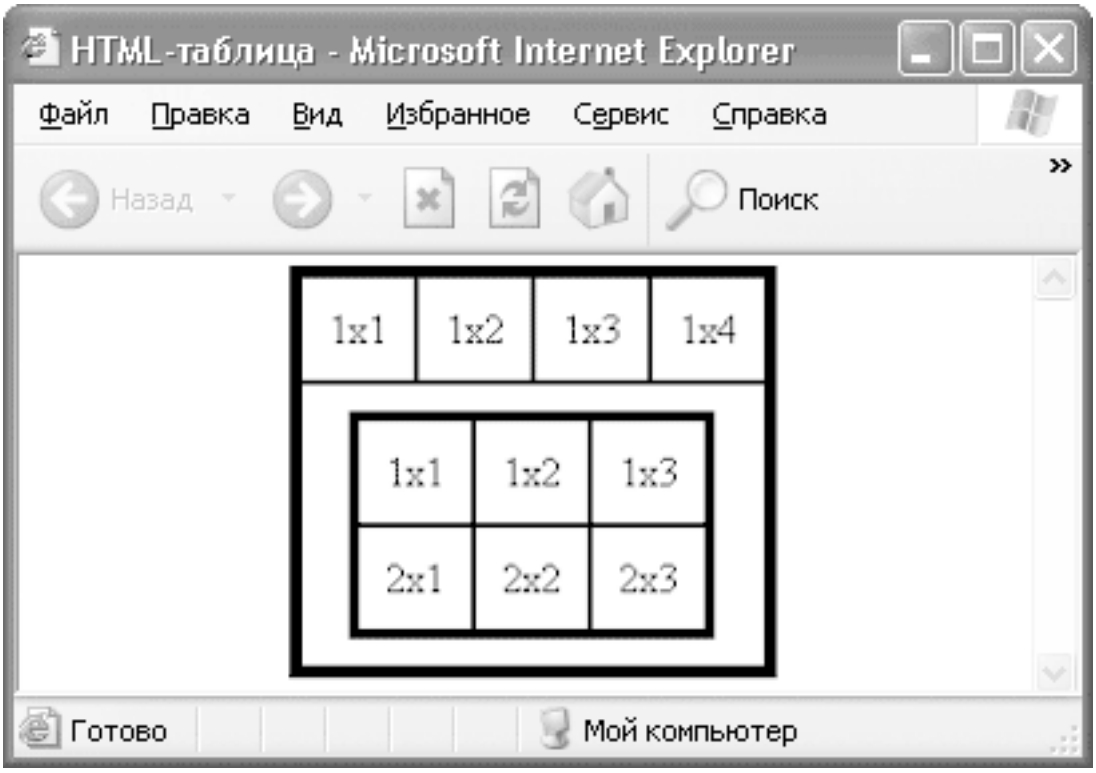

Рис. 3.14. Сложная HTML-таблица

Команды, управляющие созданием и отображением таблиц, допускают вложение одного элемента TABLE внутрь другого, поэтому в ячейке можно создать новую таблицу с независимой структурой.

В листинге 3.8 приведен код таблицы, изображенной на рис. 3.13. Здесь применяется метод вложения одной таблицы в другую. Внешняя таблица состоит из двух строк. Первая строка содержит четыре ячейки, вторая – таблицу со своей независимой структурой. Внутренняя таблица, в свою очередь, содержит две строки, каждая из которых содержит по три ячейки.

Листинг 3.8. Пример таблицы, которая содержит еще одну таблицу

```
<html>
<head>
<title>HTML-таблица</title>
<head>
<body>
<table border="4" bordercolor="#000000" cellspacing="0" cellpadding="10">
<caption>Создание вложенных таблиц</caption>
<tr><td> 1x1</td><td>1x2</td><td>1x3</td><td>1x4</td></tr>
<tr><td colspan=4>
<table border="3" bordercolor="#000000" cellspacing="0" cellpadding="10">
<tr><td> 1x1</td><td>1x2</td><td>1x3</td></tr>
```

```
<tr><td> 2x1</td><td>2x2</td><td>2x3</td></tr>
</table>
</td>
</tr>
</table>
</body>
</html>
```

Резюме

В заключение следует отметить, что таблицы являются важнейшей частью любой веб-страницы. Для контроля над элементами веб-узла фирмы Microsoft (www.microsoft.com) использованы таблицы.

Умелое обращение с таблицами позволяет жестко связать текстовые блоки документа с графикой и другими объектами. Использование таблиц позволит странице загружаться быстрее, но большие рисунки следует разбить на несколько маленьких. Чтобы не нарушить целостность всего рисунка, его отдельные части следует расположить в ячейках таблицы таким образом, чтобы они образовали целый рисунок.

Использование таблиц также позволит вам разделить всю HTML-страницу на функциональные части: заголовок страницы, место для рекламы, панель инструментов и т. д. Подобное разделение страницы даст возможность быстро создать свой индивидуальный «остов» сайта, который в последующем можно использовать для создания других страниц.

Глава 4
Добавление изображений и мультимедиа

При создании сайта невозможно обойтись без использования активных объектов вроде Flash-анимации, видео или картинок. Так мы можем улучшить вид сайта, сделать его ярче, интереснее и удобнее для посетителей. Однако неграмотное использование мультимедиа на странице способно погубить самые лучшие и интересные сайты. Поэтому в данной главе, помимо технических аспектов встраивания мультимедиа, будут освещены эстетические аспекты и моменты, связанные с удобством для посетителя.

4.1. Встраивание изображений

Начнем с добавления изображения, потому что это самый простой для добавления и самый распространенный мультимедийный элемент, встречающийся в Интернете. У изображений много плюсов: статичность, небольшие размеры файлов (относительно других типов мультимедиа-содержимого), широкая область применения. Сейчас трудно представить сайт без картинок. Дизайнеры научились использовать их очень разумно. Современные скорости соединений позволяют размещать большое количество графики на странице. Однако нужно знать меру, сайт не должен выглядеть пустым, но в то же время не стоит и злоупотреблять рисунками. Грамотное и уместное использование изображений поможет сделать сайт красивым, интересным и удобным. Бывают ситуации, когда без большого количества картинок невозможно обойтись, например при создании галереи, фотоальбома, каталога. В таких случаях умелое распределение файлов по сайту и удобная навигация помогут сэкономить время загрузки и трафик. Получается, что при создании сайта без изображений вам не обойтись, этот объект является самым простым, удобным и распространенным.

Для встраивания изображений в HTML-документ применяется элемент IMG. Он имеет обязательный атрибут src, значением которого должен быть адрес встраиваемого изображения.

Простейший вариант записи для включения картинки: <IMG src=" image.jpg">. При такой записи размер картинки на экране будет соответствовать ее реальному размеру.

> **Примечание**
>
> Адрес изображения может быть указан либо полностью (например, когда картинка находится на другом сервере (http://www.mypage.ru/IMG/myfoto.jpg)), либо относительно местоположения вашего документа (например, если картинка находится во вложенной папке (IMG/myfoto.jpg)). Во втором случае для указания директории, находящейся выше в иерархии каталогов, используются символы../ (переход из папки DOC, находящейся в одном каталоге с папкой IMG, будет выглядеть так:. /IMG/myfoto.jpg).

В листинге 4.1 продемонстрирован простейший вариант встраивания картинки.

Листинг 4.1. Простое встраивание изображения в документ

```
<html>
<head>
<title>Встраивание изображения</title>
</head>
<body>
<img src="image.jpg"/>
</body>
</html>
```

На рис. 4.1 показан результат обработки браузером кода из листинга 4.1 – простое встраивание картинки, без редактирования.

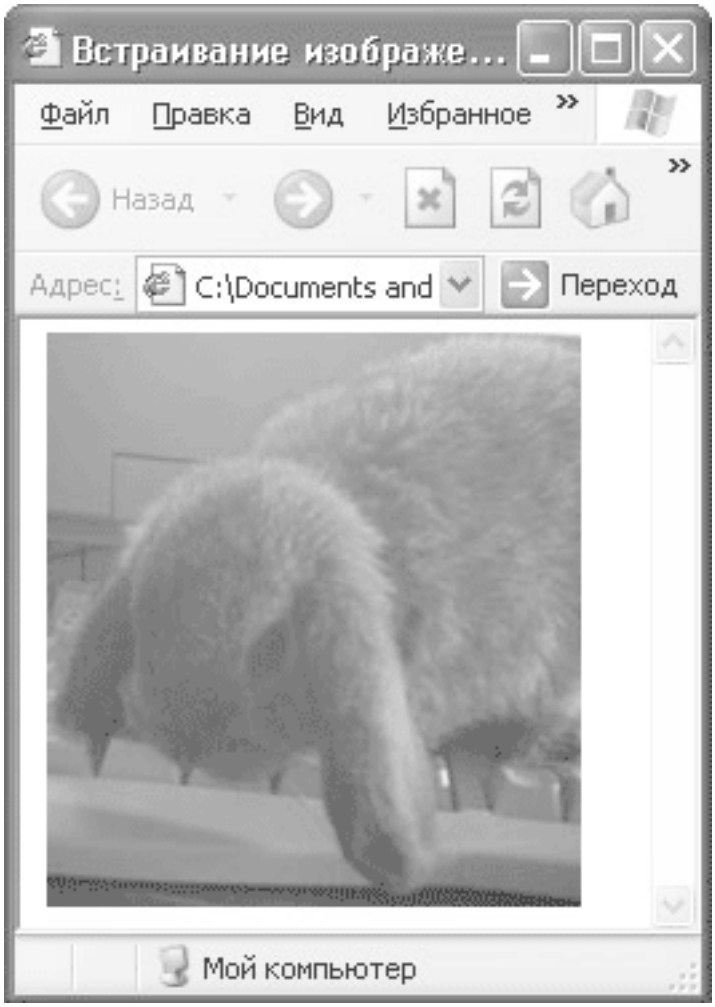

Рис. 4.1. Вставка изображения

Рисунок на странице отображается в реальном размере. Это простейший вариант вставки картинки, его можно использовать, когда нет необходимости ни в какой трансформации рисунка, например, если рисунок будет единственным элементом на странице.

Один рисунок на странице встречается нечасто, обычно на странице, помимо рисунка, присутствуют другие объекты, и большие размеры картинки становятся проблемой, но ничего страшного – размеры изображения можно легко подкорректировать.

Размер изображения

Если необходимо значительно изменить размер изображения, то лучше использовать специальные программы, однако в небольших пределах допустимо использовать и атрибуты элемента IMG.

Чтобы редактировать размер картинки, используют атрибуты width и height. Их значения можно указывать в пикселах или процентах от размера окна (в этом случае после размера ставится знак %).

Примечание

При изменении размеров окна картинка, размер которой указан в процентах, тоже меняет размер.

Можно указать только один из атрибутов, и тогда второй будет вычисляться автоматически для сохранения пропорций рисунка.

Меняя размер картинки с помощью атрибутов элемента IMG, внимательно следите за изображением, есть вероятность того, что рисунок исказится. Сам браузер не обрабатывает картинки под новый размер, поэтому, если размер выставлен неаккуратно, из произведения искусства рисунок может превратиться во что-то непонятное (особенно если задать размер больше, чем в реальности). Искажение пропорций тоже не приводит к улучшению качества изображения, как видно на рис. 4.2.

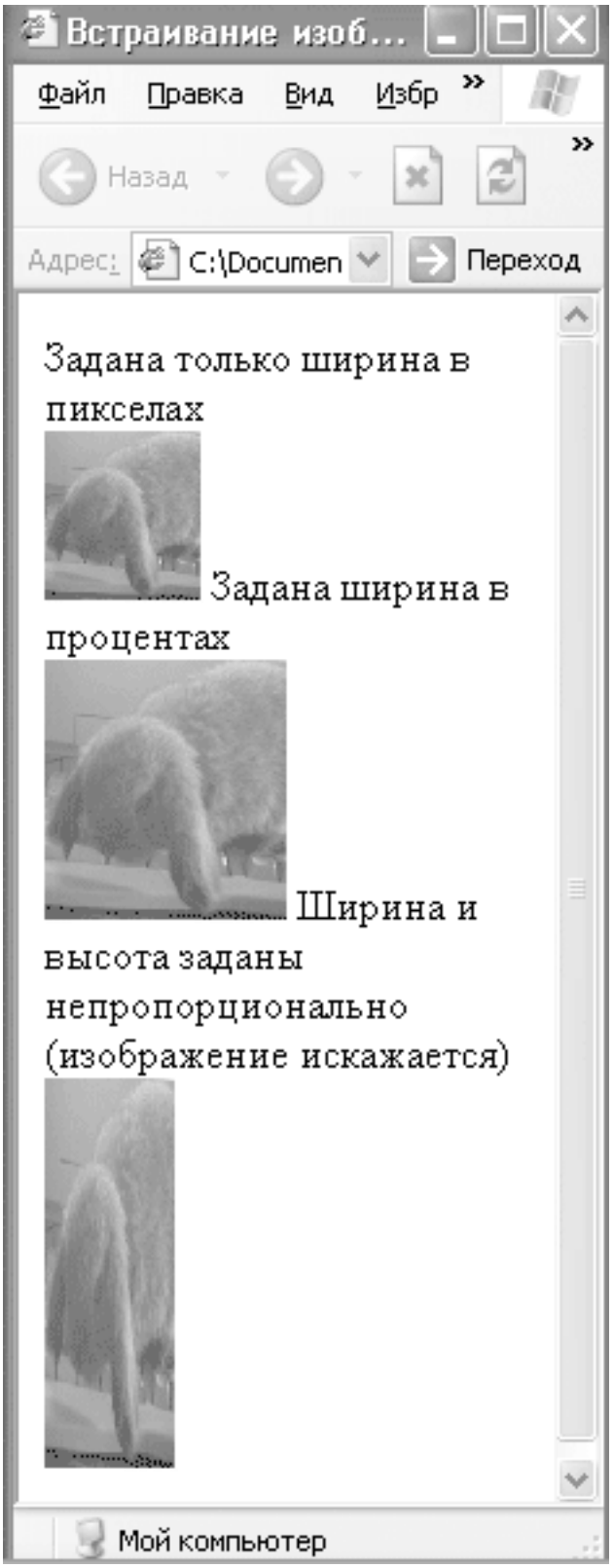

Рис. 4.2. Размеры изображения

В листинге 4.2 показан пример встраивания изображения с заданными размерами.

Листинг 4.2. Задание размеров изображения

```
<html>
<head>
<title>Встраивание изображения</title>
</head>
<body>
Задана только ширина в пикселах <br/>
<img src="image.jpg" width="60"/>
Задана ширина в процентах<br/>
<img src="image.jpg" width="50%"/>
Ширина и высота заданы непропорционально (изображение искажается)<br/>
<img src="image.jpg" width="50" height="150"/>
</body>
</html>IMG_0628.jpg
```

В листинге 4.2 для первого изображения задана только ширина, высота вычисляется автоматически. Для второго изображения размер определен в процентах, а для третьего установлены неверные размеры, которые искажают картинку.

Результат работы листинга 4.2 показан на рис. 4.2.

На рис. 4.2 видно, что размеры все-таки имеют значение, особенно для картинок, поэтому следить за их изменением нужно очень внимательно. Если нет необходимости в изменении размеров, то задавайте размеры картинки, соответствующие реальности. Это позволит избежать искажений и ускорит обработку картинки браузером.

Совет

Если вам нужно значительно изменить размер картинки, то используйте специальные программы для работы с изображениями. Они смогут проделать эту операцию, минимально исказив картинку или вовсе без искажений.

Помимо размеров картинки, на внешний вид сайта влияет расположение изображений.

Выравнивание изображения

Расположение картинки влияет на общий вид страницы, на восприятие текста вокруг нее. Удобство чтения текста, находящегося около картинки, сильно зависит от их взаимного расположения.

Есть множество вариантов выравнивания картинок относительно текста, и за все отвечает атрибут align элемента IMG. Он позволяет выравнивать изображения с правой, с левой стороны окна или относительно элементов строки.

У атрибута align много значений, которые позволяют установить картинку именно так, как надо, и именно там, где надо.

Горизонтальное выравнивание:

- left – по левому краю;
- right – по правому краю.

Вертикальное выравнивание:

- top – выравнивание верхней границы картинки по самому высокому элементу строки;

• texttop – выравнивание верхней границы картинки по самому высокому элементу текста;
• middle – середина изображения выравнивается по базовой линии строки;
• absmiddle – середина изображения выравнивается по середине строки;
• baseline – выравнивание нижней границы изображения по базовой линии строки;
• bottom – аналогично baseline;
• absbottom – нижняя граница изображения выравнивается по нижней границе текущей строки.

Примечание

Базовая линия строки – это линия, на которой расположены все элементы. При этом некоторые буквы выступают за эту линию, например буква «p». Ее палочка заканчивается ниже базовой линии и будет самым нижним элементом строки. Заглавные буквы, наоборот, выступают сверху этой линии.

В листинге 4.3 приведены примеры выравнивания картинок относительно текста по вертикали.

Листинг 4.3. Выравнивание картинки по вертикали

```
<html>
<head>
<title>Встраивание изображения</title>
</head>
<body>
<img src="image.jpg" width="70" align="top"/>Выравнивание по самому верхнему элементу в строке<br/>
<img src="image.jpg" width="70" align="absbottom"/>Нижняя граница изображения выравнивается по нижней границе текущей строки<br/>
<img src="image.jpg" width="70" align="bottom"/>Нижняя граница изображения выравнивается по базовой линии строки<br/>
```

Конец ознакомительного фрагмента.