Н. П. Кисленко

HTML

САМОЕ НЕОБХОДИМОЕ

Форматы разметки и интернет-технологии

Работа с текстовым содержимым

Ссылки и навигация по сайту

Списки, таблицы, графика, фреймы

Формы и стили

Подключение апплетов, скриптов, видео, роликов Flash

Управление сайтом и оптимизация

Николай Кисленко

HTML

САМОЕ НЕОБХОДИМОЕ

Санкт-Петербург

«БХВ-Петербург»

2008

УДК 681.3.06
ББК 32.973.26-018.2
К44

Кисленко Н. П.

К44 HTML. Самое необходимое. — СПб.: БХВ-Петербург, 2008. — 352 с.: ил. + CD-ROM

ISBN 978-5-9775-0169-9

Даны основы языка HTML. На практических примерах рассмотрены современные технологии использования гипертекста, работа с текстовым содержимым Web-страниц, ссылками, списками, таблицами, графикой, формами, стилями, мультимедиа и другими элементами сайта. Уделено внимание интерактивным технологиям HTML, приемам грамотной работы с содержимым и организации управления интернет-проектом.

Прилагаемый CD содержит более 100 примеров из книги, русскоязычную документацию по HTML и стилевым таблицам, а также удобный в обращении электронный справочник.

Для широкого круга пользователей

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

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

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 25.12.07.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 28,38.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

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

ISBN 978-5-9775-0169-9

© Кисленко Н. П., 2007
© Оформление, издательство "БХВ-Петербург", 2007

Оглавление

Введение **1**

Зачем нужно изучать HTML? 1
Для кого предназначена эта книга? 3
Краткое содержание книги 4

ЧАСТЬ I. HTML В ИНТЕРНЕТ-ТЕХНОЛОГИЯХ **7**

Глава 1. HTML и Интернет **9**

1.1. Интернет-серверы и клиенты 9
1.2. Прикладные протоколы, IP-адреса и доменные имена 11
1.3. WWW и Web-страницы 15
1.4. Адреса URL 16
1.5. Браузеры и совместимость Web-страниц 18
1.6. Надстройки и плагины 25

Глава 2. HTML и программирование **27**

2.1. Технологии и языки клиентского программирования 28
2.2. Технологии и языки серверного программирования 33

Глава 3. HTML и другие форматы **38**

ЧАСТЬ II. СОЗДАНИЕ ЭЛЕКТРОННЫХ ДОКУМЕНТОВ СРЕДСТВАМИ HTML **49**

Глава 4. Структура документа **51**

4.1. Тег и его структура 52
4.2. Общая структура документа 53
4.3. Наш первый документ 57
4.4. Метатеги и их назначение 58
4.5. Теги для связи документов 60

Глава 5. Форматирование текста 62

5.1. Абзацы, выравнивание, переносы 63
5.2. Физическое форматирование — шрифты и начертания 67
5.3. Логическое форматирование — расставляем акценты 72
5.4. Предварительно отформатированный текст 75
5.5. Специальные символы 75
5.6. Текстовые блоки и разделы 77
5.7. Горизонтальные линии 79

Глава 6. Гиперссылки — основа Web 80

6.1. Как устроена ссылка 80
6.2. Абсолютные и относительные ссылки 82
6.3. Внешние и внутренние ссылки 85
6.4. Ссылки на различные виды ресурсов 87
6.5. Организация навигации по сайту 89
6.6. Организация коллекций ссылок 93

Глава 7. Списки 97

7.1. Списки — инструмент структурирования информации 97
7.2. Нумерованные, маркированные и многоуровневые списки 98
7.3. Списки определений и другие типы списков 103

Глава 8. Таблицы — основной инструмент Web-дизайнера 106

8.1. Общие принципы табличного представления информации 106
8.2. Возможности таблиц. Все о теге *<table>* 108
8.3. Вложение таблиц 125
8.4. Организация блоков на сайте с помощью таблиц 128
8.5. Программное обеспечение для проектирования Web-страниц 134

Глава 9. Цвета и управление ими 138

9.1. Представление цветов в HTML 138
9.2. Основной цветовой набор документа 141
9.3. Изменение цвета шрифта 142
9.4. Изменение цвета в ячейках таблицы 144
9.5. Создание цветовой модели сайта 145

Глава 10. Графика 148

10.1. Общие принципы работы с графикой 148
10.2. Графические форматы Интернета 153
10.3. Вставляем картинки или все о теге *<img>* 156

10.4. Использование фоновых рисунков 163
10.5. Графические средства навигации 165
10.6. Организация коллекций картинок 168
10.7. Организация блоков с использованием графики 171
10.8. Программное обеспечение для работы с графикой 174

Глава 11. Картированные изображения 177

11.1. Преимущества и недостатки картирования 177
11.2. Конфигурация клиентского изображения-карты 178

Глава 12. Фреймы 183

12.1. Применение фреймов: за и против 183
12.2. Описание коллекций фреймов 185
12.3. Связь между фреймами и навигация с их помощью 190
12.4. Плавающие фреймы 192
12.5. Программное обеспечение для работы с фреймами 194

ЧАСТЬ III. РАЗРАБОТКА ИНТЕРАКТИВНЫХ ДОКУМЕНТОВ СРЕДСТВАМИ HTML 195

Глава 13. Формы: создаем обратную связь 197

13.1. Область применения форм 197
13.2. Теги для организации форм 198
13.3. Отправка данных по электронной почте и внешнему приложению 207
13.4. Дизайн форм 212

Глава 14. Таблицы стилей — управляем сайтом легко 213

14.1. Общие свойства таблиц стилей 213
14.2. Способы определения стилей 215
14.3. Виды селекторов. Определение классов 217
14.4. Свойства шрифта, текста и цветов 225
14.5. Свойства заполнителей, границ и рамок 231
14.6. Позиционирование элементов 235
14.7. DHTML и создание визуальных эффектов 240

Глава 15. Мультимедиа и внешнее содержимое 243

15.1. Тег *<object>* 243
15.2. Подключение звуковых файлов 247
15.3. Подключение видео 251
15.4. Использование скриптов 254

15.5. Внешние баннеры и информеры....258
15.6. Использование Java-апплетов....260
15.7. Использование технологии Macromedia Flash....263
15.8. Программное обеспечение для работы с мультимедиа....265

ЧАСТЬ IV. ОТ HTML К САЙТУ....267

Глава 16. Делаем сайт правильно....269

16.1. Прежде чем делать сайт....269
16.2. Управление структурой сайта и оптимизация HTML-кода....272
16.3. Выбор хостинга....278
16.4. Поисковые системы и ранжирование результатов поиска....281
16.5. Внешние ссылки и увеличение ссылочной популярности....285

Глава 17. Как делать не нужно?....290

17.1. Ошибки структуры сайта....290
17.2. "Неправильное" содержимое....293
17.3. Ошибки при работе с графикой....294
17.4. Средства для проверки сайта....296

ПРИЛОЖЕНИЯ....299

Приложение 1. Краткий справочник по тегам HTML....301

Приложение 2. Основные специальные символы HTML....329

Приложение 3. Названия и коды цветов в HTML....336

Приложение 4. Полезные ссылки Интернета....339

Русскоязычные ресурсы....339
Англоязычные ресурсы....340

Приложение 5. Описание компакт-диска....341

Предметный указатель....343

Введение

Зачем нужно изучать HTML?

Эта книга предназначена для изучения формата разметки гипертекста HTML (HyperText Markup Language) — основного "языка Интернета", на котором написано и пишется большинство имеющихся в Сети страниц. Возможно, вы уже ответили для себя на вопрос, вынесенный в заголовок, но книге полагается иметь введение, и я скажу в нем самое необходимое.

"HTML устарел, и знать его не нужно, как не нужно знать кодировки символов для чтения текста с экрана" — автору приходилось слышать и такое мнение. Однако мы собираемся не читать, а *писать* Web-странички, и без знания HTML — главной "кодировки" всемирной Сети — нам не обойтись.

Я уверен, что, несмотря на обилие программ для разработки сайтов, современному человеку знать основы HTML так же естественно, как уметь пользоваться мобильным телефоном.

Давайте подумаем почему.

- HTML обеспечивает компактность и гибкость кода. Визуальный редактор, такой, как всем известный Microsoft Word, стремится сохранить внешний вид разработанной в нем Web-страницы неизменным, используя для этого все мощные средства стилевого форматирования (с ними мы познакомимся в *главе 14*). Между тем, HTML создавался с противоположной целью — сохраняя логику, связи и структуру документа, дать пользователю возможность *самому* настроить удобный для себя вид, масштаб и способ просмотра. Кроме того, объем Web-страницы, созданной в визуальном редакторе, будет всегда в десятки раз больше, чем страницы с теми же возможностями, но написанной "вручную".
- Стремительно происходящая "мобилизация" Интернета — также аргумент в пользу изучения HTML. Все больше карманных компьютеров, коммуникаторов, смартфонов и даже просто "мобильников" имеют встроенный

браузер и поддерживают просмотр Web-страниц. Возможности этих устройств пока слишком слабы для полноценного просмотра огромных файлов, созданных в визуальном редакторе, но они вполне достаточны для загрузки компактного, сделанного "вручную" HTML-кода.

- HTML не устаревает, а постоянно развивается. Несмотря на то, что действующему стандарту HTML 4.01 уже около 10 лет, организация World Wide Web Consortium в 2007 году опубликовала проект новой, пятой версии стандарта, в которой HTML станет еще логичнее, удобнее и компактнее.
- HTML прост и эффективен. Несмотря на сегодняшнее обилие языков разметки, в котором мы попытаемся разобраться в *главе 3*, несмотря на то, что вычислительные мощности, пропускная способность сетей и возможности интернет-технологий, растут не по дням, а по часам, новые стандарты никак не вытеснят классического языка HTML. Все дело в том, что HTML можно изучить и успешно использовать, ничего не зная про "объектную модель документа", "идентификаторы классов" и прочие мало понятные неспециалисту вещи. Новые, значительно "усиленные" стандарты разметки обеспечивают гораздо бóльшую свободу обращения с Web-страницей — увы, ценой простоты ее устройства, независимости от программно-аппаратных средств и гибкости настроек просмотра под нужды пользователя. А сила HTML — именно в указанных факторах. Обычного HTML оказывается *достаточно* для решения большинства задач, связанных с размещением информации в Интернете, и именно это слово "достаточно" все объясняет.

И хотя мы часто будем называть получаемый нами продукт "HTML-кодом", на самом деле никакого "кодирования" здесь нет. HTML — простой текстовый формат, и это одно из главных его достоинств. Для работы с HTML и получения великолепных результатов достаточно обычного Блокнота Windows и имеющегося под рукой браузера.

С другой стороны, автор не станет утверждать, что альтернатив "ручной работе" нет совсем.

В этой книге мы дадим краткий обзор программ, полностью или частично берущих на себя труд по верстке HTML, оценим их возможности, достоинства и недостатки. Но главное следует сказать уже сейчас — там, где можно сэкономить время на ручном прописывании команд в огромном тексте, программы верстки HTML оказываются полезными и даже незаменимыми. Там же, где дело касается написания компактного, логически продуманного и "всюду работающего" кода Web-страницы, ничто пока не заменит качественной ручной работы. Да и готовые Web-страницы, встречающиеся вам, нередко нуждаются в правке, выполнить которую без знания HTML нельзя.

Следует избегать любого "фанатизма" и выбирать оптимальный способ — вот, пожалуй, наиболее здравая позиция.

Таким образом, HTML — прекрасный и, главное, доступный инструмент для реализации ваших творческих идей. Прочитав эту книгу, вы научитесь создавать Web-страницы и грамотно управлять их содержимым без больших затрат времени и изучения новых программ. Вы сможете добавить на свою страничку звук или видео, сделать на ней форму для "обратной связи" с пользователем, изготовить красивую графическую карту сайта и легко управлять его содержимым с помощью каскадных таблиц стилей CSS.

Для кого предназначена эта книга?

Сегодня все реже можно встретить человека, не видевшего персонального компьютера и ничего не слышавшего о Windows. Даже если ваш пользовательский опыт невелик — смело беритесь за изучение этой книги. Для работы с ней достаточно простейших навыков обращения с компьютером. Изложение материала ведется последовательно "от простого к сложному", так же шаг за шагом вам следует изучать книгу, если вы — новичок. Выполнение в браузере всех примеров с прилагаемого компакт-диска поможет вам быстро повысить свой уровень и почувствовать себя уверенней.

Книга будет полезна и более "продвинутым" пользователям, уже имеющим опыт создания или поддержки собственных Web-страниц. Удобные таблицы тегов и многочисленные примеры практически важных элементов сайта, краткий справочник по возможностям HTML, приведенный в *приложении 1*, сводки цветов и специальных символов языка, коллекция страничек на прилагаемом компакт-диске помогут вам систематизировать и углубить свои знания. Полезной для вас будет и *часть IV*, описывающая общие принципы создания сайта и управления им.

Не правда ли скучно, когда в Сети существует множество однообразных сайтов, сделанных на основе двух десятков популярных систем управления содержимым? Сегодня зачастую дело обстоит именно так, и только освоение HTML как можно бóльшим числом специалистов может изменить ситуацию к лучшему.

Кроме того, работа с развитыми системами управления сайтом обычно предполагает наличие HTML в текстах статей, помещаемых системой на Web-страницы. Поэтому глубокое знание принципов HTML даст вам дополнительное преимущество и в развитии сайта, работающего на "движке" (так называют в обиходе системы управления сайтом, обычно написанные на языке программирования PHP с помощью баз данных MySQL).

Наверное, эта книга понадобилась вам не только для того, чтобы получить "сухой" набор фактов. Практические примеры реализации возможностей HTML, готовые рецепты организации популярных разделов сайта (таких, как система меню, фотогалерея или коллекция ссылок) приводятся в большинстве глав. Все представленные примеры можно сразу же увидеть в работе, если не забывать про приложенный к книге компакт-диск.

Краткое содержание книги

Книга состоит из четырех частей.

В *части I* я старался показать "место HTML в мире". Как устроен Интернет? Какой обозреватель-браузер следует предпочесть? Какие средства есть для программирования Web-страничек? Существуют ли другие языки разметки, кроме HTML? Как они связаны друг с другом и как развиваются? Все эти вопросы кратко освещаются в первой части, и чтение ее будет особенно полезно всем тем, кто хочет упорядочить свои знания об интернет-технологиях.

Если вы хотите немедленно приступить к изучению HTML, то начните с *частей II* и *III*.

Часть II расскажет о создании обычных *статических* Web-страничек. Хотя что значит "обычных"? Комбинируя текст, списки, таблицы, графику, фреймы и другие описанные здесь элементы, вы сможете создать действительно привлекательные, информативные и "живые" страницы.

Изучив *часть III*, читатель сможет дополнять свои страницы *динамическими* элементами. Формы для "обратной связи" с пользователем или организации поиска по сайту, стили для быстрого и удобного оформления множества страничек сразу, подключение звука, видео, внешних программ, — вот неполный перечень того, о чем рассказывается в третьей части.

Сеть перегружена информацией, среди которой немного действительно важной и грамотно подготовленной. Поэтому в своей книге я обратил внимание не только на изложение собственно возможностей HTML, но и на способы их правильного применения. Этому особенно актуальному в последнее время вопросу посвящена *часть IV* книги, где рассказывается, как писать теги и связывать страницы, чтобы результат воспринимался и людьми, и поисковыми системами. Здесь же рассматриваются распространенные ошибки, допускаемые Web-мастерами при создании сайтов.

Не привлекая сложных специализированных технологий, которые используют специалисты по "раскрутке" сайтов, мы научимся писать HTML-код, дос-

тойно и профессионально выглядящий в Сети даже без новейшей 3D-графики и рекламных "вливаний" в сайт.

Важная составляющая — прилагаемый к книге компакт-диск. По тексту встречается немало ссылок на примеры с диска, просмотр и изучение которых поможет вам быстро освоить особенности HTML.

Книга содержит более 100 примеров, как показывающих применение отдельных команд HTML, так и дающих образцы целых сайтов. Все примеры и полезные дополнительные сведения вы также найдете в папках компакт-диска.

Наконец, пять приложений содержат множество необходимой информации: краткий, но емкий справочник по всем командам HTML, таблицы специальных символов и цветов Интернета, а также полезные ссылки по теме книги.

Я надеюсь, что чтение этой книги будет для вас приятным и полезным. Связаться со мной и высказать свои отзывы, предложения или пожелания вы можете через контактную информацию со странички **http://pers.narod.ru/guest.html**.

ЧАСТЬ I

HTML
В ИНТЕРНЕТ-ТЕХНОЛОГИЯХ

Глава 1

HTML и Интернет

Темпы роста всемирной компьютерной сети поражают воображение. Число компьютеров в Интернете и объемы доступной информации стремительно увеличиваются, и никто сегодня уже не может назвать точного количества пользователей Сети. При этом именно формат HTML — базовая среда, в которой размещаются все остальные компоненты бескрайнего Web-пространства.

Часть I книги станет вашим лоцманом в море современных Web-технологий. Она призвана дать представление об Интернете как тем читателям, которые пока чувствуют себя новичками в виртуальном мире, так и тем, кто нуждается в систематизации и упорядочении своих знаний об интернет-технологиях.

Глава 1 содержит краткое описание основных служб и сервисов Интернета, принципов адресации документов в Сети, а также популярных программ-браузеров для работы с Web-страницами.

1.1. Интернет-серверы и клиенты

Упрощенно Интернет можно представить себе как сеть связанных между собой и разбросанных по всему миру компьютеров, которые являются *клиентами* и *серверами* Сети, способными обмениваться информацией по определенным правилам. Сеть не представляет собой единое целое и никому не принадлежит, но при этом более мелкие сети, подключенные к Интернету, обслуживаются отдельными организациями-*провайдерами*, являющимися собственниками "своего" участка Сети и получающими плату за предоставление доступа к ней.

С физической точки зрения Интернет можно сравнить с сетью автодорог или с кровеносной системой человека. Сеть состоит из каналов разной пропускной

способности, и более мелкие и "тихоходные" каналы вливаются в более крупные и скоростные.

Под *сервером* (от англ. to serve — служить) в информационных технологиях понимается программа, занятая обслуживанием клиентов и выполняющая различные функции по их запросам, а также предоставляющая клиентам доступ к информационным или вычислительным ресурсам. Часто сервером называют и сам компьютер с соответствующим программным обеспечением.

Клиентом называется аппаратный или программный компонент вычислительной системы, посылающий запросы серверу и в ответ получающий от него данные.

Программа-клиент взаимодействует с сервером, используя определенный *протокол*, устанавливающий правила обмена данными. Клиент может запрашивать с сервера какие-либо документы, манипулировать данными непосредственно на сервере, запускать на нем новые программы и т. п. Полученные от сервера данные клиент может предоставлять пользователю или формировать на их основе новый запрос. Клиент и сервер могут работать как на одном и том же компьютере, так и на разных. Во втором случае для обмена информацией между ними используется сетевое соединение.

Понятия "клиент" и "сервер" описывают лишь *распределение ролей* при выполнении конкретной задачи, но не какие-либо отдельные виды компьютеров или программ. На одном и том же компьютере могут одновременно работать программы, выполняющие как клиентские, так и серверные функции. Например, *Web-сервер*, обслуживающий ваше подключение к Интернету, может, в качестве клиента, получать данные для формирования страниц от сервера баз данных.

В сетевой *архитектуре "клиент — сервер"* все устройства являются либо клиентами, либо серверами. В каждый момент времени клиент (front end) — это запрашивающий компьютер, а сервер (back end) — компьютер, отвечающий на запрос. При этом сервер обычно является *удаленным приложением*, выполняющимся на компьютере провайдера или управляющем компьютере локальной сети, а клиент — *локальным приложением*, выполняющимся на компьютере пользователя. Единственный клиент, который потребуется нам для работы с ресурсами Интернета и для тестирования собственных страничек, — установленная на вашем компьютере программа-*браузер*. Современные браузеры и работа с ними описаны в *разд. 1.5*.

Что касается приложений, реализующих возможности Web-серверов, наиболее распространены два их них.

- Apache (произносится "апач") — Web-сервер с открытым исходным кодом. Не первое десятилетие он поддерживается сообществом разработчиков

под эгидой Apache Software Foundation и включен во многие популярные программные продукты. Основными достоинствами Apache считаются компактность приложения, надежность и гибкость конфигурации сервера. Работа с Apache требует профессионализма, его недостатком чаще всего называют отсутствие удобного стандартного интерфейса для администратора. Тем не менее, в настоящее время Web-сервер Apache обслуживает более половины всех сайтов.

- IIS (Internet Information Services) — набор серверов, разработанных компанией Microsoft. IIS распространяется с операционными системами семейства Windows и может быть установлен как их компонент. IIS включает в себя Web-сервер, позволяющий размещать сайты в Интернете, а также дополнительные службы для предоставления доступа к файлам, отправки электронной почты и управления группами новостей. Один сервер IIS может обслуживать несколько сайтов, отличающихся между собой адресами, именами и настройками. При этом для каждого сайта указывается каталог в файловой системе сервера, соответствующий "домашней странице" сайта. На сегодняшний день примерно треть сайтов в мире обслуживаются Web-сервером IIS.

Все средства поддержки сетевых технологий, как со стороны сервера, так и со стороны клиента, должны соответствовать стандартам, разработка которых ведется под управлением международного консорциума W3C (World Wide Web Consortium, сайт **www.w3.org**).

1.2. Прикладные протоколы, IP-адреса и доменные имена

Традиционно различают два основных способа подключения к Интернету:

- подключение с удаленного компьютера по выделенной или обычной телефонной линии посредством внешнего или встроенного в компьютер *модема*. Связь с сервером провайдера осуществляется телефонным звонком через модем по указанному провайдером номеру; кроме номера телефона, для входа в сеть необходимы *логин* — т. е. регистрационное имя пользователя, и пароль. Основное преимущество способа — простота и доступность подключения, недостаток — невысокая скорость связи, ограниченная пропускной способностью телефонной линии;
- подключение через локальную сеть. В этом случае машина связана с внешним миром через установленную в системном блоке плату *сетевого адаптера*, а специально выделенный сервер управляет подключением к Интернету всех клиентов локальной сети. Скорость такого соединения

значительно выше, но компьютер должен быть физически подключен к сети отдельным кабелем, что существенно дороже. В последнее время Россия переживает период бурного развития локальных сетей, что позволяет надеяться на дальнейшее распространение доступных и качественных интернет-услуг.

Компьютеры, подключенные к Интернету, могут иметь самую различную архитектуру и программное обеспечение. Возможность обмена информацией и совместимость достигаются за счет использования коммуникационных *протоколов*, т. е. наборов правил, касающихся передачи информации по сетям. Главный протокол Интернета называется TCP/IP (Transmission Control Protocol — Internet Protocol). На самом деле под названием TCP/IP скрывается целое семейство протоколов, решающих те или иные частные задачи. Перечислим основные из них:

- *транспортные протоколы* TCP и UDP (User Datagram Protocol) — управляют процессом передачи данных между компьютерами;
- *протоколы маршрутизации* IP, ICMP (Internet Control Message Protocol), RIP (Routing Information Protocol) — обрабатывают адресацию данных, обеспечивают их физическую передачу и отвечают за выбор наилучшего маршрута до адресата;
- *протоколы поддержки сетевого адреса* DNS (Domain Name System), ARP (Address Resolution Protocol) — обеспечивают идентификацию компьютера в сети по его уникальному адресу;
- *шлюзовые протоколы* EGP (Exterior Gateway Protocol), GGP (Gateway-to-gateway protocol), IGP (Interior Gateway Protocol), GCP (Gateway Control Protocol) — отвечают за передачу информации о маршрутизации данных и состоянии сети, а также обрабатывают данные для взаимодействия с локальными сетями;
- *протоколы прикладных сервисов* FTP (File Transmission Protocol), Telnet и др. — обеспечивают доступ к различным услугам и службам Сети (например, передачу файлов между компьютерами);
- *протокол SMTP* (Simple Mail Transfer Protocol) — отвечает за передачу сообщений электронной почты;
- *протокол NNTP* (Network News Transport Protocol) — предназначен для работы с группами новостей.

Каждый компьютер, подключенный к Интернету, имеет уникальный адрес, называемый *IP-адресом* (читается "ай-пи"). IP-адрес компьютера может быть постоянным или каждый раз назначаться сервером при соединении с Сетью, но всегда один IP соответствует одному компьютеру. IP-адрес имеет длину

4 байта (по сетевой терминологии, *октета*) или 32 бита. При записи адреса октеты разделяют точками. Примерами реально существующих IP-адресов могут служить такие числовые комбинации, как 82.137.152.21, 204.146.46.133 или 207.68.137.53.

Поскольку одним байтом можно представить числа от 0 до 255 включительно, теоретически получается более 4 миллиардов возможных адресов компьютеров! На самом деле, некоторые комбинации адресов зарезервированы и это число меньше. Так, адрес 127.0.0.1 используется только на локальной машине для обращения к установленному на ней Web-серверу.

Любой IP-адрес состоит из двух частей: *адреса сети* и *адреса хоста* (*хостами* называют подключенные к Интернету компьютеры и некоторые другие сетевые устройства). В зависимости от того, какая часть IP выделена под адрес сети, различают сети и IP-адреса классов A, B и C. Существуют также служебные классы D и E. Символически представив IP в виде w.x.y.z, приведем информацию о классах сетей в табл. 1.1.

Таблица 1.1. *IP-адреса сетей классов A, B, C*

Класс сети	Значения первого октета	Октеты номера сети	Октеты номера хоста	Возможное число сетей	Возможное число хостов в сети
A	1-126	w	x.y.z	126	16777214
B	128-191	w.x	y.z	16384	65534
C	192-223	w.x.y	z	2097151	254

Итак, по первому октету можно судить о классе сети. Возможно определенное число сетей каждого класса, например, при современной структуре IP может существовать не более 126 гигантских подсетей класса A, каждая из которых может объединить до 16,7 млн. компьютеров.

Чтобы отделить адрес сети от адреса хоста, используется *маска подсети*, также представляющая собой 32-битное число. По умолчанию сетям класса A соответствует маска 255.0.0.0, класса B — 255.255.0.0, а сетям класса C — 255.255.255.0, т. е. в двоичном представлении маски, позиции, соответствующие адресу сети, закрыты единицами. Маска подсети может служить и для других целей, например, для логического разделения локальных сетей на подсети меньшего масштаба.

Доменное имя также представляет собой уникальный адрес компьютера в сети, но для удобства пользователей цифры в нем заменены наборами символов, разделенными точками. Доменное имя состоит из нескольких иерар-

хически расположенных *доменов*, а под доменом понимают просто поименованный набор хостов, объединенных по территориальному или организационному признаку.

Правила составления доменных имен менее жесткие, чем при назначении IP, но и здесь есть определенная структура. Например, доменное имя **www.alpha.msk.ru** включает в себя следующие части:

- www — префикс, указывающий на принадлежность сервера "всемирной паутине" World Wide Web; необязателен, но широко распространен в доменных именах;
- alpha — домен третьего уровня, в данном случае содержащий имя организации;
- msk — домен второго уровня (здесь — территориальный домен Москвы);
- ru — домен верхнего уровня (территориальный домен России).

Таким образом, если исключить префикс www, доменное имя записывается "снизу вверх", от локальных доменов — к глобальным.

Организация, владеющая некоторым доменом, обычно может создавать для него *субдомены* более низкого уровня, например, **www.subdomain.domain.ru** для **www.domain.ru**. Альтернатива — создание вложенных папок в корневой папке сервера (см. *разд. 1.4*): **www.domain.ru/subdomain**.

Домен верхнего уровня может обозначать не только территорию, но и *принадлежность* организации, владеющей хостом. Основные домены верхнего уровня приведены в табл. 1.2.

Таблица 1.2. *Основные домены верхнего уровня*

Домен	Принадлежность хостов домена
com	Коммерческие организации
edu	Образовательные учреждения
gov	Правительственные учреждения
int	Международные организации
mil	Военные учреждения
net	Организации, управляющие Интернетом или входящие в его структуру
org	Прочие организации

В последние годы к классическим доменам, описанным в табл. 1.2, прибавились новые доменные зоны: info, museum, name, pro, aero, biz и coop.

За созданием и работой всех доменов верхнего уровня следит международная некоммерческая организация ICANN (**www.icann.org**). Домены верхнего уровня существуют практически для всех признанных ООН государств и территорий. Все они состоят из двух букв, а их поддержкой в каждой стране по поручению ICANN занимается специальная организация. В России такой организацией до начала 2005 года являлся РосНИИРОС (**www.ripn.net**), в настоящее время действует несколько регистраторов, список которых можно найти в Интернете по адресу **http://www.cctld.ru/ru/regru/information/listrecorder**.

Сокращения, аналогичные приведенным в табл. 1.2, могут применяться в доменах более низких уровней для того, чтобы подчеркнуть принадлежность организации, так, сервер правительства России в Сети имеет адрес **www.gov.ru**. При некотором навыке работы в Сети нетрудно научиться по доменному имени определять примерную принадлежность хоста.

Доменные имена преобразуются в понятные для компьютера IP-адреса при помощи *системы доменных имен* DNS (Domain Name System), состоящей из иерархии *DNS-серверов*, на вершине которой находятся *серверы корневой зоны* с именами a.root_server.net, b.root_server.net и т. д., дублирующие информацию друг друга. Локальный сервер, получив от машины-клиента запрос на соединение с некоторым адресом, передает его локальному DNS-серверу, который выделит из запроса доменное имя и либо найдет соответствующий IP у себя в базе данных, либо обратится к одному из вышестоящих DNS-серверов. Последний вернет указатель на DNS-сервер известного ему домена, в который входит запрошенный адрес, и полностью устранится из процесса. Такие вложенные запросы могут повторяться, причем каждый раз локальный DNS-сервер будет обращаться к серверу имен все более низкого уровня. Только после окончания этого многоступенчатого процесса DNS-сервер вернет преобразованный адрес компьютеру, сделавшему запрос, и пользователь сможет, наконец, увидеть на своем мониторе, что же за информация расположена по введенному им адресу.

Доменные имена не только более понятны, чем IP-адреса, но и более универсальны, их проще переназначать и использовать повторно, а один хост, имеющий один IP-адрес, вполне может обладать несколькими доменными именами. В Сети существуют службы, предоставляющие бесплатно или за плату доменные имена третьего или второго уровня.

1.3. WWW и Web-страницы

World Wide Web или "всемирная паутина" — ведущий сервис Интернета, постепенно вытесняющий или включающий в себя большинство других сетевых служб. Прикладной протокол, используемый в WWW, называется *HTTP* (Hypertext Transfer Protocol), что переводится как "протокол передачи гипер-

текста". Документы, составляющие содержание WWW, называются *Web-страницами*, а формат, с помощью которого подготавливаются Web-страницы, — *HTML* (HyperText Markup Language) или "язык разметки гипертекста". Под *гипертекстом* в простейшем случае понимается текст, допускающий не только последовательное прочтение, но и связанный с другими текстами с помощью указателей-ссылок.

Формат HTML представляет собой достаточно простой набор команд, которые описывают структуру документа. HTML позволяет выделить в тексте отдельные логические части (заголовки, абзацы, списки и т. д.), поместить на Web-страницу отдельно подготовленную фотографию или картинку, организовать на странице ссылки для связи с другими документами, но не задает конкретные и точные атрибуты форматирования документа, как, например, Microsoft Word. Конкретный вид документа окончательно определяет только программа-*браузер* на вашем компьютере (см. *разд. 1.5*). Необходимость именно такого подхода связана, опять же, с разнородностью аппаратного и программного обеспечения компьютеров, подключенных к Интернету.

С точки зрения пользователя Windows, Web-страница — это файл с расширением htm или html, находящийся на сервере Интернета, в локальной сети или на жестком диске вашей машины. Этот файл *текстовый*, т. е. он может быть открыт в любой программе просмотра неформатированного текста, такой как стандартная программа Блокнот. Все команды разметки HTML, называемые *тегами*, также представляют собой строки специально оформленного текста.

Хотя формат HTML часто называют *языком*, следует понимать, что он не является языком программирования. Прежде всего, это связано с тем, что документ HTML не позволяет задавать последовательность действий, но лишь указывает, как следует расположить и оформить то или иное содержимое. Мы будем придерживаться термина *HTML-код* в отношении содержимого Web-страницы.

Тем не менее, тесно связанные с HTML *языки сценариев*, о которых рассказано в *главе 2*, дают возможность полноценного программирования Web-страниц. Кроме того, многие из существующих Web-страниц сами являются результатом работы формирующих их серверных программ.

1.4. Адреса URL

Поскольку любой компьютер в Интернете имеет свой уникальный адрес, представимый в виде IP или доменного имени, каждый файл, расположенный в какой-либо папке на таком компьютере, также обладает уникальным адре-

сом, называемым URL (Uniform Resource Locator; принято читать "урл"). Полный URL документа в Сети состоит из следующих частей:

- *префикс протокола*, указывающий на используемую службу Интернета, состоит из его имени, двоеточия и двух символов "/". Основных прикладных протоколов, с которыми вы столкнетесь, работая в WWW, не так много. Кроме протокола http://, обеспечивающего доступ к Web-страницам, нам могут понадобиться протоколы ftp:// для обмена файлами с удаленным компьютером, file:// для адресации Web-страниц, расположенных на локальной машине и mailto: для доступа к электронной почте (после mailto не указываются два символа "//");
- *доменное имя компьютера* или его IP-адрес вместо доменного имени;
- *номер порта*, через который происходит взаимодействие с сервером. Перед номером порта ставится двоеточие. С точки зрения пользователя, указание порта бывает полезно нечасто, например, сервер может быть настроен на принудительную перекодировку документа при обращении к различным портам. Так, URL-адреса **http://www.host.ru:8100** и **http://www.host.ru:8101** могут вести на один и тот же сервер, но в первом случае документ читается в кодировке KOI-8R, а во втором — в кодировке Windows-1251;
- *имя файла* на сервере, которое может включать путь от корневого каталога сервера. В записи пути по дереву каталогов сервера употребляется символ "/", а не "\", как принято в Dos и Windows. Заметим, что корневой каталог (корневая папка) на сервере — это не головной каталог диска, как на локальной машине, а при соединении с сервером мы получаем доступ не ко всем его папкам и файлам, а только к тем, которые на нем специальным образом размещены и открыты для просмотра через WWW с помощью программы Web-сервера;
- *закладка*, позволяющая перейти в нужную часть документа. Имя закладки отделяется от имени файла символом "#" ("диез"). Если закладка не указана, предполагается переход к началу адресуемого документа.

В целях совместимости имена файлов, размещаемых в Интернете, обычно строятся по простейшим правилам, т. е. состоят только из латинских букв, цифр и символа подчеркивания. Web-страницам принято давать расширение html или htm.

Большинство серверов работают в операционных системах, различающих прописные и строчные буквы в названиях файлов и папок. Поэтому следует помнить, что URL *может быть* чувствителен к регистру символов и адреса **http://www.SOMEADDRESS.ru** и **http://www.someaddress.ru** могут обозначать не одно и то же.

Например, если один из файлов, размещенных на сервере, имеет полный URL **http://www.server.ru/folder/file.html**, можно заключить, что он находится на сервере **www.server.ru** в папке с именем folder.

Весьма часто вам встретятся URL, не содержащие имени html-файла, тем не менее, при вводе такого адреса, мы все же попадаем на конкретную Web-страницу. Это означает, что документ имеет *имя по умолчанию*, которое назначается при администрировании сервера. Чаще всего это имя index.html, так что URL **http://www.host.ru** может означать то же самое, что **http://www.host.ru/index.html**. Префикс протокола http://, принимаемого по умолчанию, при записи полного URL также обычно опускают. Браузер добавляет его автоматически при наборе URL-адреса в адресной строке.

URL могут иметь не только файлы с расширением html, но и любые другие файлы, размещенные в папках сервера, доступных из Интернета.

С практической точки зрения, "единицей измерения" Интернета является не отдельная Web-страница, а *Web-сайт* — набор тематически и стилистически связанных страниц и других ресурсов. Чаще всего, но не обязательно все страницы сайта располагаются в рамках одного домена.

1.5. Браузеры и совместимость Web-страниц

Браузер или *Web-обозреватель* (от англ. Web browser, по-русски иногда *броузер*) — это программа для просмотра Web-сайтов, т. е. для запроса Web-страниц из Интернета, их обработки, вывода и перехода от одной страницы к другой. Большинство браузеров также наделены способностями к просмотру оглавления файловых ftp-серверов, а некоторые включают в себя дополнительные компоненты для работы с электронной почтой и службами новостей. Разумеется, браузер может быть использован также для просмотра Web-страниц на локальном компьютере. Как правило, типы документов html и htm уже связаны с установленным в системе браузером, так что для просмотра скопированной или разработанной вами страницы достаточно сделать на ней двойной щелчок мышью.

Со времен зарождения WWW браузеры постоянно развивались и становились все более важной программой типичного персонального компьютера. Современный браузер — это комплексное приложение как для обработки и вывода различных составляющих Web-страницы, так и для предоставления интерфейса между Web-сайтом и его посетителями. Практически все популярные браузеры распространяются бесплатно или в комплекте с другими приложениями, например, браузер Internet Explorer входит в состав операционной системы Windows, Mozilla Firefox и Opera последних версий — бес-

платные программы, браузер Safari распространяется как часть операционной системы Mac OS.

Кроме этих распространенных браузеров существует множество "самодельных" и малоизвестных обозревателей, но обычно все они основаны на одном из нескольких популярных браузерных "движков", дополняя его новыми возможностями или дизайнерскими решениями.

Движок (жаргонизм от англ. engine — мотор, двигатель) или *ядро* браузера — это выделенная часть программного кода, реализующая основные возможности, необходимые для загрузки, интерпретации и просмотра html-документов. Применение готового движка при разработке собственного браузера, например, входящего в состав прикладной программы, не только намного сокращает время работы, но и позволяет уделить больше внимания созданию других подсистем, скажем, пользовательскому интерфейсу.

С другой стороны, программные продукты, разработанные с использованием готовых движков, наследуют все их ошибки и проблемы безопасности.

К наиболее известным браузерным движкам относятся Gecko (браузеры Mozilla), Trident (браузер Internet Explorer для Windows), KHTML (браузер Safari), Presto (браузер Opera).

Кратко опишем основные современные браузеры.

Самый распространенный Web-обозреватель на сегодняшний день — Internet Explorer (IE) — пожалуй, трудно найти пользователя, которому бы не приходилось с ним работать. Главная причина распространенности этого браузера — IE поставляется вместе с операционной системой Windows.

Пятая и шестая версии IE до сих пор очень популярны и входят в качестве стандартного браузера в различные сборки операционной системы Windows XP.

Хотя за разработчиками IE стоит вся интеллектуальная и финансовая мощь компании Microsoft, пользователи указанных версий традиционно жалуются на не слишком высокое быстродействие и нестабильность работы. Будем надеяться, что постоянно выполняемые фирмой Microsoft обновления браузера решат эти проблемы.

В 2006 году компания Microsoft порадовала пользователей давно ожидаемой седьмой версией браузера, которая поставляется в составе новой операционной системы Windows Vista. Впрочем, после проверки через Интернет подлинности вашей копии Windows, браузер можно установить и под Windows XP.

Основные нововведения Internet Explorer 7 — многовкладочное окно, ставшее на сегодня стандартом для большинства обозревателей, защита от фишинга (воровства личных данных), встроенная программа для чтения новостей, масштабирование страницы в широких пределах, расширенный режим

просмотра перед печатью, встроенный поисковый механизм с выбором сервера, произвольное масштабирование страницы и ряд других изменений. Лаконичный интерфейс в стиле Windows Vista скрывает под собой значительно переработанный и улучшенный браузер, поддерживающий огромный набор расширений и дополнительных модулей. Окно Internet Explorer 7 приведено на рис. 1.1, а страница обновлений браузера в Интернете имеет адрес **http://www.microsoft.com/ie**.

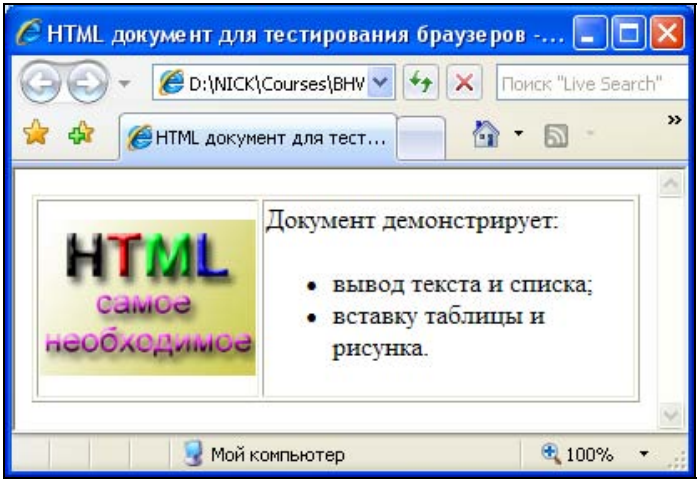

Рис. 1.1. Окно браузера Internet Explorer 7

Web-обозреватель Opera представляет собой компактный и во же время мощный пакет коммуникационных программ, состоящий из быстрого браузера с многодокументным интерфейсом (страницы открываются во вкладках одного окна), почтовой программы и модуля для получения новостей.

Opera предназначена скорее для опытных, чем для начинающих пользователей — так, даже за стандартной справкой вам придется выйти в Интернет и подключиться к сайту разработчика **www.opera.com**.

Из многочисленных улучшений браузера стоит отметить дополнительно загружаемые библиотеки, позволяющие управлять программой с помощью голосовых команд. Поисковые средства браузера изначально интегрированы с Google и рядом других популярных ресурсов.

Сейчас разработкой обозревателя занимается компания Opera Software, находящаяся в Норвегии. Заметим, что версия этого браузера Opera Mini на сегодняшний день играет доминирующую роль на рынке браузеров для карманных компьютеров, мобильных телефонов и смартфонов. Окно браузера Opera

версии 9 представлено на рис. 1.2. Получить последние версии браузера можно по адресу **http://www.opera.com/download/**.

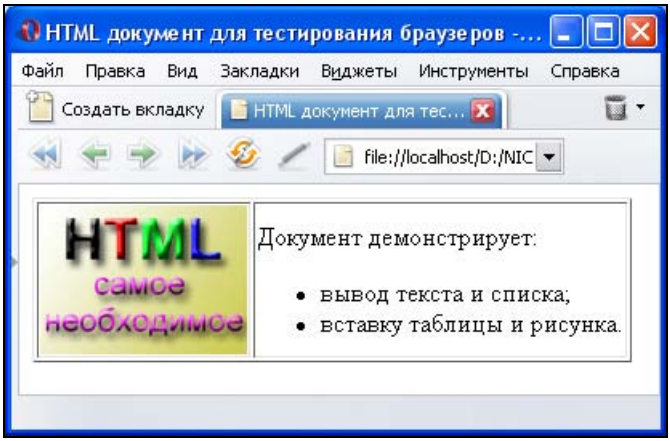

Рис. 1.2. Окно браузера Opera 9

Обозреватель Mozilla Firefox (буквально англ. "огненный лис", произносится "мозилла файрфокс"), ранее известный как Phoenix и Firebird, входит в тройку самых популярных в мире браузеров. Это первый программный продукт с открытым исходным кодом, получивший такую популярность. Mozilla и Mozilla Firefox были основаны на исходном коде Netscape, выпущенном под свободной лицензией MPL (Mozilla Public License).

Сторонники Firefox утверждают, что именно он обеспечивает наиболее высокий уровень безопасности пользователя, умеет ограждать его от навязчивой рекламы и обладает целым рядом новых возможностей. Браузер Firefox работает быстро, причем скорость работы заметна не только во время загрузки Web-страниц, но и при поиске в закладках или истории просмотра. Как и другие популярные обозреватели, Mozilla Firefox способен значительно расширить свои возможности за счет большого количества отдельно устанавливаемых надстроек.

Встроенный компонент "Менеджер загрузок" позволяет управлять процессами загрузки файлов из Сети, а также поддерживает историю загрузок.

Инструмент "Информация о странице" отображает подробную информацию о текущей Web-странице (тип страницы, ее кодировка, размер, информация о подлинности и безопасности соединения). Дополнительно устанавливаемый инструмент "Инспектор DOM" позволяет исследовать объектную модель документа на глубоком профессиональном уровне. Многие сочетания

клавиш совпадают для Internet Explorer и Firefox. Российский сайт поддержки браузера находится по адресу **mozilla-russia.org**, а окно его второй версии приведено на рис. 1.3. Mozilla Firefox поддерживает удобное обновление в режиме "онлайн", а загрузить этот браузер можно по URL-адресу **http://www.getfirefox.net**.

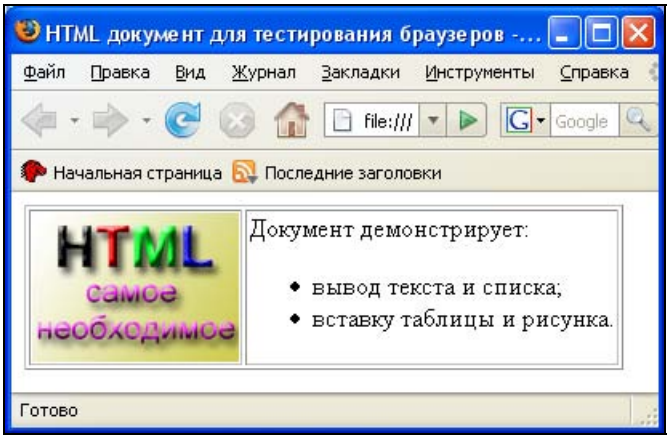

Рис. 1.3. Окно браузера Mozilla Firefox 2

Браузер Netscape Navigator, различные версии которого выходили также под названиями Netscape Communicator и Netscape Browser, еще несколько лет назад был лидером по числу пользователей.

Изначально эта серия браузеров разрабатывалась компанией Netscape Communications Corporation, позже она была приобретена компанией AOL. В настоящее время, Netscape производится на основе браузера Mozilla Firefox, начиная с версии 8, он работает только под управлением Microsoft Windows.

Традиционными для Netscape являются собственная технология работы с пользовательскими закладками, поддержка некоторых элементов HTML, игнорируемых другими разработчиками (таких, например, как мерцающий текст), определенная "строгость" в отношении правильности разметки содержимого страницы. Сегодня пользователей Netscape осталось немного, по крайней мере, в списки популярных программ его перестали включать.

Окно браузера Netscape Navigator версии 9 приведено на рис. 1.4, Web-сайт программы находится по адресу **http://browser.netscape.com**.

Название Safari носит браузер компании Apple, входящий в состав операционной системы Mac OS. Браузер включает в себя поддержку вкладок, встроенный менеджер закладок, подобный iTunes, встроенный блокировщик

всплывающих окон и поддержку новостей. В 2007 году компания Apple выпустила тестовую версию Safari 3.0, способную работать в операционных системах компании Microsoft. К сожалению, поддержка стандартов HTML браузерами Safari пока далека от совершенства. Окно браузера Safari третьей версии приведено на рис. 1.5.

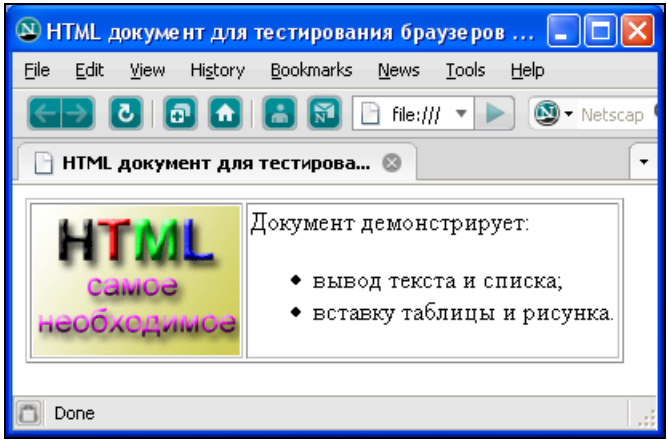

Рис. 1.4. Окно браузера Netscape Navigator 9

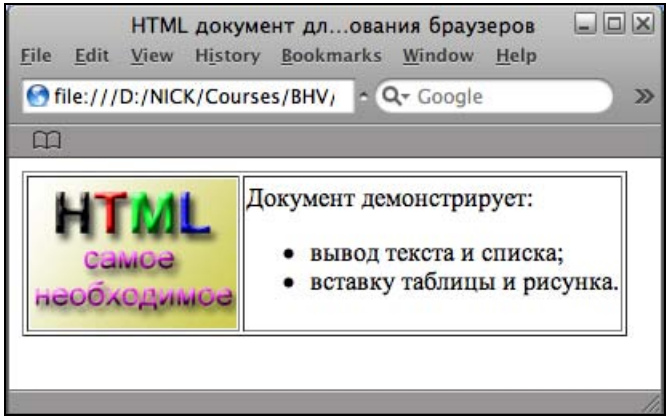

Рис. 1.5. Окно браузера Safari 3

Все это великолепное разнообразие браузеров преследует, в конечном итоге, одну цель — возможно быстрее и качественнее донести до пользователя содержимое Web-страницы. Поэтому выбор браузера остается за вами, а создать HTML-код, одинаково хорошо работающий и привлекательно выглядящий во всех браузерах, — вполне в ваших силах.

Управление любым современным браузером достаточно стандартизовано.

Для комфортной работы в обозревателе необходимы, как минимум, следующие инструменты:

- *адресная строка* (панель адреса, панель навигации, Toolbar) — содержит и позволяет ввести URL-адрес требуемой страницы или путь к локально расположенному документу, а также размещает стандартные кнопки перехода по страницам ("Вперед", "Назад", "Обновить", "Остановить", "Домой"). В некоторых браузерах стандартные кнопки вынесены на отдельную панель инструментов;
- *строка состояния* (строка статуса) — нижнее информационное поле окна браузера, содержащее важную дополнительную информацию. Так, в процессе загрузки Web-страницы здесь выводится информация о ее ходе, а при наведении курсора мыши на ссылку — показывается адрес URL, соответствующий ссылке;
- *панель вкладок* (иногда панель закладок, Tab bar) — дает возможность открывать в текущем окне дополнительные Web-страницы и переключаться между ними. Концепция вкладок позволяет, не отказываясь от возможности открыть ссылку в новом окне браузера, более удобно управлять наборами одновременно открытых Web-страниц.

Указанные панели инструментов, как правило, включены изначально, а управлять ими можно из меню **Вид** (**View**) браузера.

В заключение обзора отдельно остановимся на проблемах совместимости браузеров. Несмотря на то, что спецификации HTML существуют и продолжают развиваться, проблемы эти способны существенно испортить жизнь как начинающему, так и опытному разработчику. С чем связано такое положение дел?

Хотя рекомендованные консорциумом W3C стандарты HTML разработаны уже давно, различные браузеры могут отображать один и тот же документ по-разному. Вы могли убедиться в этом даже на примере простой тестовой странички, показанной в окнах браузеров на рис. 1.1–1.5. Ситуация объясняется как различным толкованием стандартов, так и конкуренцией фирм-производителей. Известная "война браузеров", которая в течение нескольких лет велась между программными продуктами Internet Explorer и Netscape Navigator, с одной стороны "обогатила" HTML множеством нестандартных и сомнительных нововведений, но с другой стороны, именно эта конкуренция способствовала распространению и утверждению современных стандартов языка разметки.

Нередко существенная часть материалов книг по HTML — это описания тонкостей применения тех или иных тегов в браузерах от различных производителей.

Автор придерживается убеждения, что не "страницы пишутся под браузер", а разработчики браузеров должны соблюдать стандарт языков разметки. Поэтому в моей книге совместимости уделено не слишком много места. Это означает лишь то, что мы будем работать с устоявшимся "ядром" формата HTML, поддержка которого одинакова во всех популярных сегодня браузерах. "Сомнительные" теги, если мы не сможем без них обойтись, конечно, будут описаны с указанием поддерживающих их программ.

По мере развития и все большего проникновения Сети в нашу жизнь, проблемы совместимости медленно, но верно решаются. Будем надеяться, что скоро они станут достоянием истории.

Замечу также, что эта книга — не популярное изложение рекомендованного стандарта HTML. Порой "отмененный" официально тег используется настолько широко, что вряд ли можно ожидать прекращения его поддержки производителями браузеров, тем более что новые идеи, предложенные разработчиками, часто находят отражение в официальной документации.

1.6. Надстройки и плагины

Надстройками или *плагинами* называют дополнительные программы и программные библиотеки, расширяющие возможности браузера.

С помощью плагинов можно встроить в браузер новую специализированную панель инструментов, "научить" его распознавать новый формат файлов или получить в свое распоряжение интегрированные с обозревателем мультимедийный плеер и органайзер.

Как правило, производители новых технологических решений для Web выпускают плагины к популярным браузерам, обеспечивающие пользователям Интернета предлагаемые возможности.

Крупные интернет-порталы, такие как Google (**www.google.com**) или российский Яndex (**www.yandex.ru**), также зачастую предлагают своим посетителям установить собственные плагины для расширенного поиска информации в Интернете, быстрого обмена сообщениями или адресами страниц, оценки рейтинга просматриваемых сайтов и т. п.

Браузер Internet Explorer последних версий позволяет управлять плагинами из окна **Надстройки** меню **Сервис**, Mozilla Firefox и Netscape используют для этой цели окно **Дополнения** (**Add-ons**), вызываемое из меню **Инструменты** (**Tools**), обозреватель Opera показывает список установленных дополнений по команде **Плагины** пункта **Дополнительно** из верхнего меню **Инструменты**, а Safari делает то же самое по команде **Installed plug-ins** из меню **Help**.

Как правило, окно управления плагинами позволяет подключить или отключить любые из установленных надстроек, а также перейти на сайт производителя браузера для загрузки новых дополнений.

Многие типы файлов, не имеющие прямого отношения к формату HTML, распознаются браузерами как раз потому, что в стандартной поставке уже присутствует ряд плагинов. Корректно составленный HTML-код позволяет загружать требуемый плагин автоматически, если связанный с Web-страницей тип объектов оказался "незнаком" вашему браузеру.

Именно применение плагинов позволяет превратить браузер в действительно многофункциональное приложение. Разумеется, в установке плагинов следует соблюдать осторожность, доверяя лишь тем из них, которые предлагают загрузку с официального сайта известного производителя.

Глава 2

HTML и программирование

Главное, чему вы научитесь, прочитав эту книгу — создавать Web-страницы с использованием формата HTML. Несмотря на то, что язык разметки гипертекста — основа всех остальных Web-технологий, а его изучение — необходимый для каждого грамотного пользователя этап развития, нельзя обойти стороной вопрос о многочисленных применениях средств HTML к автоматическому созданию и обработке Web-страниц.

В *главе 2* вашему вниманию предлагается обзор существующих технологий программного создания Web-страниц, их возможностей, преимуществ и недостатков. Ориентируясь в описанных здесь технологиях, вы сможете чувствовать себя уверенней среди разнообразия современного Интернета.

Ключевое для усвоения главы понятие — *динамическая Web-страница*. В отличие от *статической* страницы, разработанной "вручную" или в специализированном редакторе и хранящейся на сервере в виде отдельного файла с расширением html или htm, динамический документ программно генерируется сервером по запросу клиента и затем пересылается для отображения установленному на клиентской машине браузеру.

Все существующие технологии динамического создания HTML-кода можно разделить на *клиентские* и *серверные*. Разница между ними в том, что клиентская программа выполняется *на стороне клиента*, т. е. на компьютере пользователя, непосредственно браузером или специально установленной внешней программой, а серверная — удаленным Web-сервером, и браузер получает от него готовый html-документ. Часто клиентские и серверные программы обобщенно называют *скриптами*, хотя это слово больше подходит для серверных приложений.

Динамические страницы, работающие на стороне клиента, формируют свое содержимое под управлением программы-*сценария*, написанной на одном

из специальных *языков сценариев*. Код сценария может быть встроен непосредственно в документ HTML или храниться в отдельном файле.

Динамические страницы, формирующиеся на стороне сервера, обычно обрабатывают и выводят информацию из имеющейся на сервере базы данных, хотя могут использовать для хранения данных и отдельные файлы различных форматов.

2.1. Технологии и языки клиентского программирования

Языки сценариев в основном применяются для программирования вариантов поведения браузера при загрузке или просмотре документа. Выполняя операторы языка сценариев, браузер может динамически формировать Web-страницы в зависимости от внешних по отношению к HTML факторов, таких как сделанный пользователем выбор или введенное им с клавиатуры значение, версия обозревателя или операционной системы, текущие дата и время, сгенерированная последовательность случайных чисел и т. п. Второе популярное применение сценариев — добавление к Web-странице отсутствующих в HTML средств управления ее отдельными элементами. Наконец, языки сценариев реализуют заложенную в HTML модель *обработки событий*, таких как нажатие клавиши или перемещение курсора мыши. Это позволяет разрабатывать по-настоящему *интерактивные* Web-сайты, взаимодействующие с пользователем в реальном времени.

Важно понимать, что все сценарные языки являются *интерпретируемыми*. Это означает, что обозреватель считывает исходный текст сценария вместе с HTML-документом, а затем исполняет его без преобразования (*компиляции*) в машинный код. Таким образом, весьма часто ошибка сценария может быть обнаружена лишь при загрузке содержащего его документа.

Сценарии могут как внедряться непосредственно в HTML-код с помощью специальных тегов, так и храниться во внешних файлах, для подключения которых также требуется тег. Подробное изучение сценарных языков выходит далеко за рамки книги, но даже краткая их характеристика поможет нам лучше сориентироваться в возможностях HTML и облегчит знакомство со многими его элементами.

Язык сценариев JavaScript был разработан для браузеров Netscape Navigator, позднее к его развитию подключилась корпорация Microsoft. Версия Microsoft получила название JScipt, поскольку JavaScript является зарегистрированной маркой фирмы Netscape. JavaScript — единственный язык сцена-

риев, поддержка которого встроена во все популярные современные обозреватели, что делает его фактическим стандартом для программирования сценариев.

JavaScript — не "полноценный" язык программирования, он изначально ориентирован на использование создаваемой браузером среды, в которой исполняются сценарии.

JavaScript — объектно-ориентированный язык. И его языковые средства, и возможности среды доступны как *объекты*, а программа (сценарий) представляет собой набор взаимодействующих объектов. Объект JavaScript — это неупорядоченный набор *свойств*, каждое из которых имеет нуль или более значений, определяющих его использование. Например, свойство `bgColor` объекта `document` позволяет сценарию управлять фоновым цветом Web-страницы, а свойство `href` объекта `location` — получить ее URL-адрес. Изменяя значение свойств, сценарий непосредственно воздействует на соответствующие элементы страницы. Так, задание нового допустимого значения свойству `location.href` позволит сценарию загрузить в текущее окно новую страницу. Имя свойства или метода отделяется от имени объекта символом точки.

Объекты и свойства не обязательно простые — они могут содержать другие объекты, переменные различных типов данных и *методы* — так в JavaScript называют подпрограммы-функции.

JavaScript содержит множество встроенных объектов, предназначенных для доступа к различным элементам Web-страниц и для реализации стандартных средств работы со строками, датами, математическими функциями и т. п.

Синтаксис JavaScript во многом похож на Java, но упрощен в сравнении с ним, чтобы сделать язык более легким в изучении. Например, объявление переменной не содержит ее типа, свойства также не имеют типов, а объявление функции может располагаться в тексте сценария после ее вызова.

Браузер позволяет присоединить сценарии к таким событиям, как загрузка и выгрузка страниц или рисунков, нажатие клавиш и движение мыши, выбор текста и пересылка данных. При этом программный код сценариев лишь реагирует на события и потому не нуждается в главной программе. Набор объектов, предоставляемых браузерами, известен под названием Document Object Model (DOM). Способы подключения сценариев, написанных на JavaScript, к HTML-документам подробно описаны в *главе 15*.

Простейший скрипт, выводящий стандартное окно сообщения с классической надписью "`Hello, World!`", будет иметь на JavaScript следующий вид:

```
<script type="text/javascript">
 window.alert('Hello, World!');
</script>.
```

Язык сценариев VBScript или Visual Basic Scripting Edition используется при создании скриптов в операционных системах семейства Windows. Он был создан компанией Microsoft как замена устаревшему языку пакетных файлов, который поддерживался в операционной системе DOS. VBScript — это несколько упрощенная версия языка Visual Basic, в частности, VBScript не поддерживает типы данных.

Аналогичная приведенной ранее программа на VBScript может иметь следующий вид:

```
<script type="text/vbscript">
 Dim strMessage
 strMessage = "Hello, world!"
 MsgBox strMessage
</script>.
```

Кроме написания клиентских скриптов для браузера Internet Explorer, VBScript служит для решения ряда задач администрирования Windows.

Серверный язык Perl, описанный в *разд. 2.2*, имеет и клиентскую версию — PerlScript. Для его использования в Web-страницах необходимо установить PerlScript на каждом компьютере-клиенте, поэтому применение языка ограничено, в основном, локальными сетями. Последняя версия PerlScript бесплатна и доступна по адресу **www.activestate.com/ActivePerl**. Аналогичное ограничение имеет язык TCL (Tool Command Language), доступный по адресу **dev.scriptics.com**.

Упомянутый ранее язык JavaScript позволяет гибко управлять работой браузера и содержимым страниц, но в нем не реализованы возможности создания графических объектов, организации сетевых соединений и параллельного выполнения нескольких вычислительных потоков. В определенной мере компенсировать эти недостатки позволяет язык Java. Несмотря на сходство названий, между Java и JavaScript немного общего — разве что синтаксическая "похожесть" на C++.

Java — мощный язык программирования, разработанный в 1995 г. компанией Sun Microsystems. Изначально язык Java был предназначен для интерактивного телевидения и управления сложными бытовыми устройствами, однако быстрое развитие Интернета выявило его новое предназначение — создание программ, называемых *апплетами* (applets), которые могут быть загружены Web-браузером с сервера и выполнены на стороне клиента. Все современные браузеры поддерживают Java, и его высокая популярность в немалой степени связана с уникальной особенностью, отличающей Java от всех остальных средств программирования.

Мы уже упоминали о различии между *компилируемыми* и *интерпретируемыми* языками программирования. Программа на компилируемом языке, таком как C++ или Pascal, перед использованием должна быть преобразована (откомпилирована) из исходного текстового файла в исполняемый модуль, содержащий машинные коды. Такой подход обеспечивает высокое быстродействие и эффективность, но он жестко ограничивает сферу применения программы конкретным классом компьютеров или конкретной операционной системой, а для переноса программы в другую среду требуется, как минимум, ее повторная компиляция в соответствии с требованиями новой платформы. Более того, наличие в программе специфических для какой-либо операционной системы действий может потребовать внесения в нее значительных изменений.

Совершенно очевидно, что такая методика не годится для среды Web, объединяющей самые разные компьютеры и вычислительные платформы.

Поэтому JavaScript, Perl, PHP и вообще все сценарные языки являются интерпретируемыми, т. е. не требуют предварительной компиляции в машинный код. Сценарий на интерпретируемом языке исполняется установленным в системе *интерпретатором*, читающим и сразу же выполняющим исходный текст программы. Преимущество сценарных языков — их гибкость и быстрота разработки приложений, а существенные недостатки — низкая скорость работы и полная зависимость от интерпретатора. Кроме того, усложнение сценарных языков неизбежно требует сложной и громоздкой исполняющей среды.

Разработчики Java нашли узкий путь между Сциллой и Харибдой. Исходный текст программы, хранящийся в одном или нескольких текстовых файлах с расширением java, компилируется в промежуточный стандартный код, называемый *байт-кодом*. Файлы с байт-кодом имеют расширение class. Байт-код не является машинным языком какого-либо процессора, но он был специально создан авторами Java в качестве *машинного кода виртуального процессора*, не имеющего физической реализации. Для того чтобы байт-код мог исполняться на конкретном компьютере, на него нужно установить *виртуальную Java-машину*, представляющую собой программу, интерпретирующую байт-коды как готовые исполняемые модули. Поскольку байт-код Java — это язык машинного уровня, добиться совместимости виртуальных машин для разных платформ не слишком сложно.

В результате платформенно-независимые модули на Java способны выполняться быстрее, чем аналогичные по сложности программы на интерпретируемых языках. Разумеется, по сравнению с откомпилированными приложе-

ниями Java остается "медленным" языком, но это — обязательная плата за независимость от архитектуры ЭВМ.

Концепция виртуальной машины обеспечивает ряд важных особенностей, характерных для применений Java в Интернете:

- программа на Java, работающая в среде, созданной виртуальной машиной, имеет доступ только к ее ресурсам, но не к физической среде компьютера. Это позволяет четко определить полномочия программы по доступу к файлам и функциям операционной системы, обеспечивая достаточную безопасность апплетов;
- программа, сохраненная в виде байт-кодов и переданная сервером клиенту, может быть немедленно запущена на выполнение установленной в системе виртуальной Java-машиной. Именно так работают апплеты, доступные со многих Web-страниц;
- виртуальная Java-машина умеет "собирать" программы прямо во время исполнения, выполняя *динамическое связывание модулей*. Поскольку существует множество стандартных модулей языка, динамическое связывание позволяет передавать по сети только написанные для конкретной задачи модули, что резко сокращает объем передаваемых данных.

Как уже говорилось, синтаксически Java напоминает C++, но отличается от него некоторыми важными деталями. Из Java убраны все потенциально опасные возможности, такие как прямой доступ к памяти, переопределение операторов и множественное наследование. При этом Java — это строго типизированный язык, поддерживающий динамическое управление памятью, включая автоматическую "сборку мусора", т. е. удаление ненужных объектов, и многопоточное выполнение программ.

Основное понятие языка Java — *класс*, состоящий из набора данных и методов (функций) для их обработки. В процессе работы Java-программы создаются *объекты*, т. е. переменные, являющиеся *экземплярами* данного класса. Новые классы описываются как наследники существующих, образуя многоступенчатую иерархию. Взаимосвязанные наборы классов объединяются в *пакеты*, которые создаются как самостоятельно компилируемые программные единицы.

Базовая на сегодня спецификация — Java 2, а "домашняя страница" языка находится в Интернете по адресу **java.sun.com**.

О работе с Java-апплетами и подключении их к своим Web-страницам подробнее будет рассказано в *главе 15*.

2.2. Технологии и языки серверного программирования

Для связи внешней программы с Web-сервером предназначен стандартный интерфейс CGI (от англ. Common Gateway Interface — общий шлюзовый интерфейс). Программу, которая работает совместно с Web-сервером через интерфейс CGI, называют *скриптом*, *шлюзом* или *CGI-программой*. Интерфейс CGI разработан таким образом, что его может использовать любой язык программирования, способный работать со стандартными устройствами ввода/вывода.

Скрипты обычно помещаются в папку с именем cgi-bin внутри корневой папки сервера, но это требование не является обязательным. Скрипт может располагаться в любой другой папке, хотя при этом большинство Web-серверов требуют специальной настройки. В Web-сервере Apache такая настройка выполняется либо при помощи общего файла настроек httpd.conf, либо с помощью файла .htaccess в том каталоге, где размещен скрипт. Большинство серверов позволяют запускать все скрипты, имеющие расширение cgi, при этом скрипт чаще всего написан на одном из рассмотренных далее языков программирования.

Классическим языком Web-программирования считается Perl (от англ. Practical Extraction and Report Language — практический язык извлечений и отчетов), изначально разработанный для сложной обработки текстов в операционной системе Unix. Сегодня Perl поддерживается тысячами разработчиков как проект с исходным открытым кодом и реализован практически для всех существующих операционных систем. Синтаксис Perl напоминает синтаксис языка Си, но, благодаря обилию хорошо продуманных сокращений, разработка приложений на нем существенно быстрее. Сочетание скорости разработки с мощью получаемых программ делает Perl непревзойденным инструментом реализации сложных проектов. Для Perl существует громадная библиотека готовых модулей CPAN (Comprehensive Perl Archive Network), доступ к которой свободен для всех (**www.cpan.org**).

Файл, содержащий программу на Perl, обычно имеет расширение pl и следующее строение:

```
#!/usr/bin/perl
print "\nHello, world!";
```

Здесь первая строка указывает путь к интерпретатору Perl в операционной системе Unix. В Windows эта строка просто игнорируется, поскольку тип файла определяется его расширением. В остальном программы на Perl со-

вершенно одинаковы для обеих операционных систем, если разработчик не использует зависящих от операционной системы функций и модулей.

Преимуществами Perl считаются высокое по сравнению с другими скриптовыми языками быстродействие и мощные встроенные средства для обработки текстовых данных, однако последние версии альтернативных языков практически ничем ему не уступают. Официальный сайт Perl имеет адрес **www.perl.org**.

Основной и успешный конкурент Perl — язык PHP (от англ Hypertext Preprocessor — препроцессор гипертекста), специально созданный для генерации HTML-страниц на Web-сервере и работы с базами данных. В настоящее время PHP поддерживается подавляющим большинством серверов, именно на этом языке написана значительная часть популярных Web-порталов. PHP в рекордные сроки стал популярным благодаря своей простоте, скорости выполнения, удобной встроенной интеграции с базами данных MySQL, богатой функциональности и свободному распространению исходных кодов. PHP отличается наличием неизменного ядра и подключаемых модулей-расширений для работы с различными базами данных, динамической графикой, криптографией, документами формата PDF и т. д. Существуют сотни дополнительных модулей, но в стандартную поставку входит лишь несколько десятков из них. Интерпретатор PHP подключается к Web-серверу либо через библиотеку, созданную специально для этого сервера, либо в качестве CGI-приложения.

Синтаксис PHP также во многом подобен синтаксису Си, ряд элементов заимствован и из Perl. Простейшая программа на PHP выглядит следующим образом:

```
<?php
 echo 'Hello, world!';
?>
```

Интерпретатор PHP исполняет код, находящийся внутри ограничителей `<?php` и `?>`. Все, что находится вне ограничителей, выводится без изменений. В основном, это используется для вставки PHP-кода непосредственно в HTML-документ. Официальный сайт PHP находится по адресу **www.php.net**, а крупнейшими русскоязычными ресурсами являются **php.ru**, **phpclub.ru** и **phpconf.ru**. Чаще всего Web-страница, написанная на PHP, легко опознается по расширению php или php3.

На практике пользователь и разработчик чаще имеют дело не с отдельным PHP-скриптом, а с целым "*движком сайта*", представляющим собой набор взаимосвязанных скриптов. Существуют десятки распространенных движков, по понятным причинам, большинство из них — бесплатные проекты с открытым исходным кодом.

Python (читается "питон") — созданный уже в 90-х гг. прошлого века интерпретируемый язык программирования высокого уровня с динамической типизацией, автоматическим управлением памятью и удобными высокоуровневыми структурами данных. Питон обладает простым и выразительным синтаксисом, поддерживая при этом сразу несколько парадигм программирования (структурное, объектно-ориентированное, функциональное и аспектно-ориентированное).

Интерпретатор Питона разработан для большинства распространенных платформ и доступен практически без ограничений в любых приложениях.

К недостаткам Python критики относят невысокое быстродействие, отсутствие статических типов данных и невозможность модификации встроенных в язык классов.

Узнать больше об этом перспективном языке можно на официальном сайте **www.python.org** или одном из популярных русскоязычных ресурсов **www.python.ru**, **www.python.com.ua**.

ASP (от англ. Active Server Pages — активные серверные страницы) — это технология от компании Microsoft, позволяющая разрабатывать приложения для World Wide Web. ASP изначально была предназначена для работы в операционных системах семейства Windows на Web-серверах IIS, хотя позднее появились независимые версии для Apache. ASP не является языком программирования, представляя собой лишь технологию предварительной обработки данных, позволяющую подключать программные модули на языках сценариев VBScript или JScript в процессе формирования Web-страницы.

Страница на ASP представляет собой обычный документ HTML со вставками, обозначенными ограничителями `<%` и `%>`:

```
<%
 Response.write "Hello World!"
%>
```

Здесь внутри ограничителей находится текст программы на VBScript, интерпретируемый при запросе страницы. Для технологии ASP язык VBScript принят по умолчанию, хотя возможно использование и JScript.

ASP получила свое развитие в виде ASP.NET — новой технологии создания Web-приложений, основанной на платформе Microsoft.NET.

Программисты могут писать код для ASP.NET, применяя различные средства, поддерживаемые в системе программирования .NET Framework. Обычно это языки Visual Basic.NET, JScript.NET или C#, а также "открытые" языки, такие как Perl и Python. По утверждению компании Microsoft, ASP.NET имеет преимущество в скорости по сравнению с другими технологиями, осно-

ванными на скриптах, потому что код на стороне Web-сервера обычно компилируется в одну или несколько динамических библиотек. Страницы, выполненные по технологии ASP, обычно содержат в URL расширение файла asp или aspx, а официальный сайт ASP.NET от Microsoft находится по адресу **www.asp.net**.

Еще одна технология, позволяющая Web-разработчикам динамически генерировать страницы, называется JSP (JavaServer Pages). JSP входит составной частью в единую технологию создания бизнес-приложений J2EE (Java 2 Enterprise Edition), разработанную корпорацией Sun. Технология JSP позволяет внедрять Java-код в статичное содержимое страницы, а также может работать с библиотеками JSP-тегов для внедрения их в документы. Страницы компилируются в *сервлеты*, представляющие собой Java-классы, выполняемые на сервере.

Весь код страницы транслируется в Java-код сервлета с помощью компилятора JSP-страниц Jasper, и затем компилируется в байт-код виртуальной машины Java. Написанные на Java сервлет-контейнеры могут работать как самостоятельные Web-серверы, быть поставщиками Web-страниц для других серверов или интегрироваться в J2EE-сервер приложений. Домашняя страница JSP расположена по адресу **java.sun.com/products/jsp**.

Технология SSI (Server Side Includes — включения на стороне сервера) представляет собой несложный язык для динамической "сборки" Web-страниц на сервере из отдельных составных частей и выдачи клиенту полученного HTML-документа. Изначально технология была реализована в Web-сервере Apache.

Синтаксис SSI позволяет включать в документ другие SSI-страницы, вызывать внешние CGI-скрипты, реализовывать условные операции "если-то-иначе", работать с переменными и т. п. Благодаря крайней простоте языка, сборка SSI-страниц происходит очень быстро, однако многие возможности полноценных языков программирования, например, работа с файлами, в SSI отсутствуют. Отдельные элементы синтаксиса SSI присутствуют и в других скриптовых языках, например, в ASP.

В качестве примера приведем директиву SSI, включающую в документ результаты исполнения скрипта с URL-адресом **http://www.scripts.com/test.cgi**:

```
<!--#exec cgi=" http://www.scripts.com/test.cgi "-->.
```

Новейший и перспективный подход к построению интерактивных пользовательских интерфейсов Web-приложений — AJAX (от англ. Asynchronous JavaScript and XML — асинхронный JavaScript и XML, по-английски произносится "эй-джэкс", по-русски распространено произношение "аякс"). Суть

данной технологии состоит в том, что при использовании AJAX Web-страница не перезагружается полностью в ответ на каждое действие пользователя. Вместо этого с Web-сервера *догружаются* только запрошенные пользователем данные.

AJAX базируется на двух основных принципах:

- средства DHTML для динамического изменения содержания страницы (см. *главу 3*);
- технология динамического обращения к серверу "на лету", без перезагрузки всей страницы полностью.

Эти два принципа позволяют создавать удобные Web-интерфейсы на тех страницах, где необходимо активное взаимодействие с пользователем. Технология AJAX стала особенно популярной после того, как компания Google начала активно использовать ее при создании своих сайтов, таких как Gmail, Google Maps и Google Suggest.

Следует отметить, что AJAX поддерживается только самыми современными браузерами, поэтому ее применение может привести к проблемам совместимости.

Основной конкурент AJAX — стек технологий RIA (Rich Internet Applications), разрабатываемый и продвигаемый компанией Adobe и ее дочерним подразделением Macromedia.

Популярный русскоязычный ресурс, посвященный развитию указанных технологий, находится по адресу **www.ajaxplanet.ru**.

Глава 3

HTML и другие форматы

HTML — основной, но не единственный язык разметки документов. Существуют как более общие, так и узко специализированные решения. Объем и задачи книги не позволяют изучить их все, однако было бы обидно чувствовать себя невеждой, когда столь многие из окружающих обсуждают преимущества XHTML и недостатки WML.

Эта глава преследует простую цель — разобраться в большом и не всегда "дружном" семействе языков разметки.

Исторически первым распространенным форматом был SGML (Standard Generalized Markup Language — стандартный общий язык разметки, произносится "эс-джи-эм-эль"). SGML — наследник разработанного в 1960 г. компанией IBM языка GML (Generalized Markup Language) — это *метаязык*, т. е. на нем можно определять правила построения других языков форматирования документов.

SGML был разработан для совместной разработки машинных документов в больших правительственных и аэрокосмических проектах. Он широко применялся в печатном деле и издательской сфере, но его сложность затруднила повседневное использование. Основные наследники SGML — форматы HTML и XML.

HTML (Hypertext Markup Language — язык разметки гипертекста) — самое распространенное сегодня средство для создания Web-страниц. Технология HTML позволяет связывать между собой документы различных форматов с помощью гипертекстовых ссылок (гиперссылок или линков). Такие связи между документами, расположенными на серверах по всему миру, позволяют системе работать так, будто она представляет собой единую "всемирную паутину" информации (World Wide Web). Документ HTML — это файл, содержащий обычный текст и специальные команды — *теги* (tags). Теги определяют визуальное форматирование текста (цвет и начертание шрифта, разметку

заголовков, таблиц и т. п.), а также связи данного HTML-документа с другими ресурсами (изображениями, таблицами стилей, видеороликами, другими HTML-документами и т. д.). В SGML, HTML и XML теги оформляются открывающей ("<") и закрывающей (">") угловыми скобками, за которыми следует *наименование* тега, и затем — уточняющие его действие команды-*атрибуты*.

HTML был разработан британским ученым Тимом Бернерсом-Ли в 1991–1992 годах в стенах Европейского совета по ядерным исследованиям в Женеве (Швейцария). Поначалу HTML создавался как язык для обмена научной и технической документацией, пригодный для людей, не являющихся специалистами в области верстки.

Затем помимо упрощения структуры документа, в HTML была внесена поддержка различных видов гипертекстовых ссылок, а позднее в язык были добавлены мультимедийные возможности. Изначально HTML предназначался для структурирования и форматирования документов без их привязки к программным средствам отображения. В идеале, текст с разметкой HTML должен был без стилистических и структурных искажений воспроизводиться на оборудовании с различной технической оснащенностью (цветной экран современного компьютера, ограниченный по возможностям экран мобильного телефона или программа голосового воспроизведения текстов). Однако современное применение HTML очень далеко от изначального замысла. С течением времени, основная идея платформенной независимости HTML была отдана "в жертву" современным потребностям в мультимедийном и графическом оформлении.

HTML является приложением SGML и соответствует международному стандарту ISO 8879. Текущий стандарт HTML 4.01 существует с 1999 г. В настоящее время опубликован проект пятого стандарта языка. Новая версия HTML обещает дополнить язык многочисленными расширениями и обеспечить более простую, логичную и удобную систему правил. Разумеется, новый стандарт при этом не будет противоречить старому, полностью сохраняя обратную совместимость с уже написанными миллионами документов.

Стандартный HTML не дает возможности создавать гибкие программируемые документы и не поддерживает их стилевого оформления. Преодолеть эти недостатки позволяют две технологии, работающие в тесной связи с языком разметки.

CSS (Cascading Style Sheets, каскадные таблицы стилей) — это еще один компьютерный язык, применяемый для оформления Web-страниц. Он несколько сложнее, чем HTML, однако позволяет достигать более интересных результатов. Прежде всего, это касается *стилей*, т. е. наборов правил оформ-

ления и форматирования, которые могут быть применены к различным элементам страницы. В стандартном HTML для присвоения какому-либо элементу определенных свойств (цвета и размера шрифта, положения на странице и т. п.) приходится каждый раз описывать эти свойства с помощью тегов, даже если на одной страничке находится множество элементов, не отличающихся один от другого. В технологии CSS реализован другой, более удобный и экономичный способ. Для присвоения элементу определенных характеристик вы должны один раз описать этот элемент и определить его описание как стиль, а в дальнейшем просто указывать имя созданного стиля. Все современные браузеры поддерживают технологию CSS, в настоящее время существует ее второй стандарт. В *главе 14* мы более подробно познакомимся с основными возможностями каскадных стилей.

Dynamic HTML или DHTML — это способ создания интерактивного Web-сайта, сочетающий теги языка HTML, код JavaScript, выполняемый на стороне клиента, и каскадные таблицы стилей CSS. DHTML возник как набор методов динамического создания и изменения Web-страниц путем вызова из HTML- документа сценариев. Однако развитие этих методов привело к полному пересмотру концепции Web-документа и формированию понятия объектной модели документа DOM (Document Object Model).

DOM представляет собой платформенно-независимый программный интерфейс, позволяющий программам и скриптам управлять содержимым документов HTML и XML, а также изменять их структуру и оформление.

Модель DOM не накладывает ограничений на структуру документа. Любой документ известной структуры с помощью DOM может быть представлен в виде дерева узлов, каждый из которых содержит некоторый объект. Узлы связаны между собой отношениями "родитель-потомок".

Изначально многие браузеры имели собственную модель DOM, не совместимую с остальными. Для того чтобы обеспечить совместимость, специалисты международного консорциума W3C классифицировали эту модель по уровням, для каждого из которых была создана своя спецификация. Все эти спецификации объединены в общую группу, носящую название W3C DOM.

DHTML может быть использован для создания программы, работающей в браузере. С помощью технологии DHTML можно реализовать более простые и удобные средства навигации, работающую в браузере видеоигру или динамическое перетаскивание элементов по экрану. Подробнее технология DHTML будет описана в *разд. 14.7*.

XML (eXtensible Markup Language — расширяемый язык разметки; произносится "экс-эм-эл") — формат, представляющий собой свод общих синтаксических правил. XML предназначен для хранения структурированных данных

(вместо существующих файлов баз данных), обмена информацией между программами, а также для создания на его основе более специализированных языков разметки, иногда называемых *словарями*. XML — это упрощенное подмножество языка SGML.

XML создавался для обеспечения совместимости при передаче структурированных данных между системами обработки информации, особенно при передаче данных через Интернет. Словари, основанные на XML (например, RDF, RSS, MathML, XHTML и SVG), сами по себе формально описаны, что позволяет программно изменять и проверять документы, не зная их семантики, т. е. смыслового значения элементов.

Важное отличие XML от HTML — возможность описывать собственные теги и команды, создавая предметно ориентированные расширения формата. В отличие от HTML, XML не содержит никаких указаний на то, как описанные в XML-документе данные должны отображаться в браузере. Способ отображения данных для различных устройств задается языком описания стилей XSL, который играет для XML примерно ту же роль, что CSS для HTML.

Сторонники XML относят к его основным достоинствам строгую определенность синтаксиса, позволяющую языку оставаться эффективным и непротиворечивым, поддержку некоторых структур данных, таких как записи, списки и деревья, возможность описать средствами XML практически любые типы документов.

Тем не менее, XML не вытеснил HTML. Более того, можно с уверенностью прогнозировать, что в обозримом будущем этого не случится. Причинами являются как очевидные недостатки XML (большие размеры документов, избыточный синтаксис и ограниченность заложенной в формат иерархической модели данных), так и практически важный факт, говорящий в пользу HTML — для большинства задач не нужна вся мощь синтаксиса XML, достаточно простых и производительных решений HTML.

Один из разработанных средствами XML словарей — язык разметки гипертекста XHTML (eXtensible Hypertext Markup Language). Он разрабатывался с целью максимально приблизить возможности XML по разметке Web-страниц к текущим стандартам HTML. Часто этот формат рассматривают как переходный от HTML к XML. Как и HTML, XHTML соответствует спецификации SGML. Вариант XHTML 1.1 разработан в 2001 г. Возможности HTML и XHTML сопоставимы, но считается, что анализ XHTML проще и быстрее. Поскольку синтаксис XML строже, чем SGML, обработка XHTML возможна даже на мобильных телефонах и карманных компьютерах с малыми вычислительными ресурсами. Распространению XHTML препятствует его непривычная для Web-мастеров строгость в отношении правил языка и ошибок

в коде. Если для HTML браузеры должны сами "попытаться понять", что хотел сказать автор, то, встретив ошибку в XHTML, — сообщить о ней и не обрабатывать документ.

Специально для отображения информации на маленьком экране мобильного телефона создавался язык гипертекстовой разметки WML (Wireless Markup Language). Этот язык работает в соответствии с протоколом WAP, используемым для связи мобильного телефона с написанными на языке WML *WAP-страницами*. Разработчикам WAP-ресурсов приходится ориентироваться на то, что каждая страница WML — это короткая текстовая информация и небольшие по размерам изображения. В настоящее время, в связи с быстрым увеличением вычислительной мощности мобильных устройств, формат WML вытесняется обычными средствами HTML и XML.

Конкурент спецификации WML — CHTML (Compact HTML). Этот формат также ориентирован на устройства с ограниченными возможностями отображения информации, такие как смартфоны, коммуникаторы, мобильные телефоны и КПК. CHTML поддерживает лишь часть тегов HTML. На данный момент язык используется в приложениях i-mode. Ожидается, что, по мере роста числа браузеров, поддерживающих новые стандарты, CHTML постепенно будет вытеснен XHTML Basic.

VRML (Virtual Reality Modeling Language, язык моделирования виртуальной реальности) — стандартный формат файлов для демонстрации трехмерной интерактивной векторной графики.

Как и другие языки разметки, VRML — это текстовый формат файлов, описывающий с помощью тегов и атрибутов различные трехмерные объекты, например, вершины и грани многогранников вместе с цветом поверхности, текстурами, блеском, прозрачностью и т. д. Возможна связь URL-адресов объектов с графическими компонентами таким образом, что браузер может получать по ссылке новую Web-страницу или VRML-файл. Движение, звуки, освещение и другие аспекты виртуального мира могут быть реакцией на действия пользователя или же на другие внешние события, такие как таймеры.

VRML-файлы обычно называются мирами и имеют тип wrl. Хотя VRML-миры имеют текстовый формат, их часто сжимают по алгоритму компрессии gzip, чтобы передавать по сети за меньшее время. Большинство программ трехмерного моделирования могут сохранять объекты и сцены в формате VRML.

Хотя VRML продолжает использоваться в некоторых областях, особенно в образовательной и исследовательской среде, он практически вытеснен форматом X3D, решающим сходные задачи. VRML пока остается файловым форматом для обмена трехмерными моделями, особенно в системах автоматизированного проектирования.

Основные проблемы сегодняшнего этапа развития Интернета связаны с огромным объемом накопленной информации и ее плохой структурированностью. Самый естественный путь решения проблемы — заставить компьютер или поисковую систему "понимать" данные, а не просто механически искать их по совпадению ключевых слов. Для этого многие компании и независимые разработчики пытаются дополнить обычные средства разметки их семантическими аналогами, что дало бы возможность поиска информации по смыслу.

Ведущая современная концепция "улучшения" Интернета и языков разметки — "семантическая паутина" (Semantic Web), главная цель которой — создание технологий машинной обработки информации, доступной посредством Всемирной сети. Основной упор концепция делает на работу с *метаданными* ("данными верхнего уровня", "данными о данных"), однозначно характеризующими свойства и содержание ресурсов Интернета, вместо распространенного сейчас текстового анализа документов.

В семантической паутине предполагается повсеместное использование, во-первых, универсальных адресов URL, во-вторых — специализированных языков-*онтологий* и *языков описания метаданных*.

Для внедрения концепции семантического Web предполагается создание сети документов, содержащих метаданные о ресурсах Всемирной паутины и существующей параллельно с ними. Метаданные позволяют поисковым машинам и другим интеллектуальным системам проводить однозначные логические заключения о свойствах этих ресурсов. Все описанные далее форматы основаны на расширении некоторых возможностей языка XML.

Консорциум W3C разработал модель для соответствующего описания ресурсов, и в особенности — метаданных о ресурсах. В основе этой модели, называемой RDF (Resource Description Framework) лежит идея об использовании специальных утверждений, высказываемых о ресурсе. Каждое утверждение имеет вид "субъект-предикат-объект" и называется в терминологии RDF *триплетом*. Например, утверждение "карандаш синего цвета" в RDF-терминологии можно представить следующим образом: субъект — "карандаш", предикат — "иметь цвет", объект — "синий". Идентификация субъектов, предикатов и объектов в RDF осуществляется с помощью адресов URL. Одна из главных целей RDF — одинаковое предоставление утверждений в машинно-распознаваемом и понимаемом человеком виде. Существует несколько синтаксисов для представления RDF-информации, основные из них — RDF/XML, триплеты и графовая модель.

Стандарт RDF Schema описывает набор отношений, необходимых для определения новых типов RDF-данных. Языком поддерживается также отношение наследования типов данных.

Язык OWL (Web Ontology Language, язык Web-онтологий) расширяет возможности "семантической паутины" по описанию новых типов данных (например, добавлением перечислений), а также позволяет определять новые типы данных в терминах уже существующих (например, определять тип, являющийся пересечением или объединением двух существующих). Язык онтологий призван обеспечить средство для описания классов и отношений между ними, которые присущи для Web-документов и программ.

В основе OWL лежит модель данных "объект-свойство". По мнению создателей, OWL пригоден для описания не только Web-страниц, но и любых объектов действительности. Каждому элементу описания в этом языке ставится в соответствие URL-адрес.

Семейство XML-форматов RSS (Really Simple Syndication, очень простое приобретение информации) предназначено для описания лент новостей, анонсов статей, изменений в интернет-форумах и т. п. Информация из различных источников, представленная в формате RSS, может быть собрана, обработана и предоставлена пользователю в удобном для него виде специальными программами-*агрегаторами*.

Обычно с помощью RSS дается краткое описание новой информации, появившейся на сайте, и ссылка на ее полную версию. Интернет-ресурс в формате RSS называется *RSS-каналом* или *RSS-лентой*.

Многие современные браузеры и почтовые клиенты умеют работать с RSS, кроме того, существуют специализированные приложения, собирающие и обрабатывающие информацию RSS-каналов. Популярностью пользуются также Web-агрегаторы, представляющие собой сайты по сбору и отображению информации с различных RSS-лент. RSS-потоки, содержащие заголовки материалов и ссылки на них, позволяют одновременно публиковать материал сразу на нескольких сайтах.

Основа проекта по созданию машинно-читаемых домашних страниц и социальных сетей FOAF (Friend of a Friend) — спецификация, которая определяет ряд выражений, используемых в высказываниях о ком-либо, например, имя, пол и другие характеристики. Чтобы сослаться на эти данные, указывается идентификатор, включающий уникальные свойства "друга" (адрес E-Mail, URL домашней страницы).

Основанный на RDF, определенный с помощью OWL и легко расширяемый проект FOAF позволяет распределять данные между различными компьютерными группами.

Практическая реализуемость "семантической паутины" ставится многими специалистами под сомнение. Во-первых, разработчики склонны избегать

"лишней" работы по поддержке документов с метаданными, во-вторых, открытыми остаются проблемы истинности метаданных и возможности построения онтологий высокого уровня, описывающих совокупности абстрактных понятий.

Каждый документ "семантической паутины", по сути дела, должен быть создан в двух экземплярах: предназначенном для чтения людьми и машинно-ориентированном. Этот недостаток семантического Интернета был главным толчком к созданию *микроформатов* для языков разметки.

Микроформаты (иногда сокращается до µF) — новая часть языков разметки, которая позволяет указывать семантику (значения) элементов Web-страниц на HTML или XHTML. Программы могут извлекать данные из страниц, которые помечены одним или несколькими микроформатами.

Фактически, проблема состоит в том, что имеющиеся в языках разметки семантические средства непосредственно применимы только к элементам технических документов. Для описания любых других классифицируемых данных (ресторанных рейтингов, поэзии, прайс-листов) специальная семантика отсутствует, соответственно, невозможна их машинная обработка.

Вместо разработки дополнительных форматов или XML-шаблонов для предметных областей, микроформаты предлагают стандартизацию имеющихся в HTML атрибутов `rel` и `class` с целью внесения в документы машинно-читаемой структуры непосредственно авторами текста. При этом интерпретация документов современными браузерами не меняется.

Атрибут `rel`, имеющийся у тегов ссылок (`<a>`, `<link>`), служит для указания ее типа (постоянная, альтернативная, не требующая индексации роботом и т. п.). Атрибут `class` позволяет произвольно группировать элементы HTML, относя их содержимое к тем или иным категориям. Стандартизация такой группировки открывает перспективы машинной обработки любых категоризируемых данных.

В настоящее время крупнейшие информационно-поисковые системы, такие как Google, Yahoo и MSN Search, активно развивают и начинают внедрять технологию разметки микроформатами. Крупнейший ресурс, посвященный развитию данной технологии, — сайт **microformats.org**.

К расширениям HTML можно отнести и различные системы *псевдотегов*, которые применяются всюду, где нужен ограниченный и более "узкий", чем HTML, набор команд разметки. Например, на Web-форумах для оформления публикуемых сообщений популярен набор псевдотегов PHPbb, которые ограничиваются квадратными скобками.

Наконец, многие аналитики связывают дальнейшее развитие Интернета и форматов разметки с принципами, обобщенно названными Web 2.0 в противовес "старому" Web 1.0.

Появление этого термина вызвано созданием множества сайтов, объединенных общими принципами построения и общей тенденцией развития собственного интернет-сообщества.

Перечислим основные черты Web 2.0:

- *Web-службы* — специализированные программы, выполняющие обмен данными в формате XML, доступ к которым осуществляется через Web;
- интерфейс Web-приложений AJAX — асинхронная загрузка нужных пользователю блоков данных без полного обновления интернет-страницы;
- *Web-синдикация* — одновременная публикация одного и того же материала на различных сайтах, выполняемая с помощью технологий RSS;
- *mash-up* ("смешение") — сервис, который полностью или частично использует в качестве источников информации другие сервисы, предоставляя пользователю новую функциональность для работы. В результате mash-up сервис сам может становиться новым источником информации. Таким образом, образуется сеть зависимых друг от друга сервисов, интегрированных друг с другом;
- *метки* — ключевые слова, описывающие рассматриваемый объект либо относящие его к некоторой категории. Метки присваиваются объекту, чтобы определить его место среди других объектов;
- *социализация Web* — использование разработок, позволяющих создавать сетевые сообщества, персонализация настроек Web-сайтов, рейтингование как ресурсов, так и сетевых "репутаций" членов различных сообществ.

Появление и быстрое распространение комментируемых сетевых дневников (*блогов*) также вписывается в концепцию Web 2.0, создавая так называемую "редактируемую паутину" (writeable Web).

Реализация сервисов Web 2.0 таит в себе ряд неразрешенных проблем:

- доступность динамически обновляемых блоков с информацией зависит от наличия постоянного соединения с Интернетом;
- динамически "собираемые" из различных источников сайты зависят от решений сторонних компаний, качество работы сайта определяется качеством функционирования множества независимых сервисов;
- существующая структура сети и браузеры слабо приспособлены к распределенному решению сложных с вычислительной точки зрения задач;

- конфиденциальные данные, хранимые на серверах, в настоящее время недостаточно защищены от взлома и других действий злоумышленников;
- интерактивный, дружелюбный к пользователю и легко настраиваемый сайт эпохи Web 2.0, как правило, включает возможности сбора статистики о пользователях, их предпочтениях и интересах, что может помочь владельцам ресурса манипулировать сообществом пользователей.

В заключение заметим, что грамотное применение легких в изучении средств HTML, которыми мы займемся со следующей главы, способно решить многие из стоящих перед сетевым сообществом задач без привлечения дополнительных средств и сложных технологий.

ЧАСТЬ II

Создание электронных документов средствами HTML

Часть II книги познакомит вас с основными возможностями формата HTML. Начав с общей структуры документа и его основных тегов, мы последовательно изучим работу с текстом, гиперссылками, списками, таблицами и графикой. Изложение материала сопровождается многочисленными примерами.

Глава 4

Структура документа

Упорядочив свои знания по интернет-технологиям, мы готовы перейти непосредственно к HTML. Повторим самое главное.

"Всемирная паутина" World Wide Web — ведущий сервис Интернета. Документы, составляющие содержание WWW, называются *Web-страницами*, а формат, с помощью которого подготавливаются Web-страницы, — HTML (HyperText Markup Language) или "язык разметки гипертекста".

Документ HTML представляет собой файл типа `html` или `htm`, находящийся на сервере Интернета, в локальной сети или на жестком диске вашего компьютера. Этот файл содержит обычный текст и текстовые команды разметки, называемые *тегами*. С помощью тегов можно решить две основные задачи:

- управлять содержимым документа, включая форматирование текста, разметку заголовков, создание списков и таблиц;
- управлять связями документа с другими ресурсами (изображениями, таблицами стилей, внешними программами, сторонними Web-страницами).

Теги HTML не задают определенные и точные атрибуты форматирования документа, как, например, Microsoft Word. Конкретный вид документа окончательно определяет только программа-*браузер* на вашем компьютере. Необходимость такого подхода связана с разнородностью аппаратного и программного обеспечения устройств, подключенных к Интернету. HTML — *не язык программирования*, хотя Web-страницы могут как сами являться результатом работы серверных программ, так и включать в себя специально подготовленные клиентские программы — скрипты и апплеты.

Документ HTML должен иметь определенную структуру, которую мы изучим в этой главе.

4.1. Тег и его структура

Тег HTML имеет общий вид

```
<имя>содержимое</имя>
```

и действует на все, что расположено между парами треугольных скобок. Все теги, имеющие содержимое, должны *закрываться*, причем закрывающая часть `</имя>` отличается от открывающей `<имя>` только наличием символа "/". Теги могут вкладываться друг в друга *иерархически, но без пересечений*, т. е. допустимо вложение вида `<тег1><тег2></тег2></тег1>`, но не `<тег1><тег2></тег1></тег2>`. Тег вместе с содержимым часто называют *элементом* HTML.

Действие вложенных тегов может *объединяться*, т. е. если внутрь тега, создающего полужирное начертание шрифта, мы вложим тег курсива, то должен получиться полужирный курсив. Аналогично, внутрь тега, создающего ячейку таблицы, мы можем вставить тег подключения картинки — и картинка окажется внутри ячейки. HTML предоставляет большую свободу обращения с тегами, но существуют и ограничения — нельзя же вложить старшую матрешку внутрь младшей!

Теги делятся на *блочные* (block-level) и *текстовые* (inline). Первые могут содержать как текстовые, так и другие блочные теги. При отображении они всегда выводятся *с новой строки*. Вторые могут содержать только текст и другие текстовые теги, но не блочные. При отображении они выводятся *в текущей строке*.

Теги, имеющие содержимое и нуждающиеся в закрывающей части, называют *контейнерными*, а теги без содержимого и закрывающей части — *унарными*.

Внутри открывающей части большинство тегов содержит *атрибуты*, называемые также *параметрами* или *опциями*. Все эти названия просто означают команды, уточняющие действие тега. Какой ширины должна быть ячейка таблицы, формируемая тегом? Где хранится картинка, которую тег должен отобразить? На эти и многие другие вопросы отвечают атрибуты, всегда находящиеся в открывающей части тега. Атрибуты имеют общий вид

```
имя="значение"
```

и разделяются между собой хотя бы одним пробелом, символом табуляции или перевода строки. Если значение атрибута состоит из одного слова, символы двойных кавычек можно не писать, хотя стандарт рекомендует указывать их всегда. Например, элемент вида

```
<p align="justify">Hello</p>
```

описывает контейнерный тег с именем `p`, имеющий один атрибут с именем `align` и значением "`justify`". Содержимое тега — строка текста "`Hello`".

В закрывающей части тега атрибуты не применяются.

Названия всех тегов и атрибутов *нечувствительны* к регистру символов. Тем не менее, лучше придерживаться единообразного их написания.

У каждого тега имеется набор допустимых для него атрибутов. Для многих атрибутов также заранее известен набор значений, которые они могут принимать. Чаще всего атрибуты можно пропускать, тогда браузер выводит документ, придерживаясь правил, принятых "по умолчанию". Существуют и обязательные атрибуты. Например, тег вставки картинки не сможет ее найти, если в атрибуте `src` не будет указано ее местоположение. Как правило, порядок следования атрибутов, допустимых для данного тега, может быть произвольным.

HTML довольно "демократичен", неправильный тег, лишний атрибут, недопустимое вложение тегов обычно просто игнорируются браузером и не приводят к "зависаниям" или сообщениям об ошибках. Разумеется, при этом может произойти неправильное форматирование документа.

4.2. Общая структура документа

HTML-документ состоит из трех основных частей:

- строка декларации типа документа;
- заголовок документа, заключенный в тег `<head>…</head>`;
- тело документа, заключенное в тег `<body>…</body>` или `<frameset>…</frameset>`.

Заголовок и тело документа заключаются в объединяющий их тег `<html>…</html>`.

Строка декларации является служебной и указывает на стандарт разработки документа. Для документа, строго соответствующего стандарту HTML 4.0, она представляется унарным тегом вида

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">.
```

На сегодняшний день поддержка деклараций рекомендуется стандартами, но редко реализуется Web-мастерами, в дальнейшем мы будем ее пропускать.

Заголовок в теге `<head>` содержит информацию об общих свойствах документа. Содержимое тега `<head>` непосредственно не отображается в окне браузера, однако используется браузером при работе с файлом. Заголовок может включать в себя следующие теги:

- титул окна документа (тег `<title>`);
- *метатеги* документа (теги `<meta>`);

- базовый URL-адрес для ссылок (тег `<base>`);
- связь с другими документами (теги `<link>`);
- встроенные таблицы стилей для оформления (тег `<style>`);
- встроенные программы-сценарии клиента (теги `<script>`).

Из всех этих тегов нам необходим пока только `<title>`, остальные будут кратко описаны в *разд. 4.4* и *4.5*. `<title>` представляет собой контейнерный тег без атрибутов, его содержимым может быть любая строка текста, не содержащая других тегов. Именно эта строка будет выведена браузером в заголовке окна или вкладки. Содержимое тега `<title>` должно кратко отражать суть документа, его рекомендуемый размер — не более 60–80 символов. При добавлении ссылки на документ в папку "**Избранное**" браузера именем новой закладки становится именно содержимое тега `<title>`. Многие браузеры создают для каждой закладки отдельный файл, поэтому внутри тега `<title>` нежелательны символы двойных кавычек, `<`, `>`, `&`, `!`, `?` и другие знаки, недопустимые в именах файлов. Составление "правильных" `<title>` и других заголовочных тегов — целое искусство, некоторые аспекты которого мы обсудим в *главе 16*. Пока же просто не будем забывать, что заголовок в теге `<title>` — "самый главный". Возможно, ваш пользователь, уставший ждать загрузки по медленному каналу связи, только его и увидит.

Тело документа содержит теги, предназначенные для отображения браузером, и ограничено контейнерным тегом `<body>…</body>` или `<frameset>…</frameset>`, если документ разбит на несколько окон-*фреймов* (о них мы поговорим в *главе 12*).

В наших первых документах мы будем записывать `<body>` без атрибутов. Тем не менее, существует целый ряд атрибутов, которые можно указать в этом теге. Более того, все описанные далее атрибуты — *общие* для большинства существующих тегов.

Атрибут `id` присваивает элементу имя, уникальное в пределах одного документа. Никакие два элемента не могут обладать одинаковыми `id`. Этот атрибут имеет несколько применений:

- указатель стиля, которым следует оформить элемент;
- закладка для создания гиперссылки;
- указание на конкретный элемент для программы-сценария;
- указатель имени внешнего элемента, подключенного тегом `<object>`.

Например, здесь тег `<p>` с содержимым "`Example`" помечен атрибутом `id` со значением `paragraph1`:

```
<p id="paragraph1">Example</p>.
```

Это позволяет, в частности, управлять содержимым тега `<p>` с помощью стилевых указаний или использовать значение атрибута `paragraph1` для программирования поведения объекта из сценария.

Атрибут `class` указывает, что элемент является членом *определенного класса*. В отличие от `id`, любое количество элементов может относиться к одному классу. Элемент может принадлежать и к нескольким классам — в этом случае значением атрибута будет список имен классов, разделенных пробелами.

В HTML не предусмотрены теги "на все случаи жизни", да это и невозможно. Между тем, очень часто нужно найти, оформить или изменить в документе только данные одного определенного типа (номера телефонов, денежные суммы, названия цветов, все упоминания о кошках и т. д.). Указание атрибута `class` на этапе разработки документа позволяет в будущем более эффективно управлять его данными.

Атрибут `style` позволяет задать стиль оформления конкретного элемента внутри открывающей части тега, например:

```
<p style="font-family:Arial">в этом элементе был изменен стиль оформления</p>.
```

Для использования этого атрибута в заголовок документа должен быть включен *метатег*

```
<meta http-equiv="Content-Style-Type" content="text/css">.
```

Подробнее мы познакомимся с `id`, `class` и `style` при работе со стилями в *главе 14*. Эти три атрибута применимы ко всем тегам, кроме `<base>`, `<head>`, `<html>`, `<meta>`, `<param>`, `<script>`, `<style>` и `<title>`.

Атрибут (не тег!) `title` определяет дополнительное описание для элемента, обычно он служит для выведения всплывающей подсказки (tooltip) при наведении курсора мыши на элемент.

Атрибут `lang` определяет язык, на котором описаны значения остальных атрибутов тега и его содержимое. Пока что применение атрибута ограничено, в перспективе же он призван решать следующие задачи:

- индексирование поисковыми системами с учетом языка содержимого;
- синтез правил произношения и голосовой вывод содержимого в зависимости от указанного языка;
- использование браузерами будущего национальных правил переноса слов, проверки правописания и т. д.

Для указания на русский язык содержимого достаточно написать в открывающей части тега атрибут `lang="ru"`, для английского языка этот атрибут будет иметь вид `lang="en"`.

Атрибут `dir` определяет направление вывода текста: слева направо (`dir="ltr"`, по умолчанию) или справа налево (`dir="rtl"`). Так, русские и латинские буквы выводятся слева направо, а арабские и еврейские — справа налево.

Подобно современным языкам программирования, HTML поддерживает обработку *событий*, которые могут происходить на компьютере-клиенте. Все события связаны либо с нажатием клавиш на клавиатуре, либо с движением мыши и нажатием ее кнопок. Поддерживаемые события приведены в табл. 4.1.

Таблица 4.1. *Стандартные события в HTML*

Имя события	Когда происходит
`onclick`	Щелчок кнопкой мыши на элементе
`ondblclick`	Двойной щелчок кнопкой мыши на элементе
`onmousedown`	Нажатие кнопки мыши на элементе
`onmouseup`	Отпускание кнопки мыши на элементе
`onmouseover`	Попадание курсора мыши на элемент
`onmousemove`	Движение курсора мыши по элементу
`onmouseout`	Выход курсора мыши за пределы элемента
`onkeydown`	Нажатие клавиши на элементе
`onkeypress`	Нажатие и отпускание клавиши на элементе
`onkeyup`	Отпускание клавиши на элементе
`onload`	Происходит один раз, когда браузер загрузил документ в окно (только теги `<body>`, `<frameset>`)
`onunload`	Происходит один раз, когда браузер выгрузил документ из окна (только теги `<body>`, `<frameset>`)

Для обработки события достаточно в открывающей части тега записать атрибут, именем которого является название события, а значением — операторы языка обработки скриптов. Например, указанный далее атрибут `onmouseover` тега `<p>`

```
<p onmouseover="window.alert('Hello!')">подведи ко мне курсор!</p>
```

приведет к тому, что при наведении мыши на текст "`подведи ко мне курсор!`" браузер выдаст окно сообщения с текстом "`Hello!`".

Если документ содержит обработку событий, то язык обработки сценариев должен быть указан в заголовке документа метатегом

```
<meta http-equiv="Content-Script-Type" content="text/javascript">.
```

Обработка событий применима ко всем тегам, кроме `<base>`, `<br>`, `<frame>`, `<frameset>`, `<head>`, `<html>`, `<iframe>`, `<meta>`, `<param>`, `<script>`, `<style>` и `<title>`.

Существуют также атрибуты тега `<body>` для изменения набора цветов документа. Мы изучим их в *главе 9*.

4.3. Наш первый документ

Итак, простейший документ HTML имеет вид, показанный в листинге 4.1.

Листинг 4.1. Простейший документ HTML

```
<html>
<head>
 <title>Простейший документ HTML</title>
</head>
<body>
 Содержимое документа
</body>
</html>
```

Разумеется, "содержимое документа" будет постепенно усложняться. Изучив основные теги HTML и достигнув достаточного уровня сложности примеров, мы еще вернемся в *разд. 8.5* к теме выбора HTML-редактора, способного облегчить нелегкую "ручную" работу с HTML-кодом.

Исходные тексты примеров, записанные на прилагаемом к книге компакт-диске в папках Glava_4, Glava_5 и т. д., также помогут вам в изучении материала. В частности, наш первый документ хранится в папке Glava_4 под именем Simple.html.

Вы можете сравнить размер файла Simple.html с находящимся в той же папке документом SimpleWord.html. Последний получен открытием примера в Microsoft Word 2002 с последующим сохранением в формате HTML. Мало того, что сгенерированный редактором документ в 20 раз больше по объему, он еще и категорически "отказывается" подчиняться пользовательским установкам **Размер шрифта** в Internet Explorer пятой-шестой версий. Правда, в новых браузерах эта проблема решена (там есть возможность произвольно масштабировать всю страницу). Но все-таки, я думаю, понятно, почему для выполнения примеров нам хватит всем знакомых программ Блокнот, Paint, а также имеющегося под рукой браузера.

Полагаю, вы справитесь с задачей как открытия примеров с диска, так и просмотра их в Блокноте. Не уверены? Тогда вот вся нужная информация:

- Для просмотра примера в установленном у вас браузере откройте нужную папку на компакт-диске и сделайте двойной щелчок кнопкой мыши на документе.
- Чтобы открыть исходный текст примера, достаточно щелкнуть правой кнопкой мыши на нужном HTML-документе, выбрать вложенное меню **Открыть с помощью**, а в нем — приложение Блокнот. Если Блокнота там нет, доберитесь в списке программ до последнего пункта **Выбрать программу…** и укажите на Блокнот. Второй путь — щелкнув правой кнопкой мыши на любом свободном месте страницы, открытой в браузере, выбрать команду **Просмотр HTML-кода** (или **View Page Source**).
- Набрав HTML-документ в Блокноте "с нуля", достаточно в меню **Файл** выбрать пункт **Сохранить как…**, перейти к нужной папке, в поле ввода **Имя файла:** указать желаемое имя с добавлением типа html, например, first.html, в списке **Тип файла:** выбрать значение **Все файлы** и нажать клавишу <Enter>.

При работе с текстами примеров и при создании собственных Web-страниц не забывайте о следующем:

- HTML форматирует документ только с помощью тегов, построчное расположение слов и тегов не имеет значения;
- при "ручном" наборе текста HTML-документа следует избегать переносов слов по слогам и выравнивания текста с помощью пробелов или табуляций;
- при "ручном" наборе удобнее, когда содержимое каждого блочного тега сдвинуто на один-два символа вправо, а закрывающая часть тега находится точно под открывающей;
- необязательно стирать "неудавшийся" фрагмент — его можно *закомментировать*. Комментарий в HTML начинается с символов `<!--` и заканчивается символами `-->`. Комментарий может занимать любое число строк, он не влияет на отображение документа, а только поясняет его содержимое или служит для временного скрытия фрагментов кода.

4.4. Метатеги и их назначение

Метатеги предназначены для описания свойств документа в целом.

Спецификации HTML пока не содержат стандартного списка метатегов, но среди них есть общепринятые и рекомендуемые к использованию.

Каждый тег `<meta>` содержит пару атрибутов: *название свойства* в атрибуте `name` и его *значение* в атрибуте `content`, например, следующий распространенный метатег задает имя автора документа:

```
<meta name="Author" content="А.С. Пушкин" lang="ru">.
```

Атрибут `lang` указывает здесь язык, на котором описано значение свойства.

Многие поисковые системы используют свойства `description` (описание документа) и `keywords` (ключевые слова) для извлечения дополнительной информации. При этом описание должно быть кратким (до 180–200 символов), а список ключевых слов следует разделять запятыми:

```
<meta name="Description" content="Изучение основных возможностей языка
гипертекстовой разметки HTML">
<meta name="Keywords" content="HTML, Web, www, учебник">
```

Достаточно часто поддерживается свойство `robots`, содержащее указания для поисковых машин. Значение свойства `robots` — это список директив, разделенных запятыми:

- `index` — страница должна быть проиндексирована поисковой машиной;
- `noindex` — страница не должна быть проиндексирована;
- `follow` — следовать по гиперссылкам на странице;
- `nofollow` — не следовать по гиперссылкам на странице;
- `all` — то же, что "`index, follow`" (принято по умолчанию);
- `none` — то же, что "`noindex, nofollow`".

Вместо атрибута `name` метатег может содержать `http-equiv`, указывающий, что в атрибуте `content` указана информация прикладного протокола HTTP. Таким способом задается ряд важных свойств документа.

```
<meta http-equiv="Content-Type" content="text/html; char-
set=windows-1251">.
```

Здесь для текущего документа указана русифицированная кодировка Windows (кодовая страница 1251). Для задания другой кодировки ее можно вписать вместо "`windows-1251`".

```
<meta http-equiv="Expires" content="Sun, 6 Jan 2008 09:00:00 GMT">.
```

Здесь заданы дата и время истечения срока действия документа (по Гринвичу). Тег предназначен для того, чтобы информировать браузеры, когда следует выполнить повторную загрузку страницы:

```
<meta http-equiv="Refresh"
content="10;URL=http://www.domain.com/new.html">.
```

Браузеры поймут эту запись как инструкцию ожидать 10 секунд, а затем загрузить новый документ, расположенный по адресу `www.domain.com/new.html`.

Тег может оказаться полезным при "переезде" страницы на другой URL-адрес.

Метатеги для задания языков стилей и сценариев были рассмотрены в *разд. 4.2*.

4.5. Теги для связи документов

Тег `<base>` задает в обязательном атрибуте `href` адрес URL, который будет считаться "базовым" для отсчета ссылок из текущего документа.

Если в документе нет тега `<base>`, его собственный URL-адрес считается базовым по умолчанию. Обычно этого вполне достаточно и на практике `<base>` употребляется нечасто.

Тег `<link>` определяет *связь* между документами. Заголовок документа может содержать любое количество этих тегов. Поддержка тегов `<link>` до сих пор не общепринята.

Атрибут тега с именем `rel` определяет *прямую ссылку*, а атрибут `rev` — *обратную*. При этом сам URL-адрес ссылки содержится в атрибуте с именем `href`. Существуют также другие атрибуты тега, указывающие на язык и кодировку ссылки.

```
<link rel="Next" href="page2.html">
```

здесь указано, что документ с именем `page2.html` является следующим в коллекции (прямая ссылка), а в теге

```
<link rev="Subsection" href="main.html">
```

указано, что текущий документ — подраздел документа с именем `main.html` (обратная ссылка).

Значение атрибутов `rel` и `rev` — список *типов ссылок*, разделенных пробелами. Вот некоторые из общепринятых типов ссылок:

- Alternate — альтернативная версия документа;
- Appendix — ссылка на приложение;
- Chapter — ссылка на главу в коллекции документов;
- Contents — ссылка на документ, содержащий оглавление;
- Copyright — ссылка на документ, содержащий сведения об авторском праве;
- Glossary — ссылка на документ, содержащий словарь терминов;
- Help — ссылка на документ, содержащий справочную информацию;
- Index — ссылка на документ, содержащий алфавитный указатель;
- Made — ссылка на E-mail автора;

- Next — ссылка на следующий документ в коллекции;
- Prev — ссылка на предыдущий документ в коллекции;
- Section — ссылка на документ, служащий разделом в коллекции;
- Start — ссылка на первый документ в коллекции;
- Subsection — ссылка на документ, служащий подразделом в коллекции.

Важное применение тега `<link>` — подключение к документу таблиц стилей. Эти возможности, а также элемент `<style>`, служащий той же цели, будут описаны в *главе 14*.

Элемент `<script>` будет нами изучен в *главе 15*.

Глава 5

Форматирование текста

Не нуждается в доказательствах утверждение, что текст — основа Web-страницы. В большинстве случаев пользователь выходит в Сеть именно в поисках текстовой информации по интересующей его теме. Даже если сам пользователь так не думает, а просто ищет в Интернете видеоклипы или заставки для мобильного телефона, ему не избежать чтения текста ссылок, рекламных баннеров (почему-то они оказываются на самом видном месте!) либо результатов собственного запроса к Яndex. Поэтому от того, как написан, размечен и преподнесен текст Web-страницы, во многом зависит ее дальнейшая судьба.

Просто разыскать или набрать в редакторе текст, предназначенный для размещения в Интернете — задача, с которой справится каждый из нас. Технически и стилистически правильно разметить текст тегами HTML — главное, чему мы научимся в этой главе.

HTML имеет развитые средства работы с текстом, которые можно разделить на несколько групп.

- Теги *структурного форматирования* разбивают документ на содержательные и смысловые части, назначая им различные способы оформления, такие как разделы, абзацы и заголовки различного уровня.
- Теги *физического форматирования* определяют, *как именно* будет отображаться текст. С их помощью можно задать шрифт, начертание, цвет, размер и некоторые другие характеристики текста.
- Теги *логического форматирования* предназначены для выделения в тексте документа отдельных смысловых фрагментов, таких как цитата, листинг программы, важный по смыслу или помеченный для удаления фрагмент текста. Согласно спецификации, эти теги не определяют непосредственно, как будет отображен заключенный в них текст, но относят его к тому или иному классу.

Мы будем изучать эти группы не только по приведенной схеме, но и с учетом значимости или распространенности входящих в них тегов. Кроме того, некоторые из рассмотренных далее тегов нельзя однозначно отнести ни к одной из перечисленных групп.

5.1. Абзацы, выравнивание, переносы

Как и в Microsoft Word, основой структуры текста в HTML является абзац. Для выделения абзаца предусмотрен контейнерный тег `<p>…</p>` (от англ. "paragraph" — абзац). Визуально тег создает сверху и снизу от заключенного в него текста вертикальный отступ, служащий разделителем абзацев. По умолчанию абзац занимает всю доступную ширину окна или блока с выравниванием текста по левой границе. Изменить способ выравнивания текста в абзаце позволяет основной атрибут тега, называемый `align` (от англ. "to align" — выравнивать, ставить в ряд). В табл. 5.1 приведены его допустимые значения.

Таблица 5.1. *Значения и действия атрибута align*

Значение	Действие
`left`	Выравнивание текста по левому краю; это значение атрибута действует по умолчанию и его можно не указывать
`right`	Выравнивание по правому краю
`center`	Выравнивание по центру
`justify`	Выравнивание по ширине блока, занимаемого текстом

В спецификации HTML атрибут `align` помечен как "нежелательный", несмотря на это, он широко распространен. На рис. 5.1 приведен пример использования тега `<p>` с различными значениями атрибута `align`. Этот же пример (его код содержится в листиге 5.1) вы можете выполнить, открыв документ Align.html из папки Glava_5 компакт-диска.

Листинг 5.1. Использование атрибута `align` для выравнивания

```
<html>
<head><title>Выравнивание текста абзаца атрибутом Align</title></head>
<body>
<p>16 июля 2007 года</p>
<p align="center">Дорогая Алина!</p>
```

```
<p align="justify">Позвольте поздравить Вас с днем рождения и пожелать
всего наилучшего! 18 лет - прекрасный возраст, время надежд и свершений,
желаю Вам всегда оставаться такой же милой, жизнерадостной и внимательной
к окружающим.</p>
<p align="right">Искренне Ваш, Максим</p>
</body></html>
```

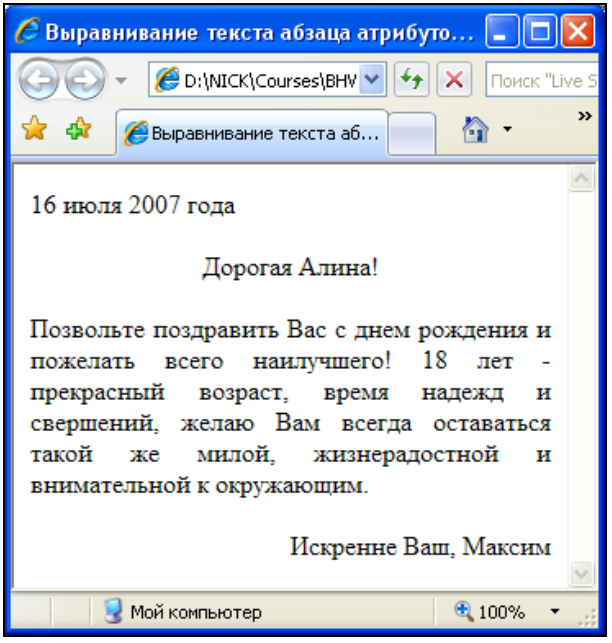

Рис. 5.1. Выравнивание текста абзаца

Необходимо обратить внимание на следующее:

- тег `<p>` не должен содержать других блочных элементов, например, разделов и заголовков;
- браузер игнорирует разрывы строк и лишние пробелы внутри абзаца, размещая его текст как можно плотнее и целиком заполняя текущую строку экрана прежде, чем начать новую;
- значение атрибута `align="justify"` появилось позже других, очень старые браузеры не отобразят его действие.

Внутри тега абзаца могут находиться более "мелкие" теги физического и логического форматирования, описанные в *разд. 5.2* и *5.3*. Разумеется, это

не мешает поместить в текст абзаца гиперссылку или связать с ним изображение, а также использовать внутри абзаца любой другой текстовый элемент.

Тег `<p>` может также применяться для форматирования текста, помещенного в ячейку таблицы или фрейм (*см. главы 8* и *12*).

Часто тег абзаца не закрывают, это разрешено спецификацией и вполне допустимо, если за ним следует новый тег `<p>`. Однако последовательность абзацев лучше закрыть.

Создавать в документе "пустое место" с помощью нескольких тегов `<p>` не рекомендуется, т. к. пустые абзацы игнорируются браузерами.

Как правило, браузеры не переносят слова по слогам. Тем не менее, существуют теги для управления построчным размещением текста и переносами.

Тег `<br>` (break) создает принудительный разрыв строки. Закрывать этот тег запрещено. Несколько тегов `<br>` могут быть указаны друг за другом для создания соответствующего числа пустых строк. Если разрыв строки следует сразу за открывающей частью любого тега или непосредственно перед его закрывающей частью, то он может быть проигнорирован браузером. Следует также учесть, что при изменении ширины окна или размера шрифта наличие `<br>` может привести к появлению "лишних" строчек на экране.

Атрибут тега `clear` определяет, где в браузере должна появиться строка, следующая за переводом строки, вызванным действием тега. Возможные значения атрибута: `none` (по умолчанию), `left` (у левого поля), `right` (у правого поля) и `all` (у любого поля). Атрибут учитывает прикрепляемые объекты (изображения, таблицы и т. д.). Спецификацией атрибут `clear` помечен как "нежелательный" и встречается редко.

Листинг 5.2 и рис. 5.2 иллюстрируют применение тега `<br>`. На компакт-диске с примерами вы можете найти этот код в документе Glava_5\Break.html.

Листинг 5.2. Использование тега `<break>`

```
<html><head>
 <title>Использование тега разрыва строки</title>
</head>
<body>
<p>Это стихотворение написал Иннокентий Анненский:<br>
Среди миров, в мерцании светил<br>
Одной Звезды я повторяю имя...<br>
Не потому, чтоб я Ее любил,<br>
А потому, что я томлюсь с другими.<br>
<br>
```

```
И если мне сомненье тяжело,<br>
Я у Нее одной молю ответа,<br>
Не потому, что от Нее светло,<br>
А потому, что с Ней не надо света.<br>
<br></p>
</body></html>
```

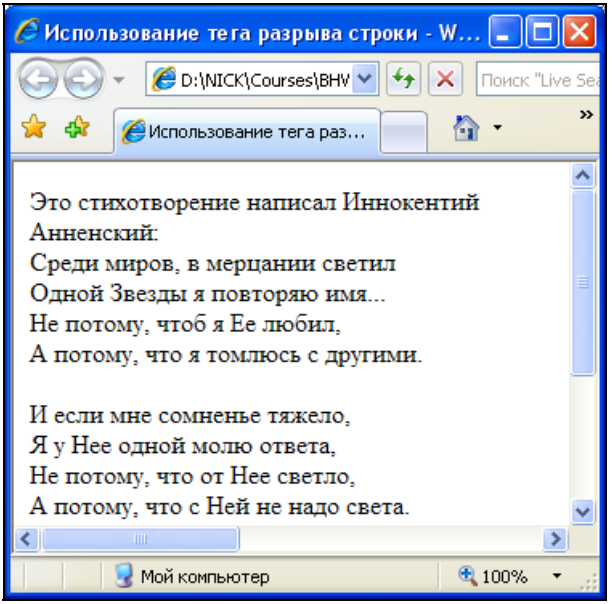

Рис. 5.2. Использование тега `<break>`

Первое и последнее вхождения тега `<br>` в этом документе — лишние, при выводе они были проигнорированы браузером.

Действие тега `<nobr>…</nobr>` противоположно `<br>` — он запрещает переносить на новую строку заключенный в него текст. С точки зрения многих браузеров, строку можно переносить на любом разделителе, так что слова "интернет-каталог" и "Web-технология" могут оказаться разорванными в месте дефиса. Слова или строки, перенос которых может повредить целостности текста, можно заключить в тег `<nobr>`. Платой за использование тега может оказаться нарушение выравнивания текста или появление горизонтальной полосы прокрутки в окне. Тег считается нестандартным, но не имеет эквивалентной замены в стандарте. Тег `<nobr>` обсуждается в стандарте HTML 5.0, возможно, он будет "реабилитирован".

Тег `<wbr>` (word break), не имеющий закрывающей части, употребляется только внутри `<nobr>` и указывает, что в этом месте *возможен разрыв строки*. Тег также пока не включен в стандарт. Закрывающей части у `<wbr>` нет. Приведем пример использования данных тегов:

```
<nobr>Написав эту длинную строку, мы пытаемся запретить браузеру переносить ее. Однако в этом месте<wbr> перенос возможен.</nobr>.
```

Стандарт рекомендует для запрета разрыва строки между двумя словами вставлять *неразрывный пробел*, записываемый как ` `. Последовательность символов ` ` можно использовать для задания нескольких идущих подряд пробелов или для создания в начале абзаца "красной строки":

```
<p>Слова  разделены  двумя  пробелами.
<p>   Абзац начат с "красной строки".
```

С ` ` и другими *специальными символами* мы подробнее познакомимся в *разд. 5.5*.

5.2. Физическое форматирование — шрифты и начертания

Прежде чем говорить о тегах для работы со шрифтами, кратко обсудим то, что необходимо знать о самих шрифтах. Все современные шрифты — *векторные*. Это означает, что для каждого символа шрифта создается и хранится контурная модель, состоящая из геометрических примитивов, таких как отрезки прямых и дуги окружностей. Пользуясь этой моделью и учитывая требуемый в текущий момент размер символа, компьютер рассчитывает и выводит на экран поточечное изображение символа. В частности, такой подход позволяет сохранять видимую гладкость букв шрифта при задании различных его размеров.

Немногочисленные *растровые* (точечные) шрифты, символы которых хранились в виде "матриц", состоящих из отдельных точек, и не "умели" произвольно масштабироваться, оставлены в современных версиях Windows, в основном, для совместимости со старыми программами. Исторически эти шрифты предшествовали векторным, иногда они встречаются и сегодня, например, при черновой печати.

Помимо гарнитуры, шрифты различаются между собой такими свойствами, как *размер*, *начертание*, *кернинг* (способ выравнивания межсимвольных интервалов) и некоторыми другими.

Стандартная единица измерения размера шрифта называется *пунктом* (pt). 72 пункта составляют 1 дюйм или 2,54 см. В отличие от Microsoft Word и других популярных программ, позволяющих непосредственно задать раз-

мер шрифта в пунктах, стандарт HTML использует для его указания безразмерные единицы.

Начертания — это "варианты" представления символов шрифта. Каждый шрифт обычно имеет набор начертаний, некоторые из них, такие как **полужирный**, *курсив* и подчеркивание, реализованы в тегах языка HTML.

Пользоваться кернингом и другими расширенными свойствами шрифта HTML непосредственно не позволяет, хотя описанные в *главе 14* таблицы стилей во многом решают эту проблему.

Несмотря на обилие шрифтов, как входящих в состав Windows, так и созданных сторонними разработчиками, скорее всего, ваш браузер "считает" стандартными всего два-три шрифта:

- Times New Roman — стандартный шрифт "печатной машинки"; именно этим шрифтом, если в коде страницы или настройках браузера не указано другое, выводится на экран основной текст страницы;
- `Courier New` — это стандартный *моноширинный* шрифт (все его буквенно-цифровые символы имеют на экране одну и ту же ширину). Поэтому Courier New применяется для отображения листингов программ, блоков заранее отформатированного текста, который не нужно форматировать средствами HTML, и т. п.;
- Arial — обычно служит для отображения заголовков. Символы этого шрифта лишены *отточий* и *засечек* ("тонких деталей" букв), они более гладкие, чем обычные символы Times New Roman.

Продуманные выделения тем или иным шрифтом способны не только облегчить восприятие текста читателем-человеком, но и сделать вашу страничку более привлекательной для поисковых машин. В *главе 16* мы вернемся к правилам работы с текстовым содержимым, пока же обсудим главное.

- Форматируйте текст. Нет ничего скучнее, чем большой объем текста, однообразно выглядящего на страничке. Если в обычной жизни мы вынуждены постоянно читать такие тексты, то в окне браузера они вызывают немедленное желание щелкнуть по кнопке **Назад**. К исключениям относятся литературные произведения и любого рода документация, которые лучше выглядят "как есть".
- Форматируйте в меру. Использование на одной странице более трех видов шрифтов (начертаний) приведет лишь к тому, что глаза пользователя начнут "разбегаться". Следует выделять шрифтовыми средствами лишь действительно важные места в документе — заголовки всех видов, определения, термины, отдельные фразы, имена или названия, к которым нужно привлечь внимание читающего, в крайнем случае, просто первые буквы абзацев.

- Отделяйте дизайн от содержимого. Написать текст и разметить его — это две разные задачи. Как правило, форматирование и дизайнерские изыски становятся уместными лишь тогда, когда содержание текста окончательно определено. Независимо от способа создания сайта, следует иметь под рукой продуманный набор стилей, которые несложно применить к готовому документу. Многие из необходимых стилей уже предоставляют теги логического форматирования.

Следует сразу же смириться с мыслью, что возможности физического форматирования в HTML куда более ограничены, чем в обычном документе Word. Например, без привлечения стиля (или описанного в *разд. 5.1* "хитрого приема") нельзя заставить тег абзаца соблюдать "красную строку" или хотя бы уменьшить расстояние до следующего абзаца. Задавая "конкретный" размер шрифта, нельзя быть уверенным, что пользователь увидит на экране то же, что и вы. Более того, сам указанный разработчиком шрифт может быть не установлен на компьютере клиента, что приведет к выводу текста стандартным шрифтом.

Вспомним также о спецификации, отдающей предпочтение логическому форматированию перед физическим. Однако теги физического форматирования столь просты и широко распространены, что в Интернете редко встретишь страницу, где бы они не использовались.

Главный тег этой группы — `<font>...</font>` — позволяет задать желаемый для отображения шрифт и его параметры. Основные атрибуты тега перечислены в табл. 5.2.

Таблица 5.2. *Атрибуты тега <font>*

Атрибут	Описание
`face`	Определяет список разделенных запятыми названий шрифтов, поиск которых браузер должен выполнить в порядке приоритета
`color`	Устанавливает цвет текста. Возможные значения — имена цветов (`gray`, `green`, `red` и т. п.) или их 16-ричные коды вида `#xxxxxx`, где `x` — символы из набора `0,1,…,9,A,B,C,D,E,F`. Подробнее см. *главу 9* и *приложение 3*
`size`	Устанавливает размер шрифта. Возможные значения атрибута: • целое число от `1` до `7`. Устанавливает фиксированный размер шрифта, представление которого зависит от браузера. Не все браузеры могут представлять все 7 размеров; • относительное изменение размера шрифта. Значение "`+1`" означает "на один размер больше". Значение "`-3`" означает шрифт на три размера меньше. Все размеры относятся к шкале от 1 до 7

Как указывалось в *главе 4*, любые из атрибутов тега могут быть опущены или заданы совместно. Таким образом, браузер, встретив тег

```
<font face="Arial" size="5" color="gold">привет</font>,
```

попытается вывести слово "привет" шрифтом Arial золотого (gold) цвета с относительным размером, равным 5. Как правило, принятый по умолчанию размер шрифта равен 3, в браузере Internet Explorer ему соответствует размер 12 пунктов, если пользовательская настройка **Размер шрифта** установлена в значение **Средний**.

Если HTML-код содержит строку

```
<font face="Tahoma, Verdana, Arial">Мои увлечения</font>,
```

то браузер выведет фразу "Мои увлечения" шрифтом Tahoma, если он установлен, в противном случае будет выбран шрифт Verdana, а при его отсутствии — Arial. Если не установлен ни один шрифт из списка, будет использован шрифт по умолчанию.

При относительном указании размера шрифта, во всех случаях он отсчитывается от базового размера. Так, встретив конструкцию тегов

```
3<font size=+2>5</font><font size=+4>7</font><font size=+6>9</font>,
```

браузер выведет цифру "3" размером по умолчанию, цифры "5" и "7" будут иметь соответствующий размер шрифта, цифра "9" будет отображена шрифтом размера 7, потому что размера 9 не существует.

Для указания типа, размера и цвета шрифта, установленных по умолчанию, существует тег `<basefont>`. Его атрибуты и их возможные значения точно такие же, как у `<font>`, только закрывающей части этот тег не имеет. Указанные в `<basefont>` параметры шрифта браузер принимает как обязательные для всего документа, за исключением тех мест, где они переопределены тегом `<font>` или заголовком. Тег `<basefont>` может появляться в документе неоднократно, каждый раз переопределяя параметры шрифтов с места своего появления. Он допустим и в части `<head>` документа.

Для работы с начертаниями шрифта существует целая группа контейнерных тегов. Все они просты и понятны в действии, для удобства сведем информацию о них в табл. 5.3. В связи с тем, что многие из перечисленных в табл. 5.3 тегов спецификация считает "устаревшими" или "нерекомендуемыми", для них приводится список альтернативных средств HTML, создающих тот же видимый эффект.

Действие вложенных тегов группы обычно объединяется. Например, вложение тегов `<b><i>полужирный курсив</i></b>` или `<i><b>полужирный курсив</b></i>` создаст полужирный курсив.

Таблица 5.3. *Основные теги физического форматирования*

Тег	Описание	Альтернативные средства
`<tt>…</tt>`	Указывает, что содержимое должно отображаться "телетайпным" (моноширинным) шрифтом	`<code>`, `<samp>`, `<kbd>`
`<b>…</b>`	Создает полужирное начертание шрифта	`<strong>`
`<i>…</i>`	Создает курсивное начертание шрифта	`<em>`, `<dfn>`, `<var>`, `<cite>`
`<big>…</big>`	Указывает, что содержимое должно отображаться крупным шрифтом	Указание в стиле `font-size: larger`
`<small>…</small>`	Указывает, что содержимое должно отображаться мелким шрифтом	Указание в стиле `font-size: smaller`
`<strike>…</strike>` `<s>…</s>`	Создает перечеркнутый текст. Не рекомендуется	`<del>` или указание в стиле `text-decoration: line-through`
`<u>…</u>`	Создает подчеркнутый текст. Не рекомендуется	`<ins>` или указание в стиле `text-decoration: underline`
`<sup>…</sup>`	Создает надстрочный текст ("верхний индекс")	Нет
`<sub>…</sub>`	Создает подстрочный текст ("нижний индекс")	Нет

Теги начертаний должны корректно вкладываться. Представление начертаний может зависеть от браузера. Браузер может придерживаться собственных ограничений на размер шрифта, так что конструкции

```
<font size=7><big>Размер шрифта больше 7!</big></font>
<font size=1><small>Размер шрифта меньше 1!</small></font>
```

не обязаны на самом деле отображать шрифты "больше 7" и "меньше 1" (хотя в Mozilla Firefox и Netscape это срабатывает).

Грамотный выбор начертаний дает возможность записывать средствами HTML многие выражения, которые невозможно записать обычным текстом,

например, несложные формулы или особенности естественного письма в некоторых языках:

```
<p>Формула воды - это <b><font face=Arial>H<sub>2</sub>O</font></b>
<p>"Первый" по-английски можно записать как 1<sup>st</sup>.
```

Посмотреть изученные теги "в действии" вы можете, открыв документ `Glava_5\Fonts.html` с компакт-диска.

5.3. Логическое форматирование — расставляем акценты

Главное, что мы вынесли из предыдущего раздела, — использовать теги форматирования текста можно и нужно, но всегда следует помнить об относительности всех "шрифтовых указаний" в HTML. Сугубо *относительные* размеры шрифтов могут на первых порах смущать человека, привыкшего к конкретности программ Microsoft Office. "Пересчитать" эти размеры в пункты или миллиметры нельзя даже для конкретной версии конкретного браузера — ведь все интернет-обозреватели позволяют пользователю самому выбрать вид основного шрифта! Поэтому главное отличие HTML-верстки от обычной — *мы не знаем, как это будет выглядеть у пользователя*.

Соответственно, HTML-верстка должна быть больше *логической*, чем физической, как уже подчеркивалось ранее. Только средства логического форматирования позволяют гибко управлять представлением документа — ведь конкретный *способ отображения* тега можно легко изменить с помощью таблиц стилей, а *логика* документа, понимаемая, кстати, и современными поисковыми машинами, останется той же!

Средства логического форматирования обладают теми же "недостатками", что и средства физические — они не позволяют указать размер заголовка в пунктах или определить фиксированный размер шрифта. Однако с точки зрения свободы пользователя в обращении с документом эти особенности следует считать скорее преимуществами.

Видимые результаты действия различных тегов этой группы могут совпадать как между собой, так и с результатами применения тегов физического форматирования. Спецификация HTML не определяет, как конкретно должен выглядеть фрагмент текста, помеченный тегом `<strong>` ("сильный"), но взамен четко устанавливает, что текст, помеченный этим тегом, предназначен для "более сильного выделения". Например, синтезатор речи, "читающий" документ HTML, может увеличить громкость воспроизведения слова, выделенного этим тегом. Со словом, помеченным тегом `<b>` и визуально выглядящим так же, синтезатор поступать подобным образом не обязан. Давайте

рассмотрим эти замечательные теги подробнее. В табл. 5.4 приведено описание и область применения основных тегов логического форматирования.

Таблица 5.4. *Основные теги логического форматирования*

Тег	**Описание**	**Область применения**
`<em>…</em>`	Смысловое выделение текста	Важные фрагменты текста
`<strong>…</strong>`	Сильное выделение текста	Более важные фрагменты текста, чем помеченные `<em>`
`<cite>…</cite>`	Цитата или ссылка на другие ресурсы	Названия книг, журналов, газет и других источников
`<dfn>…</dfn>`	Определение	Определение термина, встречающегося в тексте
`<code>…</code>`	Фрагмент программного кода	Короткие фрагменты программ в основном тексте
`<samp>…</samp>`	Вывод примера программ, сценариев и т. д.	Отметка текста, выдаваемого программой
`<kbd>…</kbd>`	Ввод пользователя	Отметка текста, вводимого пользователем с клавиатуры и т. п.
`<var>…</var>`	Переменная или аргумент программы	Имена переменных, номера версий программ и т. п.
`<abbr>…</abbr>`	Сокращение	Маркировка аббревиатур и акронимов соответственно. Атрибут `title` содержит расшифровку сокращения
`<acronym>…</acronym>`	Акроним (аббревиатура, произносимая как единое слово)	
`<blockquote>…</blockquote>`	Длинная цитата (содержимое в отдельном блоке)	Цитирование. Атрибут `cite` может указывать URL источника цитаты
`<q>…</q>`	Короткая цитата (содержимое в тексте)	
`<ins>…</ins>`	Вставка текста	Маркировка изменений. Атрибут `cite` может указывать на URL документа, объясняющего причину исправлений, атрибут `datetime` — содержать время их внесения
`<del>…</del>`	Удаление текста	

Первые четыре тега в табл. 5.4 дают возможность смыслового выделения любого слова или фразы, затем четыре тега посвящены разметке компью-

терных данных, следующие два — работе с сокращениями, далее два — цитированию и еще два — правке текста. Как раз последние три пары тегов и требуют пояснений.

Как уже указывалось в *главе 4*, атрибут `title`, значением которого может быть любая строка текста, применим не только к тегам `<abbr>` и `<acronym>`, но и к большинству других. Он определяет "титул" тега и часто интерпретируется браузерами в виде всплывающей подсказки, которая появляется на экране при наведении курсора мыши на помеченный этим атрибутом объект. Что выглядит естественней, чем расшифровка аббревиатуры, появляющаяся при наведении на нее курсора? Также полезным может оказаться использование `title` в атрибутах ссылки, картинки или внешнего объекта, как дополнительного пояснения пользователю. Пока же применим его к цитированию:

```
<p>Как справедливо сказал Станислав Лем, <blockquote
cite="http://www.lib.ru" title="Цитата из книги &quot;Осмотр на
месте&quot;">любая возможная информация содержится в компьютерных инвен-
тарях благ, а ее недоступность обусловлена лишь ужасающей избыточностью
накопленных сведений</blockquote>.
```

При выводе цитата будет отображена как блок текста, сдвинутый вправо. По крайней мере, так обстоит дело во всех известных мне браузерах.

Обратите внимание на "странные" символы `"`, в которые заключено название книги-источника. Дело в том, что значение атрибута в нашем случае должно быть записано в кавычках, но оно и само содержит кавычки. Поэтому нам пришлось применить служебный символ двойной кавычки, записываемый как `"`.

Internet Explorer до наведения курсора мыши никак не выделяет объект, содержащий атрибут `title`, некоторые другие браузеры выделяют такие объекты пунктирным подчеркиванием.

Допустимые в HTML строки, описывающие дату и время, должны быть указаны в формате `ГГГГ-ММ-ДДТчч:мм:ссУЧП`, где поле `УЧП` содержит информацию о часовом поясе, год — обязательно четырехзначный, остальные поля — двузначные, а латинская буква `T` — не опечатка, но указатель на слово "time" (время). Например, изменение, внесенное в документ 1 сентября 2007 г. в 14:05 по московскому времени, может быть помечено атрибутом `datetime="2007-09-01T14:05:00+03:00"`. Указание значения `УЧП` в виде `+03:00` означает, в нашем случае, сдвиг часового пояса от времени по Гринвичу.

Не все документированные возможности логических тегов поддерживаются современными браузерами. Например, встретив тег цитирования `<q>`, браузеры должны сами заключать цитату в кавычки, чего некоторые из них не делают. Проверить, как обстоит дело с вашим браузером, вы можете, запустив документ Glava_5\Logical.html с компакт-диска. Там приведены примеры применения всех описанных здесь тегов.

5.4. Предварительно отформатированный текст

Тег `<pre>…</pre>` содержит *предварительно отформатированный текст*. Это означает, что при его отображении:

- пробелы выводятся без сжатия;
- не производится автоматический перенос слов на новую строку;
- используется моноширинный шрифт (обычно `Courier New`).

Тег полезен при выводе любого текста, в котором существенно размещение слов по строкам: фрагментов программ, стихов, написанных "лесенкой", или любого другого текста, выровненного по количеству символов в строке и содержащего переносы слов.

Внутри блока `<pre>` не следует использовать рисунков, объектов и тегов форматирования текста, кроме того, его не рекомендуется изменять с помощью стилей. Вывод отформатированного текста всегда начинается с нового абзаца.

Атрибут `width="N"`, где `N` — положительное число, дает браузеру подсказку относительно желаемого числа символов в строке отформатированного текста. Браузер может учитывать эту информацию для выбора размера шрифта или создания отступа. Этот атрибут часто не поддерживается, так что надеяться на него не стоит. Кроме того, при указании значения атрибута, меньшего, чем реальное количество символов в строке, обозреватель может испортить вид текста, разорвав строки.

Аналогичные задачи решали устаревшие теги `<listing>`, `<plaintext>` и `<xmp>`, которых следует избегать. Пример Glava_5\Pre.html, имеющийся на компакт-диске, иллюстрирует использование тега `<pre>`.

5.5. Специальные символы

Некоторые символы в языке HTML зарезервированы и должны представляться специальным образом. Прежде всего, это символы, используемые HTML для маркировки тегов и значений атрибутов (табл. 5.5).

Отдельная пара двойных кавычек или знак <, встретившийся вне тега, скорее всего, не введут ваш браузер в заблуждение. Но ввести в документ символ авторского права, обозначение фунта стерлингов или разместить в тексте Web-страницы "исходный код" HTML-документа невозможно без применения специальных символов.

Таблица 5.5. *Специальные символы HTML*

Символ	Обозначение в HTML
Левая угловая скобка (<)	`<`
Правая угловая скобка (>)	`>`
Амперсанд (&)	`&`
Двойная кавычка (")	`"`

Для многих символов, отсутствующих на клавиатуре и вводимых в Microsoft Word через окно **Символ** верхнего меню **Вставка**, в HTML также есть специальные обозначения. Например, символ авторского права © представляется как `©`.

При желании авторы могут пользоваться и числовой кодировкой символов. При этом каждый специальный символ задается своим десятичным или шестнадцатеричным кодом в виде `&#xxx;`, где `xxx` представляет собой десятичную или шестнадцатеричную запись кода символа в стандарте Unicode. Например, наш знак `©` может быть записан кодами `©` или `©`.

Кратко остановимся на специальных символах, служащих для работы с пробелами и переносами. Иногда авторам нужно избежать перехода на новую строку между двумя определенными словами. Запись ` ` (` ` или ` `) действует как неразрывный пробел. В *разд. 5.1* мы уже использовали этот специальный символ для создания в абзаце "красной строки", он применим и во всех остальных случаях, где пробел имеет значение. Поместив символ ` ` в тег абзаца, можно получить на странице вертикальный отступ:

```
<p> </p>.
```

Стандартный и присутствующий на клавиатуре символ дефиса понимается в HTML как обычный печатаемый символ, а вот символ "мягкого переноса" `` указывает браузеру, где возможен разрыв строки, и должен обрабатываться по следующим правилам:

- если по мягкому переносу выполняется разрыв строки, он отображается дефисом;
- если по мягкому переносу разрыв строки не выполняется, он не отображается;
- при поиске или сортировке слов мягкий перенос игнорируется.

В написании специальных символов имеет значение регистр символов. Например, `†` и `‡` — это два разных символа.

Перечень важнейших специальных символов приведен в *приложении 2*.

5.6. Текстовые блоки и разделы

HTML позволяет гибко управлять структурой документа, разбивая его на *разделы*. Общий механизм добавления в документ структуры обеспечивают теги `<div>…</div>` и `<span>…</span>`. Разница между ними в том, что `<div>` является блочным тегом, а `<span>` — текстовым. Сами по себе эти теги не налагают никаких требований на представление информации, хотя браузеры обычно помещают символ перевода строки до и после содержимого тега `<div>`. Таким образом, с помощью `<div>`, в отличие от `<span>`, нельзя задавать отдельные свойства оформления внутри абзаца. Назначив с помощью атрибута `class` имена тем или иным элементам документа, мы можем позднее использовать полученные классы для форматирования документа с помощью таблицы стилей (подробнее об этом рассказано в *главе 14*).

Оба тега должны закрываться, а также могут использовать атрибут `align` со стандартными значениями, приведенными в *табл. 5.1*. Указанный в теге `<div>` способ выравнивания текста распространяется на все абзацы этого раздела, если в абзаце в явном виде не задан другой способ выравнивания.

Тег `<center>…</center>` определяет блок, содержимое которого центрируется *по горизонтали*. Во всех распространенных браузерах этот тег работает точно так же, как `<div align="center">…</div>`.

Ни один документ не обходится без *заголовков* различного уровня. В стандарте HTML определено целых 6 уровней заголовков, маркируемых тегами от `<h1>…</h1>` до `<h6>…</h6>` соответственно. Заголовок первого уровня `<h1>` — самый старший, например, название части книги, заголовок второго уровня `<h2>` может быть использован для маркировки глав, входящих в часть, и т. д.

Заголовок первого уровня выводится довольно крупным шрифтом, тогда как содержимое тега `<h6>` оказывается меньше размера стандартного текста.

Если стандартное оформление заголовков нас не устраивает — его легко изменить с помощью стилей. Использовать теги заголовков по прямому назначению можно и нужно, кроме удобного структурирования информации, это обеспечивает лучшую "видимость" сайта в поисковых системах.

Все теги заголовков должны быть закрыты. Браузеры всегда выводят заголовки с новой строки. Не следует делать текст заголовков слишком длинным. Стандарт допускает в заголовках атрибуты `align` и `title`. Приведенный в листинге 5.3 HTML-код иллюстрирует создание заголовков и разделов. Результат выполнения этого кода представлен на рис. 5.3. Вы можете выполнить код, открыв на компакт-диске с примерами документ Glava_5\Headers.html.

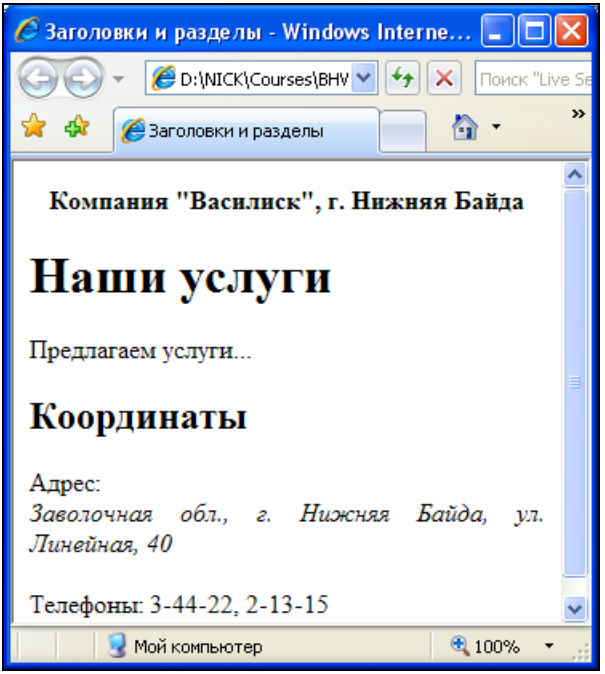

Рис. 5.3. Использование заголовков и разделов

Листинг 5.3. Использование заголовков и разделов

```
<html>
 <head><title>Заголовки и разделы</title></head>
<body>
<div align=center><strong>Компания "Василиск", г. Нижняя
Байда</strong></div>
<div align=justify>
<h1>Наши услуги</h1>
<p>Предлагаем услуги...
<h2>Координаты</h2>
<p>Адрес: <address>Заволочная обл., г. Нижняя Байда, ул. Линейная,
40</address>
<br>Телефоны: <span class="phone">3-44-22</span>, <span class="phone">2-
13-15</span>
</div></body></html>
```

Обратите внимание, что атрибут `class="phone"`, примененный дважды в теге `<span>`, помечающем номера телефонов, не вызовет на экране абсолютно

никакого эффекта. Зато создание стиля для этого класса позволит впоследствии эффективно управлять отображением *всех номеров телефонов сразу*. Формат HTML сделан "с запасом", и наличие тега `<span>` с атрибутом `class` для классификации отдельных элементов текста — полезный прием грамотной работы, который может пригодиться позже.

Тег `<address>…</address>`, примененный в листинге, должен содержать *контактную информацию* о данном документе. Обычно он размещается в начале или конце документа и содержит имена авторов, адреса их электронной почты и т. п. В строгом HTML-документе он может содержать только текстовые элементы. Как правило, браузеры выполняют перевод строки перед выводом содержимого тега `<address>` и выводят его содержимое курсивом.

5.7. Горизонтальные линии

Тег `<hr>` (horizontal rule) позволяет вывести на экран *горизонтальную черту*, которая может служить разделителем отдельных частей документа.

Тег имеет целый ряд атрибутов, управляющих видом разделителя (табл. 5.6).

Таблица 5.6. *Атрибуты тега <hr>*

Атрибут	Описание
`size`	Указание толщины линии в *пикселах* (экранных точках)
`width`	Указание длины линии в виде `width="600"` (пикселы) или `width="90%"` (проценты от доступной в текущем блоке ширины окна). Значение по умолчанию равно `100%`
`align`	Выравнивание относительно внешнего содержимого (только значения `left`, `right` или `center`)
`color`	Указание цвета именем (`color="red"`) или кодом (`color="#ff0000"`). Подробнее см. *разд. 9.1*
`noshade`	При указанном атрибуте линия создается без трехмерных эффектов

Линия, созданная тегом

```
<hr width="90%" size="3" color="green" noshade>,
```

будет представлять собой плоскую горизонтальную черту зеленого цвета шириной 3 пиксела, занимающую 90% доступной ей ширины окна.

Применение горизонтальной линии предусматривает также перенос строки.

Кроме тега `<hr>`, разработчики часто используют всевозможные графические разделители. О них мы поговорим в *главе 10*.

Глава 6

Гиперссылки — основа Web

Главное, что делает Web-страницы *гипертекстом* — наличие на них *гиперссылок* (или просто ссылок). В этой главе мы научимся связывать с помощью ссылок Web-страницы и другие виды ресурсов, пользоваться закладками для быстрого перехода в пределах документа, а также обсудим способы применения ссылок для решения основных связанных с ними задач — организации меню сайта и коллекции ссылок.

6.1. Как устроена ссылка

Гиперссылками называют выделенные области документа, позволяющие перейти к другой его части или к другому документу в Сети.

Ссылка состоит из "внутренней" части, т. е. URL-адреса документа, на который она указывает, и "внешней", видимой на экране и называемой *якорем* ссылки (anchor). Якорь может представлять собой слово или группу слов, картинку или ее часть. Если подвести указатель мыши к ссылке, он примет форму ладони с вытянутым указательным пальцем — и это самый надежный способ ее определить. При этом в строке состояния браузера обычно можно прочитать URL-адрес, на который указывает ссылка.

Текстовые ссылки практически всегда выделены другим цветом и очень часто подчеркнуты. Графические ссылки часто заключены в рамку того же цвета, которым выделяются текстовые ссылки. По умолчанию это синий цвет, но каждая Web-страница может использовать свое собственное оформление, заданное в атрибутах тега `<body>` или с помощью стиля. Ссылки, которые уже выбирались пользователем, обычно отличаются по цвету от непосещенных ссылок и выглядят более "бледными" (по умолчанию для них используется фиолетовый цвет).

Для создания гиперссылки служит тег вида

```
<a href="адрес" target="окно">…</a>,
```

содержимое которого — текстовый или графический якорь. Атрибут `href` обязателен, `target` может быть опущен, тогда документ открывается в текущем окне браузера, или указан в виде `target="_blank"`, тогда документ открывается в новом окне. Текст ссылки внутри тега `<a>` может быть любым. Точно так же в тег `<a>` можно поместить и изображение (см. *главу 10*).

Например, создание в документе ссылки вида

```
Нажав <a href="http://www.mysite.com/page1.html">сюда </a>, Вы перейдете
на страницу 1
```

приведет к тому, что при щелчке мышью по слову "сюда" в текущее окно браузера начнет загружаться страница с URL-адресом **http://www.mysite.com/page1.html**.

Средствами HTML нельзя проверить корректность ссылок, ответственность за нее целиком лежит на Web-мастере. Браузер прореагирует на неправильную ссылку лишь тем, что после безуспешной попытки открыть ее выдаст сообщение "Страница не найдена". Возможны и другие ошибки — например, URL-адрес существует, но вы не имеете прав доступа к нему. Будем исходить из оптимистичного предположения, что все адреса, на которые мы желаем сослаться, нам известны, открыты для доступа и мы сможем их записать без ошибок:

```
Зайди на <a href="http://www.reclama.ru" target="_blank">www.disney.com</a>!
```

Вот это да! Увидев в якоре ссылки многообещающий адрес "`www.disney.com`", мы, не задумываясь, щелкнули по нему мышью, а в открывшемся новом окне загрузился сайт рекламного агенства! Усвоим урок — якорем ссылки может быть что угодно, и надежных способов узнать, куда ведет ссылка, *на самом деле*, только два:

- прочитать URL ссылки в нижней части окна браузера, а именно в строке состояния его окна (если строка состояния выключена, обычно ее можно включить из меню **Вид** браузера);
- щелкнуть правой кнопкой мыши на ссылке, в появившемся меню найти команду **Копировать ярлык** (или **Копировать ссылку**, **Copy link location**), а затем в удобной для вас программе (подойдет и Блокнот) выполнить команду **Вставить** меню **Правка** или просто нажать комбинацию клавиш <Ctrl>+<V>.

Оба способа могут быть "отменены" хитрыми Web-мастерами. Например, с помощью события `onmouseover` легко заставить браузер показывать в строке состояния не URL активной ссылки, а любую чушь. Но мы таких подозрительных ссылок посещать и не будем.

Браузер никак не предупредит вас о том, что ссылка собирается открывать новое окно. Максимум, на что он способен — блокировать появление большинства надоедливых всплывающих окон с рекламой.

Ссылка не обязательно должна указывать на другую Web-страницу с типом файла htm или html. Любой файл, доступный из Сети и имеющий URL, может быть адресован с помощью ссылки. Разумеется, браузер откроет в своем окне только те типы документов, что ему известны. В отношении остальных документов, скорее всего, будет выдан запрос о том, что вы желаете сделать — сохранить адресуемый файл на жесткий диск, найти программу для открытия этого типа файлов или все же попытаться открыть его средствами браузера. Таким образом, браузер не обязан уметь работать с архивами программ WinZip или WinRAR — вполне достаточно, если пользователь сможет "скачать" эти архивы к себе на компьютер. Вот пример:

```
<a href="http://www.mysite.org/archives/video.zip">Архив видео</a>.
```

По нажатию на ссылку с текстовым якорем "Архив видео", браузер предложит сохранить на компьютере пользователя файл-архив с именем video.zip, находящийся в папке archives на сервере **www.mysite.org**. Узнать подробнее о ссылках на различные виды ресурсов вы сможете из *разд. 6.4*.

Помимо базовых событий, описанных в *табл. 4.1*, элемент `<a>` допускает обработку еще двух:

- `onfocus` — происходит, когда гиперссылка получает фокус;
- `onblur` — происходит, когда гиперссылка теряет фокус.

6.2. Абсолютные и относительные ссылки

Ссылка, в которой указан полный URL документа, называется *абсолютной*.

Все приведенные ранее ссылки абсолютные, это означает, в частности, что при удалении с сервера адресуемого документа или изменении его адреса все они перестанут работать. Разумеется, это крайне неудобно, поэтому абсолютные ссылки используются, в основном, для связи с внешними ресурсами Интернета, URL которых известен и не меняется. В листинге 6.1 абсолютные ссылки служат для адресации популярных поисковых систем русскоязычного сегмента Сети. В папке Glava_6 на компакт-диске этот пример записан под именем Abslinks.html.

Листинг 6.1. Использование абсолютных ссылок

```
<html>
<head>
 <title>Использование абсолютных ссылок</title>
```

```
</head>
<body>
<p>В <acronym title="Русскоязычный сегмент Интернета">ВРунете</acronym>
наиболее популярны следующие поисковые машины:
<br><a href="http://www.yandex.ru">Яndex</a>
<br><a href="http://www.rambler.ru">Rambler</a>
<br><a href="http://www.google.ru">Google</a>
<br><a href="http://www.aport.ru">Апорт</a>
</body></html>
```

К каждой из ссылок мы могли бы добавить атрибут `target="_blank"` для открытия ее в новом окне. Что делать, когда ссылок на странице много и указание атрибута для каждой из них существенно увеличит размер кода? Помочь может упомянутый в *главе 4* тег `<base>` с атрибутом `target="_blank"`. Добавив в теге `<head>` конструкцию вида

```
<base target="_blank">,
```

мы сделаем все ссылки открываемыми в новом окне. В этом случае открыть нужную ссылку (например, навигационную) в текущем окне можно тегом вида

```
<a href="URL-адрес" target="_self">якорь ссылки</a>.
```

Следует учесть, что обилие открывающихся окон может сбить посетителя с толку.

Так как визуально ссылки на текущее и новое окно ничем не отличаются, на некоторых сайтах рядом со ссылкой ставят специальную иконку или подпись, показывающую, что документ будет открыт в новом окне.

Относительная ссылка адресует документ, опуская общую адресную часть. Например, если из документа index.html нужно сослаться на документ test.html, находящийся в той же папке, это можно сделать ссылкой вида

```
<a href="test.html">документ test.html</a>.
```

Относительная ссылка позволяет ссылаться на папки, которые являются как вложенными, так и родительскими по отношению к папке расположения исходного документа. Например, ссылка на рисунок с именем my.jpg, находящийся во вложенной папке images, может иметь вид

```
<a href="images/my.jpg">посмотрите рисунок</a>.
```

Иногда разработчики применяют адресацию, начинающуюся с головной папки сервера. В листинге 6.2 показан документ, адресующий другой документ с именем 2.html, расположенный в папке `2`, вложенной в корневую папку сервера. Обратите внимание, что путь к корневой папке (в данном случае,

URL-адрес локального сервера) прописан в теге `<base>` документа. Если вы не уверены в том, на каком именно сервере будет работать ваш html-код, не пользуйтесь этим способом адресации!

Листинг 6.2. Адресация от головной папки сервера

```
<html>
<head>
<title>Документ 1</title>
  <base href="http://127.0.0.1">
</head>
<body>
  <a href="/2/2.html">Ссылка на документ 2</a>
</body>
</html>
```

Есть способы лучше. Но сначала давайте поймем, в чем суть вопроса.

Создавая сайт, вы очень быстро столкнетесь с проблемой управления большим количеством страниц, картинок, стилевых файлов и прочего содержимого. Худшее из возможных решений — размещать все в одной папке. Работая в Windows, вы уже поняли, зачем нужны папки и их вложение. С Web-сайтом то же самое. С ним проще обойтись, если все картинки общего назначения хранятся в папке images (кстати, это и браузеру поможет их загрузить), а документы — в папке docs. Еще лучше разделить абстрактные "документы" по назначению и дать папкам осмысленные имена, например, services — для файлов с описаниями услуг вашей фирмы и prices — для архивов с прайс-листами. Теперь представим, что из документа в папке services, описывающего конкретную услугу, нужно сослаться на прайс-лист, хранящийся в папке prices. Это легко сделать, если указать *относительный путь*:

```
<a href="../prices/price1.html">прайс-лист на услугу</a>.
```

Две точки (`..`) в URL означают то же самое, что во многих программах Windows — "подняться на один уровень выше по папкам".

Получается, что все очень просто:

- по умолчанию все пути отсчитываются от папки, где расположен текущий документ, поэтому для ссылки на документ из той же папки достаточно указать в атрибуте `href` только его имя;
- символы `..` в начале URL означают переход на один уровень вверх от текущего документа;

- если URL сразу же начинается с имени папки, это означает, что папка вложена в текущую.

Иногда в URL ссылки можно встретить путь в виде ./file/doc.html. Точка со слэшем (./) также означают, что отсчет ведется от текущей папки. Подобная запись избыточна, и ее можно сократить до file/doc.html.

Посмотреть пример "структурированного" сайта в работе вы можете, открыв файл index.html из папки Glava_6/Site1 компакт-диска. Внимательно изучите связи между файлами этого сайта. Вы можете даже скопировать куда-нибудь папку Site1 — все связи между страницами все равно будут работать!

Именно поэтому при разработке собственного Web-сайта для перехода между его страницами используются относительные ссылки, что позволяет просматривать сайт на локальной машине, не внося в него изменений, а также при необходимости легко переместить страницы на другой сервер.

Не рекомендуется делать структуру папок очень "глубокой", если не хотите, чтобы поисковые серверы узнали о ваших документах только к концу века.

6.3. Внешние и внутренние ссылки

Одно из преимуществ электронного документа в сравнении с обычным — возможность быстро перейти к нужному месту или найти его по ключевому слову.

Внутренние ссылки предназначены для навигации в пределах документа. Они имеют вид `<a href="#закладка">якорь ссылки</a>`.

Для *создания закладки* служит тег следующего вида:

```
<a name="закладка">якорь закладки</a>.
```

Имена закладок должны быть уникальными в пределах документа. Якорь закладки может быть пустым.

Например, если первое слово документа заключить в тег `<a name="start">...</a>`, то в нижней его части можно разместить ссылку вида

```
<a href="#start">к началу страницы...</a>,
```

щелчок по которой позволит быстро перейти к началу документа.

Внутренние ссылки удобны и для организации оглавления документа. В листинге 6.3 показан пример документа, представляющего собой список вопросов по HTML с ответами на них, причем в верхней части странички находится меню. Большинство примеров можно открыть в новом окне браузера для выполнения, после прочтения примера можно вернуться к началу документа, используя ссылку

```
<a href="#start">вернуться в начало</a>.
```

Для выполнения примера с компакт-диска откройте файл index.html из папки Glava_6\Content. В листинге 6.3 приведен только фрагмент документа index.html.

Листинг 6.3. Использование внутренних ссылок в документе

```
<html><head>
 <title>Использование внутренних ссылок</title>
</head>
<body>
<p><a name="start">Выберите</a> вопрос для перехода к ответу:
<br><a href="#alternate">Как сделать строку текста, плавающую по экра-
ну?</a>
<br><a href="#title">Зачем нужен атрибут title?</a></p>

<hr><a name="alternate"></a>
<p>Строка, плавающая влево-вправо (работала и в IE3!)
<br><tt>&lt;MARQUEE BEHAVIOR='ALTERNATE'&gt;ТЕКСТ&lt;/MARQUEE&gt;</tt>
<br><a href="alternate.html" target="_blank">посмотреть пример в
работе...</a>
<a href="#start">вернуться в начало</a>
</p>

<hr><a name="title"></a>
<p>Это атрибут, применимый почти в любом теге.
<p>Он создает всплывающую подсказку - подобно атрибуту ALT в теге IMG.
<p>Вставьте следующий HTML-код на свою страницу  в тег A или другой:
<br><tt>TITLE='подсказка'</tt>
<br><a href="title.html" target="_blank">посмотреть пример в рабо-
те...</a>
<a href="#start">вернуться в начало</a>
</p></body></html>
```

Обратите внимание: для того, чтобы символы "<" и ">" показывались в тексте страницы, нам пришлось заменить их специальными символами `<` и `>`. Следует также помнить, что элемент `<a>` не может быть вложен в другой элемент `<a>`, поэтому якоря закладок не должны включать в себя гиперссылок.

Все ссылки, не относящиеся к внутренним, являются внешними. Внешняя ссылка также может содержать адрес закладки в документе. Например, для

перехода из текущего документа к закладке с именем `q1` документа `work.html`, находящегося в этой же папке, достаточно записать тег вида

`<a href="work.html#q1">перейти…</a>`.

При работе в Microsoft Word внутренним ссылкам соответствует пункт **Закладка** меню **Вставка**, внешним — инструмент **Гиперссылка…** этого же меню.

6.4. Ссылки на различные виды ресурсов

В *разд. 6.1* мы уже упоминали о том, что ссылаться можно не только на Web-страницы, но и на другие типы файлов. Браузер в этом случае может предложить пользователю сохранить файл на жесткий диск, либо указать программу для открытия этого типа файлов, либо попытается открыть файл самостоятельно как текстовый. Но многообразие ссылок не исчерпывается тем, что адресовать можно любые файлы, имеющие URL. Адресовать можно и *различные виды ресурсов* Интернета, вспомнить о которых вы можете, обратившись к *разд. 1.3*. Повторим главное: Интернет — не только огромная коллекция Web-страниц, но и объединение ряда *служб*, решающих различные задачи электронного документооборота и, с точки зрения пользователя, отличающихся между собой используемым *протоколом прикладного уровня*. Поскольку протокол указывается в URL гиперссылки, легко сослаться на адрес электронной почты, файловый архив FTP или локальный файл с вашего жесткого диска. В табл. 6.1 приведены ссылки на самые распространенные виды ресурсов.

Таблица 6.1. *Ссылки на различные виды ресурсов*

Протокол ссылки	**Описание**
`http://`	HTTP — основной протокол, обеспечивающий доступ к Web-страницам
`ftp://`	Протокол передачи файлов FTP, позволяющий при помощи программы FTP-клиента обмениваться файлами с удаленным компьютером
`mailto:`	Доступ к электронной почте. Вслед за `mailto:` указывается адрес электронной почты, имеющий общий вид `имя@домен`
`file://`	Указывается в качестве протокола при обращении к Web-странице, расположенной на локальной машине

Приведем примеры соответствующих ссылок.

```
<a href="ftp://ftp.narod.ru" target="_blank">вход в файловый архив</a>
```

Здесь щелчок по якорю "вход в файловый архив " открывает в новом окне содержимое папок, расположенных по URL-адресу **ftp.narod.ru**. Как правило, браузер обеспечивает элементарные средства навигации по папкам. Чаще всего для доступа к файлам следует сначала ввести логин и пароль. Можно учесть их и в коде ссылки. Если вы зарегистрированы на сервере narod.ru как пользователь igor, имеющий пароль 13igr1T7, свои данные можно указать следующим образом:

```
<a href="ftp://igor:13igr1T7@ftp.narod.ru">моя страничка</a>.
```

Разумеется, обнародовать свой пароль вы не станете и лучше всего ограничиться ссылкой

```
<a href="ftp://igor@ftp.narod.ru">моя страничка</a>.
```

Протокол FTP используется как для обновления файлов на сервере с помощью программы *FTP-клиента*, так и для доступа к сетевым коллекциям файлов.

На многих FTP-серверах разрешен и анонимный вход пользователя, а общедоступные документы располагаются в папке с именем `pub`:

```
<a href="ftp://ftp.nsk.ru/pub" target="_blank">коллекция файлов</a>.
```

Очень часто на Web-страницах указываются адреса электронной почты:

```
<a href="mailto:gates@microsoft.com">отправь письмо Биллу Гейтсу!</a>.
```

Щелчок по этой ссылке приведет к запуску окна создания сообщения, которое можно отправить по указанному в ссылке адресу **gates@microsoft.com**. Если почтовая программа еще не настроена, придется сначала настроить ее. Если ни одна почтовая программа не установлена, браузер сообщит об этом в окне диалога. Для ссылки на адрес E-mail не имеет смысла указание атрибута `target`. Визуально ссылка на E-mail ничем не отличается от обычной ссылки на Web-страницу, поэтому ее якорем чаще всего делают сам адрес электронной почты:

```
<a href="mailto:pers@mail.ru">pers@mail.ru</a>.
```

Открыв с компакт-диска пример Glava_6\Content\about.html, вы можете посмотреть в коде страницы, как указать в атрибутах ссылки *тему* и *текст* письма.

Наконец, протокол file:// может оказаться полезным при организации ссылок на локальном компьютере. Например, щелчок по ссылке

```
<a href="file://c:/" target="_blank">оглавление диска </a>
```

открывает в новом окне содержимое диска c:. Если нужно адресовать таким способом папку или документ на диске компьютера, в записи пути следует указать символ "/", а не "\", как принято в Microsoft Windows.

Посмотреть эти и другие примеры в работе вы можете, запустив документ Glava_6\Resources.html с компакт-диска.

6.5. Организация навигации по сайту

Итак, с механизмом работы ссылок все более-менее ясно, остается вопрос, как именно эти ссылки организовать. Во-первых, нам не обойтись без ссылок для организации *навигации* по сайту, т. е. переходов между его отдельными страницами. Во-вторых, каждый Web-мастер жаждет поделиться с миром ссылками, которые кажутся ему интересными (или способными принести прибыль). Именно эти два основных аспекта использования ссылок мы рассмотрим.

Определиться с навигацией — одно из важнейших решений, которые вам предстоит принять в самом начале работы над сайтом. От этого зависит, во-первых, удобство расширения сайта и работы над ним, а, во-вторых, навигация — один из ключевых факторов оценки вашего сайта пользователями. Для удобной и эффективной навигации не обязательно подключать сложные скрипты и апплеты, можно обойтись, в основном, средствами HTML. Изучим основные виды навигации подробнее.

Для несложной странички вполне подойдет *одноуровневая* навигация. Все документы при этом расположены в одной (обычно главной) папке сайта и связаны относительными ссылками, организованными в горизонтальный или вертикальный блок меню:

```
<center>
<a href="index.html">главная</a>
 &diams; <a href="biography.html">биография</a>
 &diams; <a href="creative.html">творения</a>
 &diams; <a href="contacts.html">контакты</a>
</center>
```

Специальный символ `♦` между ссылками горизонтального меню добавлен здесь просто "для красоты". Чтобы пункты меню визуально не сливались, он обрамлен неразрывными пробелами ` `. Организовать подобным же образом вертикальное меню вы сможете, изучив в *главе 8* работу с таблицами.

Данный способ навигации реализован в примере из папки Glava_6\Navigation1 на компакт-диске. Из каждого файла исключена ссылка на текущую страницу, так что один пункт меню всегда выводится как обычный текст.

Основное неудобство способа очевидно — при добавлении или удалении странички придется изменять меню во всех файлах. Решить проблему может подключение внешнего меню директивой SSI (о них упомянуто в *разд. 2.2*). К сожалению, директивы SSI работают далеко не на каждом хостинге. Суще-

ствует несложное альтернативное решение — организация блока ссылок с помощью скрипта на языке JavaScript. В *разд. 15.4* приведен пример такой навигации, учитывающий возможность того, что JavaScript у пользователя может быть отключен.

Линейная или *последовательная* модель сайта тоже достаточно проста. При такой навигации происходит последовательный переход от страницы к странице. Это характерно для просмотра серии фотографий, частей длинного текста или набора однотипных документов.

Переход не обязан быть строго последовательным, часто на сайте предлагается набор страниц, к которым можно перейти непосредственно. Например, поисковые системы показывают первую страницу результатов поиска и дают возможность перейти сразу ко второй, третьей или самой последней странице.

Несмотря на простоту и наглядность, линейная навигация имеет ряд недостатков:

- термин "следующая страница" или подобный ему дает лишь представление о том, что происходит переход на другую страницу, но ничего не говорит о ее содержании;
- если на сайте просто показан список с номерами страниц, невозможно определить, на какой из них находятся нужные нам материалы. Например, поисковые системы сортируют результаты поиска по релевантности (важности) таким образом, что в начале списка выдачи оказываются ссылки на страницы, "наиболее удовлетворяющие" условиям поиска, а страницы, показавшие результат, далекий от желаемого, выводятся вслед за ними. Но это может оказаться совсем не так, и пользователю приходится просматривать множество страниц, чтобы понять, содержат ли они нужную информацию;
- список из большого количества страниц занимает много места и неудобен для просмотра. В этом случае большую часть вариантов сокращают, заменяя их многоточием. Это приводит к тому, что для перехода к нужной странице, даже если известен ее номер, нужно посетить ряд дополнительных страниц.

Несмотря на описанные минусы, линейная навигация пользуется большим успехом на сайтах, где требуется отобразить список однотипных страниц или последовательно переходить от одного документа к другому. Часто она дополняет другие виды навигации на сайте. Трудоемкость "ручной" организации линейной навигации нисколько не меньше, чем у простой одноуровневой.

Линейная модель подходит не только для вывода по частям длинных произведений, но и для регулярно обновляемых сайтов с быстро устаревающим

содержимым, впрочем, такие сайты обычно используют готовый "движок" на языке PHP.

Пример линейной навигации представлен на компакт-диске в папке Glava_6\Navigation2. Приведем фрагмент кода, содержащий меню, из файла с именем 2.html:

```
<center>
[<a href="index.html">начало</a>]
[<a href="1.html">назад</a>]
[<a href="3.html">вперед</a>]
[<a href="end.html">заключение</a>]
</center>
```

Иерархическая модель организации сайта разбивает документы на уровни, связанные между собой древовидной структурой. Попросту говоря, каждый документ может иметь только одного "предка" и произвольное количество "потомков". Например, документ "Мои программы" может ссылаться на сколько угодно написанных вами программ (только список их станет все длиннее), а добавление новой программы потребует всего *одной* новой ссылки на сайте. Равно как и документ с описанием этой новой программы будет иметь небольшое количество внешних ссылок — только на своих предков вышестоящих уровней. Такой способ организации сайта действительно похож на генеалогическое древо, в том плане, что каждый уважающий себя документ должен знать своих прямых предков и их предков тоже, желательно, вплоть до "Адама", представленного главной страницей сайта.

Моя домашняя страничка организована именно так. Например, по ссылке

```
<a href="../../index.html ">Главная страница</a>. <a
href="../index.html">Мои программы</a>. <a
href="index.html">Delphi</a>. Программа Tester
```

сразу же видно, что она представляет собой описание программы с именем Tester, написанной на языке Delphi. Ссылка с якорем "Delphi" адресует документ index.html из текущей папки. В нем собраны все ссылки на программы, созданные на этом языке. Ссылка "Мои программы" поднимает нас на уровень выше — к документу, содержащему список языков программирования и ссылки на страничку каждого из них. Ссылка "Главная страница" позволяет перейти в начало сайта, находящееся двумя уровнями выше. Кстати, почему все документы называются index.html? Ответ на этот вопрос уже был дан в *разд. 1.4*. Имя index.html — самое распространенное название "документа по умолчанию", так что его использование для главного документа в *каждой* папке позволит пользователю указывать адрес вида **www.domain.com/programs** вместо **www.domain.com/programs/index.html**.

Обратите внимание на указание относительных ссылок в адресации вышестоящих документов и "жесткого пробела" ` ` между отдельными словами якоря ссылки. Это сделано на тот случай, если браузер "вздумает" разорвать строку именно в этом месте. Сделав в каждой папке сайта шаблон, содержащий ссылки на папки верхних уровней и общие элементы оформления, вы сможете легко добавлять в любой раздел новые страницы.

Развернутый пример иерархической навигации представлен в папке Glava_6\Navigation3 компакт-диска.

Очевиден основной плюс иерархического представления — оно однозначно адресует каждый документ. Так, в иерархии папок Windows каждая из них имеет уникальный адрес. Кроме того, иерархию *легко* адресовать, чем и воспользовались разработчики операционных систем.

При организации иерархической навигации желательно соблюдать следующие требования:

- в качестве разделителей, если они есть, следует использовать символы, способные показать вложенность пунктов (например, ">", "/");
- предельно допустимое число уровней вложения — три-четрые, иначе индексация сайта поисковыми машинами будет затруднена;
- начинать адресацию всегда следует с главной страницы;
- блок навигации лучше размещать в верхней части страницы, возможно, есть смысл дублировать его и внизу;
- последний пункт должен описывать текущую страницу, при этом он не является ссылкой.

В чем неудобство? Вы достаточно легко сможете добраться до любой категории, а вот поиск в ней конкретного документа (особенно, если категория разрослась) может отнять время. Проблема обычно решается тем, что на сайте существует специальная *страница новостей*, где фиксируются все обновления с прямыми ссылками на обновленные документы, а также отдельно созданная *карта сайта*, содержащая список всех разделов и документов на одной Web-странице. В папке Glava_6\Navigation3 представлены оба этих важных элемента сайта (документы news.html и map.html соответственно).

В *сетевой* модели сайта возможно существование *любых* взаимосвязей между документами. Очевиден основной плюс такого представления — теоретически можно реализовать сколь угодно сложные и разветвленные связи. С этим же связан и основной минус — реализация может оказаться весьма трудоемкой, а навигация — крайне запутанной. Сам www однозначно реали-

зует эту сетевую модель — оттого и пользоваться им непросто. Для несложного сайта сетевая модель представляется избыточной, если только вы не делаете "Adventure page" для любителей всего необычного. Впрочем, такие любители перевелись еще в конце прошлого века — современный пользователь требует от сайта удобства, простоты, ясности и узнаваемости. Оттого и процветают два десятка плохо написанных "движков".

Однако существует класс сайтов, для которых такая навигация просто необходима. Это различного рода сетевые энциклопедии, словари и базы знаний. Состоящие из отдельных словарных статей, описывающих какие-либо понятия и термины, они ссылаются на другие статьи энциклопедий или внешние источники. Примером может служить открытая энциклопедия Wikipedia, русскоязычная версия которой находится по адресу **ru.wikipedia.org/wiki**.

Мы не станем приводить примера сайта с запутанной сетевой навигацией, их и так более чем достаточно в Интернете. Так что если ваша организация сайта покажется пользователям неудачной — сошлитесь на "сетевую концепцию", которой вы следуете.

Разумеется, на реальном сайте различные виды навигации могут смешиваться и дополнять друг друга. Здесь все зависит от вашего мастерства, вкуса и чувства меры.

6.6. Организация коллекций ссылок

Коллекции ссылок имеют широчайшее применение и способны решать целый круг задач:

- организация тематического каталога ресурсов Интернета;
- поддержка каталогов программ, звуковых файлов, фото- или видеоизображений;
- всевозможные перечни товаров, услуг и предложений;
- организация на сайте терминологического справочника, энциклопедии или словаря.

Основное отличие "коллекционной" ссылки от навигационной — ее *однократное* использование, так что здесь Web-мастер гораздо менее скован необходимостью дублировать ссылки или подключать сложные скрипты, чтобы уменьшить степень этого дублирования. Следует только позаботиться о том, чтобы сама страница, содержащая коллекцию, была хорошо структурирована, удобна в использовании и вовремя проиндексирована поисковой системой.

Во-первых, следует сказать о каталогах внешних ссылок, без которых не обходится почти ни один сайт. Сегодня существуют многие тысячи каталогов, содержащие миллионы ссылок. Если вы хотите создать каталог, от которого действительно будет польза, придерживайтесь следующих принципов:

- определитесь с тематикой коллекции и ее рубрикацией. Коллекции "обо всем и ни о чем" давно никому не интересны, а вот ссылки на сайты, тематически или регионально близкие вашему, могут заинтересовать посетителей, особенно если они снабжены краткими описаниями;
- вообще говоря, давая любую внешнюю ссылку, вы повышаете рейтинг кому-то и рискуете потерять посетителя у себя. Опытные мастера довольно скупы в размещении у себя чужих ссылок. Чужие ссылки должны быть интересными и важными, это повысит интерес и к вашему собственному сайту;
- по возможности, указывайте в коллекции URL ссылки — пользователи будут признательны за возможность распечатать вашу коллекцию;
- постарайтесь использовать первоисточники или более авторитетные ресурсы. У ссылки на книгу, расположенную на сервере **www.lib.ru**, больше шансов на долговечность, чем у ссылки на клон произведения, помещенный на чьей-то домашней страничке;
- якорь ссылки, особенно если это слово, должен быть тематически и логически обоснован. Ссылка "жми сюда" в большинстве случаев будет плохой, а вот фраза "узнать подробнее о моих работах и увлечениях" — почти всегда удачной;
- если ссылка ведет "на сайт", а не адресует конкретный документ, должен быть указан URL сайта, а не какой-либо его папки или файла. Странно видеть, как часто нарушается это простое правило;
- избегайте ссылок на ленты новостей, конкретные странички форумов и другое содержимое, не имеющее постоянного URL. Впрочем, на хороших лентах и форумах *статические URL*, не меняющиеся при добавлении новых записей, есть у каждого объекта. Уточнить, так ли это в отношении интересующего вас сайта, можно у его администрации или у знакомого опытного пользователя;
- не забывайте про метатеги, описанные в *разд. 4.4* и *4.5*.

Листинг 6.4 показывает начало мини-каталога, содержащего всего две рубрики: о русскоязычных поисковых машинах и коллекциях текстов на русском языке. Каждая ссылка имеет видимый в якоре URL и снабжена кратким опи-

санием. Вы можете посмотреть полный пример в браузере, открыв документ Glava_6\Links.html на компакт-диске.

Листинг 6.4. Коллекция ссылок

```
<html>
<head>
<title>Коллекция ссылок</title>
<base target="_blank">
</head>
<body>
<p><strong>Русскоязычные поисковые машины</strong>
<p><em>Здесь собраны ссылки на лучшие поисковики Рунета</em>
<p><a href="http://www.yandex.ru">www.yandex.ru - Яndex</a>.
Это самая мощная поисковая машина в Рунете. Яndex существует с 1997 года
и занимает лидирующие позиции на российском рынке. Он объединяет
в себе поисковик, тематический каталог, электронную почту и множество
других служб. Существует "краткая версия" поискового сервиса
по адресу <a href="http://www.ya.ru">www.ya.ru</a>.
<p><a href="http://www.rambler.ru">www.rambler.ru - Rambler</a>.
Очень популярный поисковик, в нем есть также много дополнительных служб.
Rambler нравится мне тем, что часто проводит различные конкурсы и акции.
. . .
</body></html>
```

Что касается остальных применений коллекций ссылок, то организацию коллекций графики и фотоальбомов мы обсудим в *разд. 10.6*, а о справочниках и словарях следует сказать несколько слов сейчас. Организация тематического словаря или справочника на сайте — его преимущество, благодарно принимаемое и пользователями, и поисковыми системами. В простейшем случае ссылка, якорем которой является название статьи или термин, может вести к закладке документа или даже к текстовому файлу, открываемому в новом окне:

```
<br><a href="#tag">тег</a>
. . .
<p><a name="tag">тег</w> - элемент разметки или оформления в HTML.
Общий вид тега следующий: …
```

Следует позаботиться о рубрикации справочника, для большого по объему проекта желательно создать отдельный файл или блок ссылок для перехода к началу рубрик.

Справочник может быть и алфавитным, например, создав в файле-шаблоне перечень ссылок

```
<a href="a.html">A</a> <a href="b.html">B</a> ... <a href="z.html">Z</a>,
```

вы можете сохранить его под именами a.html, b.html, …, z.html и затем наполнять полученные 26 файлов информацией о терминах.

Для управления внешним видом и нумерацией в любых перечнях и списках идеально подходят теги работы со списками, которые мы изучим в *главе 7*.

Глава 7

Списки

Любой развитый формат работы с текстом должен включать в себя средства для организации списков. В этой главе мы научимся управлять с помощью HTML списками любой сложности.

7.1. Списки — инструмент структурирования информации

О распространенности списков вы можете судить по тексту этой или любой другой технической книги. Да и в обычной жизни нам постоянно приходится иметь с ними дело — это может быть список студенческой группы или список покупок, которые мы хотим совершить, ведомость начисления зарплаты или обычная телепрограмма, перечень файлов в открытой на вашем компьютере папке или библиографический каталог. Служебная инструкция и банковский договор, описание повестки дня заседания или порядка сборки деталей также вполне естественно выглядят в виде списка. Наконец, всевозможные перечни, каталоги, перечисления, оглавления и словари также являются списками.

Почему списки так популярны и их следует использовать? Ответить на этот вопрос также удобнее всего в форме списка:

- списки позволяют лучше структурировать информацию, разбивая текст на небольшие блоки, удобные для восприятия человеком;
- списки идеально подходят для описания любых пошаговых и последовательных процессов;
- списки удобны для краткого аннотирования информации, в этом случае пункты списка являются ссылками на более полные описания объектов.

Возникает вопрос — нужны ли для описания списков специальные теги? До сих пор мы вполне обходились абзацами и переводами строки, а визуальное оформление списка можно создать простым указанием номера или маркера:

```
<p>1. Элемент списка 1
<p>2. Элемент списка 2
```

или

```
<br>&bull; элемент 1
<br>&bull; элемент 2
```

В последнем случае использован специальный символ "маркер списка".

На самом деле теги оформления списков имеют несколько преимуществ:

- нумерация или маркировка элементов выполняются автоматически, вместе с соответствующим оформлением;
- порядок и оформление не нарушаются при добавлении и удалении элементов списка;
- для изменения способа нумерации или маркировки достаточно изменить атрибуты тега списка;
- теги позволяют легко создавать сложные многоуровневые списки.

7.2. Нумерованные, маркированные и многоуровневые списки

В этом разделе мы изучим два основных имеющихся в HTML типа списков.

Маркированные (неупорядоченные) списки используют для выделения своих элементов специальные символы, называемые *маркерами*. Вид маркеров определяется браузером, причем при создании вложенных списков браузер автоматически назначает разные маркеры элементам разного уровня вложенности. Как правило, кроме маркеров, элементы списков выделяются визуальным сдвигом вправо.

Маркированный список создается контейнерным тегом `<ul>…</ul>` (от англ. "unordered list" — несортированный список), внутри которого помещаются теги отдельных элементов списка. Тег `<ul>` является блочным и обеспечивает перевод строки до и после списка, таким образом, абзац или `<br>` для отделения списка от текста не нужны. Тег `<ul>` обязательно должен быть закрыт.

Отдельные элементы списка, вложенные в тег `<ul>`, маркируются текстовым элементом `<li>` (от англ. "list item" — элемент списка), который можно не закрывать. Как правило, браузер автоматически переводит строку перед каждым новым элементом списка. Тег `<li>` может содержать любые другие

текстовые элементы или гиперссылки. Листинг 7.1 показывает пример создания маркированного списка, внешний вид которого иллюстрирует рис. 7.1. Пример записан на компакт-диске под именем Glava_7\Unsorted.html.

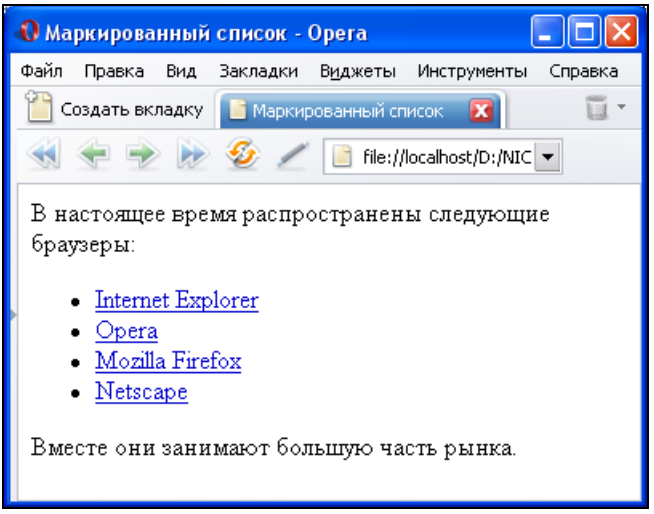

Рис. 7.1. Маркированный список

Листинг 7.1. Маркированный список

```
<html>
<head>
 <title>Маркированный список</title>
</head>
<body>
<p>В настоящее время распространены следующие браузеры:
<ul>
<li><a href="http://www.microsoft.com/ie" target="_blank">Internet Ex-
plorer</a>
<li><a href="http://www.opera.com/download/" target="_blank">Opera</a>
<li><a href="http://www.getfirefox.net/" target="_blank">Mozilla Fire-
fox</a>
<li><a href="http://browser.netscape.com/" target="_blank">Netscape</a>
</ul>
<p>Вместе они занимают большую часть рынка.</p>
</body></html>
```

Здесь элементами списка являются ссылки на "домашние страницы" программ-браузеров.

В теге `<ul>` могут быть заданы два атрибута. Атрибут `compact`, не имеющий значений, указывает браузеру, что список должен быть выведен в более компактном виде (например, может быть уменьшен шрифт или расстояние между пунктами). В настоящее время все популярные браузеры игнорируют этот атрибут, спецификацией он также не рекомендуется.

Атрибут `type` может быть применен как к тегу списка, так и к тегу элемента `<li>`. Он позволяет определить желаемый вид маркера. Возможные значения атрибута — `disc` (по умолчанию), `circle` и `square`. Конкретный внешний вид маркеров зависит от браузера. Браузеры *должны* пытаться представлять "`disc`" в виде небольшого заполненного кружка, "`circle`" — полой окружности, а "`square`" — маленького квадрата.

Современные стандарты рекомендуют использовать вместо атрибута `type` указание в стиле тегов `<ul>` и `<ol>` свойства `list-style-type`, принимающего значения `disc`, `square` и `circle`.

Пример маркированного списка с атрибутом `type` приведен в листинге 7.2, а проверить его в своем браузере вы можете, открыв документ Glava_7\UnsortedType.html с компакт-диска.

Листинг 7.2. Маркированный список с атрибутами `type`

```
<html>
<head>
 <title>Маркированный список с атрибутами type</title>
</head>
<body>
<p>Времена года по-английски:
<ul type="square">
<li>весна - spring;
<li type="circle">лето - summer;
<li>осень - autumn;
<li type="disc">зима - winter.
</ul>
</body></html>
```

Нумерованные (упорядоченные) списки также представляют собой последовательность элементов, создаваемых тегом `<li>`. Основное отличие от маркированных списков — порядковый номер, который автоматически простав-

ляется перед каждым элементом. Вид номеров зависит от браузера, порядок нумерации может настраиваться с помощью атрибутов, в остальном реализации списков очень похожи.

Элементы нумерованного списка помещаются внутрь контейнерного тега `<ol>…</ol>` (от англ. "ordered list" — упорядоченный список). В открывающей части тега могут быть указаны атрибуты `compact`, `type` и `start`.

Устаревший атрибут `compact` имеет тот же смысл, что для тега `<ul>`. Атрибут `type` определяет способ нумерации элементов списка (табл. 7.1).

Таблица 7.1. *Значения атрибута type тега *

Значение	Вид нумерации
`type="1"`	Арабскими цифрами (значение по умолчанию)
`type="I"`	Большими римскими цифрами I, V, X, …
`type="i"`	Малыми римскими цифрами i, v, x, …
`type="A"`	Прописными латинскими буквами A, B, C,…
`type="a"`	Строчными латинскими буквами a, b, c,…

Атрибут `type` можно указывать и внутри тега `<li>`, переопределяя тем самым стиль вывода конкретного номера в списке.

Атрибут `start` позволяет начать нумерацию элементов списка не с единицы. При этом значением параметра может быть только арабская цифра. Так, для списка, описанного тегом `<ol type="I" start="5">`, нумерация начнется с римской цифры пять (`V`).

В упорядоченных списках HTML невозможно продолжить нумерацию автоматически из предыдущего списка или убрать ее для некоторых элементов. Однако существует возможность пропустить любое число элементов списка, установив для тега `<li>` атрибут `value`, значением которого также может быть только арабская цифра. Нумерация для последующих элементов списка продолжается с нового значения. В листинге 7.3 показано составление таблицы римских цифр с помощью этого атрибута. Листингу соответствует пример Glava_7\Ordered.html с компакт-диска.

Листинг 7.3. Нумерованный список с атрибутом `value`

```
<html>
<head>
 <title>Таблица римских цифр</title>
```

```
</head>
<body>
<ol type="I">
<li>один
<li value="5">пять                <li value="10">десять
<li value="50">пятьдесят          <li value="100">сто
<li value="500">пятьсот           <li value="1000">тысяча
</ol>
</body></html>
```

Любой элемент нумерованного или маркированного списка может содержать *вложенный* нумерованный или маркированный список, что позволяет создавать *многоуровневые* списки. При этом следует соблюдать правило контейнерного вложения тегов: список нижнего уровня целиком находится внутри элемента `<li>` списка верхнего уровня (или *перед следующим* тегом `<li>`, если элементы `<li>` внешнего списка не закрываются). Число уровней вложения теоретически не ограничено, но на практике оно не должно быть слишком большим, ведь элементы каждого нового уровня списка браузер выведет с отступом вправо. В листинге 7.4 показан HTML-код, реализующий многоуровневый список на основе тегов `<ul>`, `<ol>` и `<li>`, а на рис. 7.2 — внешний вид списка.

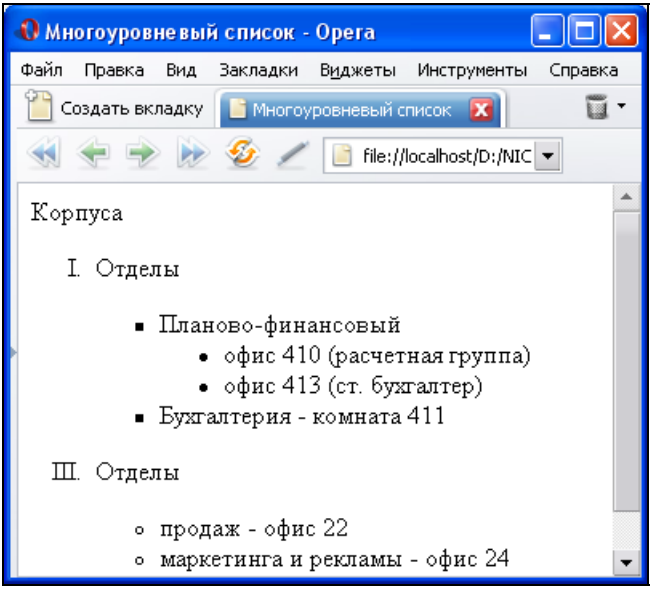

Рис. 7.2. Многоуровневый список

Листинг 7.4. Многоуровневый список

```
<html><head>
 <title>Многоуровневый список</title>
</head>
<body>
<p>Корпуса
<ol type="I">
<li>
  <p>Отделы
  <ul type="square">
    <li>Планово-финансовый
      <ul type="disc">
        <li>офис 410 (расчетная группа)
        <li>офис 413 (ст. бухгалтер)
      </ul>
    <li>Бухгалтерия - комната 411
  </ul>
<li value="3">
  <p>Отделы
  <ul type="circle">
    <li>продаж - офис 22
    <li>маркетинга и рекламы - офис 24
  </ul>
</ol>
</body></html>
```

На компакт-диске с примерами откройте документ Glava_7\MultiLevel.html. Обратите внимание на визуальное выравнивание кода листинга с помощью отступов. Это не имеет никакого значения для браузера, но помогает разработчику не запутаться в создаваемом им сложном списке.

7.3. Списки определений и другие типы списков

Тег `<dl>` (буквально от англ. "definition list") создает *список определений*. Он отличается от остальных списков тем, что каждый его элемент представляется двумя блочными тегами HTML:

- `<dt>` (от англ. "definition term" — термин определения) содержит *имя* или *название* определяемого термина;

- <dd> (от англ. "definition description" — описание определения) содержит *описание* или *определение* этого термина.

Таким образом, список определений является списком пар "свойство — значение".

В тег <dt> могут быть включены любые *текстовые, но не блочные* элементы. Тег <dd> может содержать *как текстовые, так и блочные* элементы, позволяя создавать многоуровневые вложенные списки.

Визуально браузер сдвигает вправо блок описания термина, определяемый тегом <dd>, при этом содержимое тега <dt> никак не выделяется шрифтовыми средствами. Поэтому для лучшей видимости и выделения термина можно использовать любой подходящий тег логического или физического форматирования.

Стандарт допускает возможность давать несколько определений одному термину (несколько <dd> после <dt>) или одно определение нескольким терминам (несколько <dt> с одним <dd>):

```
<DL>
  <DT>www
  <DT>всемирная паутина
  <DD>world wide Web - основной сервис сети Интернет
</DL>
```

В листинге 7.5 показан более объемный пример списка определений, внешний вид которого иллюстрирует рис. 7.3. Содержимое тегов <dt> выделено логическим тегом для придания им большей визуальной привлекательности. Открыть пример в браузере можно, запустив файл Glava_7\Definitions.html с компакт-диска.

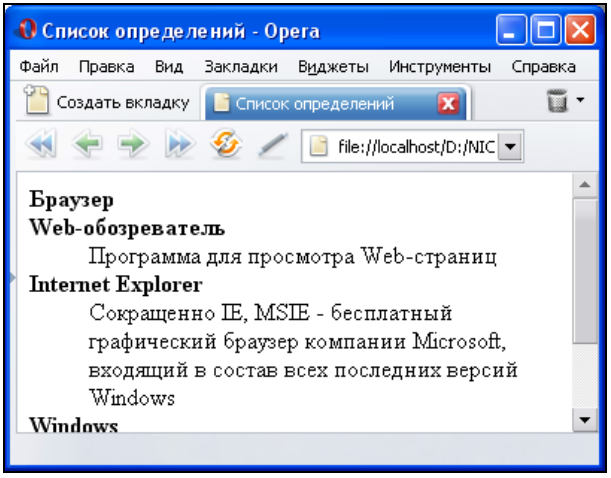

Рис. 7.3. Список определений

Листинг 7.5. Список определений

```
<html>
<head>
 <title>Список определений</title>
</head>
<body>
<dl>
<dt><strong>Браузер</strong>
<dt><strong>Web-обозреватель</strong>
<dd>Программа для просмотра Web-страниц
<dt><strong>Internet Explorer</strong>
<dd>Сокращенно IE, MSIE - бесплатный графический браузер компании Micro-
soft, входящий в состав всех последних версий Windows
<dt><strong>Windows</strong>
<dd>Семейство операционных систем компании Microsoft
</dl>
</body></html>
```

Списки определений применяются довольно широко. Одно из применений, согласно примеру из спецификации, — представление диалогов, в которых элемент `<dt>` содержит имя говорящего, а `<dd>` — его (или ее) слова.

Список определений удобен и для составления перечня адресов Web-страниц с описаниями, терминологического словаря или ленты новостей (дата в теге `<dt>` и описание новости — в `<dd>`). Всюду, где элемент данных описывается парой "свойство" и "значение свойства", рекомендуется применять этот тип списков.

Напоследок упомянем о двух устаревших и нежелательных к использованию типах списков.

Тег `<dir>` ранее предназначался для создания многостраничных списков каталогов, `<menu>` — списков меню, состоящих из одного столбца. Оба эти элемента имеют ту же структуру, что и `<ul>`, различаясь только при генерации. На практике браузеры генерируют списки `<dir>` или `<menu>` точно так же, как список `<ul>`.

Эти элементы настоятельно рекомендуется заменять тегом `<ul>`.

Глава 8

Таблицы — основной инструмент Web-дизайнера

В HTML таблицы служат не только для удобного представления данных, но и являются одним из основных средств форматирования и структурирования Web-страницы. Эта глава научит вас основным приемам работы с таблицами.

8.1. Общие принципы табличного представления информации

Все списки, приведенные в *разд. 7.1*, имели важные общие черты — содержащиеся в них данные либо являлись *перечислениями* каких-то объектов, либо описывали некоторые *последовательные процессы*. Теперь представим, что нам нужно описать множество *объектов*, имеющих общие для всех *свойства*. Так, объектами могут быть сотрудники нашей фирмы, а свойствами — фамилия, имя, отчество, занимаемая должность, номер отдела и оклад, или же объектами являются наши любимые книги, а свойствами каждой из них — название книги, список ее авторов, наименование издательства и год издания, краткая аннотация и т. п. Думаю, уже понятно, что подобные данные лучше всего описать в виде *таблицы*.

При этом каждое интересующее нас свойство объекта будет описано собственным *столбцом* таблицы или *полем*, а информацией об одном объекте будет одна *строка* или *запись* таблицы. Помимо удобного для человека и компактного представления данных, таблицы важны и тем, что легко поддаются автоматизированной обработке, принципы которой почти не зависят от конкретного аппаратного или программного обеспечения.

Все существующие системы управления базами данных (СУБД) построены на основе кратко изложенной нами *реляционной* модели представления данных, описывающей произвольные объекты и их свойства в виде системы взаимосвязанных таблиц.

Итак, именно таблицы дают возможность расположения многомерных данных по строкам и столбцам. При этом ячейки таблиц в HTML могут содержать текст, графические образы, гиперссылки, заполняемые пользователем формы и даже другие таблицы. Таким образом, перспективы использования таблиц для представления ваших данных огромны, но и это еще не все.

Таблицы в HTML применяются для создания на странице колонок газетного типа, отдельных информационных блоков, и даже для выделения части документа фоновым цветом.

Чаще всего просматриваемая вами в Интернете Web-страница имеет невидимый "скелет", образованный сложной системой вложенных таблиц. Между прочим, многие "мэтры" Web-дизайна рекомендуют, по возможности, избегать использования таблиц *только* для размещения элементов на экране. Причин этому две:

- изменение пользователем размеров окна или шрифтов может исказить внешний вид таблицы или привести к появлению "лишних" полос прокрутки;
- большинство браузеров не отображает таблицу до тех пор, пока она не будет загружена полностью. В случаях, когда весь или почти весь документ оформлен в виде единой таблицы, это может привести к долгому ожиданию перед "пустым" экраном.

На компакт-диске с примерами в папке Glava_8 записаны документы с именами TextBig.html и TableBig.html. Первый из них представляет собой довольно большой фрагмент текста, размеченного тегами абзацев. Второй документ содержит тот же текст, заключенный в одну-единственную таблицу. В оба документа встроен скрипт, выводящий по окончании его загрузки окно с сообщением о времени загрузки в секундах. Так что оценить задержку, создаваемую даже одной простейшей таблицей, вы сможете сами, запустив по очереди эти документы (для чистоты эксперимента можно загрузить каждый документ по несколько раз). Представляете, что бывает, когда таблиц много и они сложны?

Тем не менее, таблицы — это единственный сложный элемент HTML, который более или менее одинаково поддерживается всеми браузерами, и зачастую только они оказываются единственным приемлемым вариантом оформления страницы, если, конечно, вы не хотите писать отдельные версии страниц для каждого браузера.

Основная проблема организации таблиц в HTML — необходимость перевода информации из "одномерного" текстового вида в двумерный. Мы пишем текст и размечаем его тегами, а в окне браузера должна появиться таблица

с ячейками. Кроме того, число тегов и атрибутов, описывающих различные элементы таблиц, не так уж мало.

Поэтому сразу же акцентируем внимание на главном:

- каждая таблица описывается последовательно по строкам, каждая строка — слева направо по отдельным ячейкам;
- каждая ячейка при этом может быть растянута на несколько строк и/или столбцов;
- внутрь ячейки внешней таблицы может быть вложена внутренняя таблица.

8.2. Возможности таблиц. Все о теге *<table>*

Для удобства работы кратко опишем основные табличные теги, затем подробнее остановимся на возможностях каждого из них.

Каждая таблица, вложенная в контейнерный тег `<table>`, может иметь заголовок в теге `<caption>`, содержащий ее краткое описание. За ним могут следовать необязательные теги `<col>` и `<colgroup>`, определяющие ширину столбцов таблицы и их группировку. Основное содержимое таблицы может состоять из трех секций: необязательных строк верхнего заголовка в теге `<thead>`, необязательных строк нижнего заголовка в теге `<tfoot>` и одного или нескольких элементов `<tbody>`. Каждый элемент `<tbody>` — это группа строк таблицы. Содержимое отдельных строк, входящих в секции `<thead>`, `<tfoot>` или `<tbody>`, задано тегами `<tr>`, а каждая строка таблицы состоит из ячеек, описанных тегами `<th>` или `<td>`, определяющими, соответственно, содержимое заголовочных ячеек и ячеек с данными.

Если тело таблицы состоит из одной секции данных, тег `<tbody>` обычно опускается. Таким образом, из всех перечисленных тегов можно пока запомнить главные:

- тег `<table>…</table>` описывает всю таблицу;
- каждая строка таблицы заключена в тег `<tr>…</tr>`;
- внутри тегов строк содержимое каждой ячейки заключено в тег `<td>…</td>`.

Все табличные теги весьма желательно закрывать. "Умный" современный браузер способен и самостоятельно закрыть некоторые из них, но экспериментировать не стоит — некорректно созданная таблица может быть искажена самым причудливым и непредсказуемым образом!

Листинг 8.1 содержит HTML-код простейшей таблицы из двух строк и двух столбцов, внешний вид которой иллюстрирует рис. 8.1. Этот пример записан на компакт-диске под именем Glava8\Table1.html.

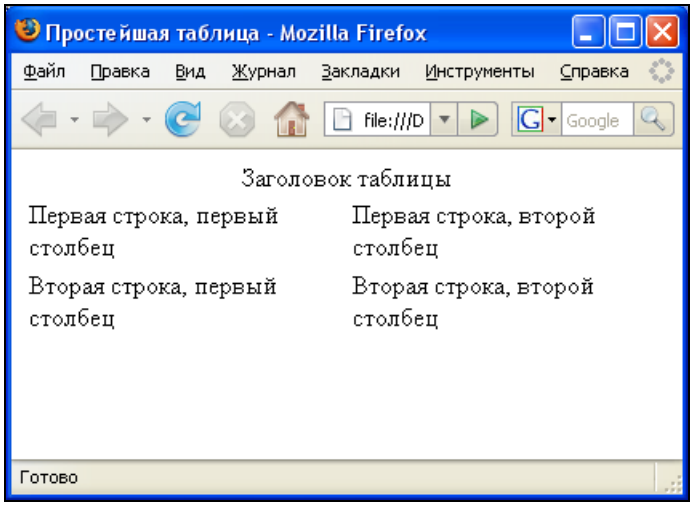

Рис. 8.1. Простейшая таблица

Листинг 8.1. Простейшая таблица

```
<html><head>
 <title>Простейшая таблица</title>
</head>
<body>
<table>
 <caption>Заголовок таблицы</caption>
 <tr>
  <td>Первая строка, первый столбец</td>
  <td>Первая строка, второй столбец</td>
 </tr>
 <tr>
  <td>Вторая строка, первый столбец</td>
  <td>Вторая строка, второй столбец</td>
 </tr>
</table></body></html>
```

Все настройки отображения таблицы здесь оставлены по умолчанию, например, ячейки не разделены видимыми линиями, а ширину таблицы браузер определяет самостоятельно.

Теперь рассмотрим возможности табличных тегов более подробно.

Как мы уже упоминали, тег `<table>...</table>` определяет *всю таблицу*. Его атрибуты подробно описаны в табл. 8.1, все они необязательные. Значениями различных атрибутов могут быть как текстовые, так и числовые данные. Для удобства в табл. 8.1 и далее в качестве значений атрибутов примем следующие обозначения:

- *текст* — любая последовательность символов в двойных кавычках. При отображении текста браузером переносы строк игнорируются, а символы возврата каретки и табуляции заменяются пробелами. Также игнорируются "лишние" пробелы в значении атрибута;
- *размер* — либо целое число, задающее количество экранных точек (*пикселов*), либо доля в процентах от размера окна браузера по горизонтали или вертикали. Например, значение атрибута `"75%"` означает три четверти всего доступного размера, а `"600"` — 600 пикселов;
- *пикселы* — целое число, задающее количество пикселов;
- *цвет* — значение атрибута — код цвета (см. *разд. 9.1*) или его имя, например `"#FFFFFF"` и `"white"` обозначают белый цвет;
- *символ* — единственный символ Unicode, который может быть также специальным (см. *разд. 5.5*);
- значения не указаны — либо атрибут не имеет значений, либо они перечислены в описании.

Таблица 8.1. *Основные атрибуты тега <table>*

Атрибут и тип значения	Описание
`width="размер"`	Задает *рекомендуемую* ширину таблицы в пикселах или в процентах от ширины окна браузера. Задание в пикселах может привести к неудобной для пользователя горизонтальной прокрутке документа. Если атрибут не указан, ширина таблицы вычисляется браузером
`height="размер"`	Задает *рекомендуемую* высоту отображения таблицы. Если атрибут не указан, высота таблицы вычисляется браузером
`border="пикселы"`	Задает ширину рамки таблицы в пикселах. Значение по умолчанию равно нулю, т. е. таблица выводится без рамки
`cellspacing="размер"`	Задает расстояние между ячейками таблицы. Применяется ко всем четырем сторонам ячейки. Значение по умолчанию зависит от браузера
`cellpadding="размер"`	Задает расстояние между содержимым ячеек и рамкой. Применяется ко всем четырем сторонам ячейки. Значение по умолчанию зависит от браузера

Таблица 8.1 *(окончание)*

Атрибут и тип значения	Описание
`frame`	Указывает, какие части внешней рамки таблицы должны быть видны. Возможные значения: `void` — не показывать (принято по умолчанию, если `border=0`); `above` — только верхняя граница; `below` — только нижняя граница; `lhs` — только левая граница; `rhs` — только правая граница; `hsides` — только левая и правая границы; `vsides` — только верхняя и нижняя границы; `box` или `border` — показывать внешнюю рамку полностью (по умолчанию, если `border>0`). Атрибут используется редко
`rules`	Указывает, какие части внутренней рамки таблицы должны быть видны. Возможные значения: `none` — не показывать внутреннюю рамку (принято по умолчанию, если `border=0`) `groups` — показывать только границы между группами строк и столбцов `rows` — показывать только границы между строками `cols` — показывать только границы между столбцами `all` — показывать внутреннюю рамку полностью (принято по умолчанию, если `border>0`) Атрибут используется редко
`align`	Задает выравнивание всей таблицы относительно содержимого документа. Возможные значения — `left`, `center`, `right`. Считается устаревшим, но широко применяется
`bgcolor="цвет"`	Задает фоновый цвет всей таблицы. Считается устаревшим, но широко распространен
`bordercolor="цвет"`	Задает цвет рамки таблицы. Этого атрибута нет в спецификации, и он по-прежнему не поддерживается большинством браузеров. Internet Explorer поддерживает как этот атрибут, так и дополнительные атрибуты `bordercolorlight` и `bordercolordark` для создания "трехмерных" рамок таблицы

Атрибутов довольно много, и имеет смысл указать основные их сочетания, используемые в зависимости от того, для чего нам понадобилась таблица.

Если таблица предназначена исключительно для *структурирования* информации, например, для создания в документе колонок газетного типа, мы не указываем атрибут `border`, чтобы у таблицы не было видимой рамки, а расстояние между колонками устанавливаем атрибутами `cellpadding` или `cellspacing`. Следует учесть, что при указании первого атрибута отступ "удваивается", он ведь будет применен к каждой из двух соседних ячеек.

Ширина таблицы чаще всего может быть задана в процентах от ширины окна браузера, для надежности можно также указать атрибут `align="center"`, чтобы таблица была центрирована по горизонтали:

```
<table width="90%" align="center" cellspacing="8">.
```

Как быть с указанием *высоты* таблицы и с выравниванием *по вертикали*? Хотя в спецификации ничего не сказано о применимости атрибута `height="размер"` к тегу `<table>`, все проверенные браузеры с задачей справились. Значит ли это, что атрибут необходим? Чаще всего — нет, ведь указанные нами "сто процентов" от высоты окна браузера или "семьсот пикселов" вовсе не гарантируют, что содержимое таблицы в них поместится, или, напротив, может оказаться так, что содержимое заполняет таблицу лишь на несколько процентов. Не будем забывать, что верстка HTML — *логическая*, а все используемые Web-мастером размеры и высоты *относительны*. Поэтому применение атрибута высоты таблицы оправдано лишь в немногих случаях, таких, как изготовление специальной страницы-заставки, заполняющей все окно браузера, или создание таблицы, предназначенной для размещения рисунка строго определенного размера.

Что же касается *вертикального выравнивания* содержимого таблицы, вполне логично, что соответствующий атрибут `valign` предназначен только для отдельных строк и ячеек, а не для всей таблицы. В некоторых браузерах этот описанный ниже атрибут, указанный для всей таблицы, приведет к тому, что содержимое всех ячеек окажется выровненным в соответствии с его значением.

В случаях, когда таблица предназначена для *размещения* информации, будет неплохо, если у нее есть сплошная видимая рамка, а содержимое немного отступает от краев ячейки, не сливаясь с рамкой. Таким образом, мы устанавливаем ненулевое значение атрибутов `border` и `cellpadding`, а нулевое — для `cellspacing`, если только вы не хотите получить "двойные" рамки:

```
<table width="90%" align="center" border="1" cellpadding="4".cellspacing="0">.
```

Использовать ли атрибут `bordercolor` — дело вкуса. Пользователи Opera, например, его действия просто не увидят.

Наконец, для создания различных таблиц-разделителей и специальным образом форматированных таблиц можно вспомнить о существовании атрибутов `frame` и `rules`:

```
<table width=90% align="center" border="1" frame="hsides" rules="rows">.
```

Этот код создаст таблицу, в которой видимы все горизонтальные разделители ячеек и невидимы вертикальные.

Тег `<caption>` определяет *заголовок таблицы*. Он должен употребляться только как первый элемент внутри тега `<table>`. Его содержимым могут быть исключительно текстовые элементы. Единственный атрибут этого тега — уже знакомый нам `align`, принимающий значения `top` (заголовок находится над таблицей, значение по умолчанию) и `bottom` (заголовок под таблицей).

Второй важнейший табличный тег — `<tr>...</tr>` (table row) — определяет *одну строку* таблицы. Его содержимое — один или несколько тегов `<th>` и `<td>`. Тег может иметь ряд атрибутов, все они необязательны. Используемые в настоящее время атрибуты тега описаны в табл. 8.2.

Таблица 8.2. *Основные атрибуты тега <tr>*

Наименование	**Значение**
`align`	Горизонтальное выравнивание содержимого ячеек строки. Возможные значения: `left` — выравнивание влево; `center` — выравнивание по центру ячеек; `right` — выравнивание вправо; `justify` — выравнивание по левому и правому краю ячеек
`valign`	Вертикальное выравнивание содержимого ячеек строки. Возможные значения: `top` — выравнивание по верхней части ячеек; `middle` — выравнивание по центру ячеек; `bottom` — выравнивание по нижней части ячеек; `baseline` — выравнивание всей строки по общей базовой линии
`bgcolor="цвет"`	Задает фоновый цвет строки таблицы. Считается устаревшим, но широко распространен

В теге строки также возможно задание атрибута `height="размер"` для указания высоты всех ее ячеек. Как уже говорилось, использовать этот атрибут

следует с осторожностью, независимо от того, указан размер в пикселах или процентах. И то, и другое может очень по-разному отобразиться у разных пользователей.

Тег `<td>…</td>` (table data cell) определяет *ячейку данных* таблицы. Его содержимым могут быть как блочные, так и текстовые элементы. Очень похож на него тег `<th>…</th>` (table header cell), определяющий *ячейку заголовка* таблицы.

Атрибуты `align`, `valign`, `bgcolor` и `nowrap` прменяются к `<td>` и `<th>` *точно так же*, как к `<tr>`, однако действие их распространяется только на текущую ячейку. Поскольку ближайшим контейнером верхнего уровня по отношению к ячейке является строка, атрибуты тегов `<td>` и `<th>` могут *переопределять* в данной ячейке действие атрибутов, указанных для всей строки. Если этого не сделано, ячейка *унаследует* ряд атрибутов от строки. Так, в простой однострочной таблице из листинга 8.2 только левая ячейка будет закрашена желтым цветом и выровнена по центру. В правой ячейке действие атрибутов строки `bgcolor="yellow"` и `align="center"` будет прекращено ее собственными атрибутами `bgcolor="green"` и `align="right"`. Результат вы можете видеть на рис. 8.2. Код примера приведен в листинге 8.2 и записан на компакт-диске под именем Glava_8\Table2.html. Кстати говоря, атрибут `bgcolor`, указанный *для всей таблицы*, точно так же мог быть переопределен у конкретной строки или ячейки.

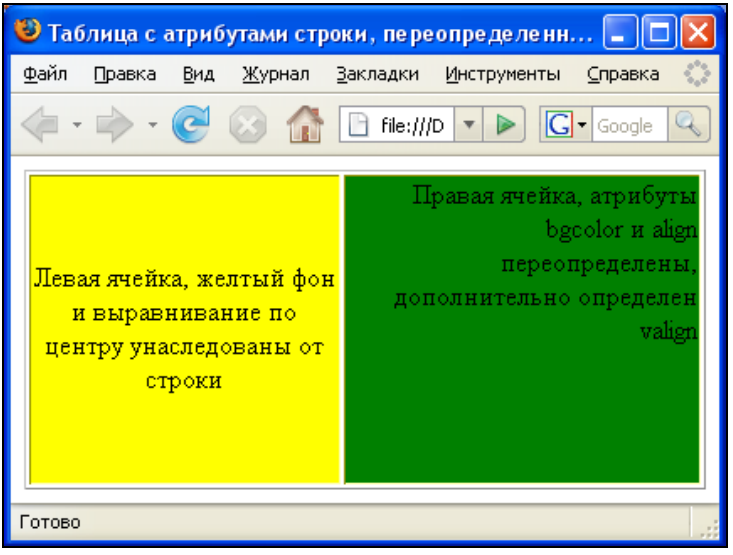

Рис. 8.2. Таблица с атрибутами строки, переопределенными для ячейки

Листинг 8.2. Таблица с атрибутами строки, переопределенными для ячейки

```
<html>
<head>
 <title>Таблица с атрибутами строки, переопределенными для ячейки</title>
</head>
<body>
<table border="1" width="100%" height="100%">
 <tr bgcolor="yellow" align="center">
  <td>Левая ячейка, желтый фон и выравнивание
   по центру унаследованы от строки
  </td>
  <td bgcolor="green" align="right" valign="top">Правая ячейка,
   атрибуты bgcolor и align переопределены, дополнительно
   определен valign
  </td>
 </tr>
</table>
</body></html>
```

Как правило, текстовое содержимое ячеек с данными `<td>` по умолчанию выводится обычным шрифтом и выравнивается по левому краю (`align="left"`), а содержимое заголовочных ячеек `<th>` — полужирным шрифтом с выравниванием по центру (`align="center"`). Выравнивание данных в каждой конкретной ячейке можно изменить, а для шрифтового выделения указать в элементе `<td>` или `<th>` все имеющиеся текстовые теги.

Для закрепления изученных возможностей рассмотрим несложную таблицу с данными, показанную на рис. 8.3. Соответствующий код приведен в листинге 8.3 и содержится на компакт-диске в файле Glava_8\Table3.html.

Листинг 8.3. Таблица с атрибутами тегов `<table>`, `<tr>` и `<td>`

```
<html>
<head>
 <title>Таблица с атрибутами тегов TABLE, TR, TD</title>
</head>
<body>

<table width="400" align="center" border="1" cellpadding="4" cellspacing="0">
```

```
 <caption align="bottom">Сотрудники отдела 2</caption>
 <tr>
  <th width="33%">Фамилия И.О.</th>
  <th width="33%">Должность</th>
  <th width="33%">Дата рождения</th>
 </tr>
 <tr>
  <td>Иванов М.А.</td>
  <td align="center">Менеджер</td>
  <td align="right">12.05.74</td>
 </tr>
 <tr>
  <td>Петрова Н.Н.</td>
  <td align="center">Бухгалтер</td>
  <td align="right">15.01.60</td>
 </tr>
 <tr>
  <td>Сидоров В.В.</td>
  <td align="center">Маркетолог</td>
  <td align="right">11.07.79</td>
 </tr>
</table>
</body></html>
```

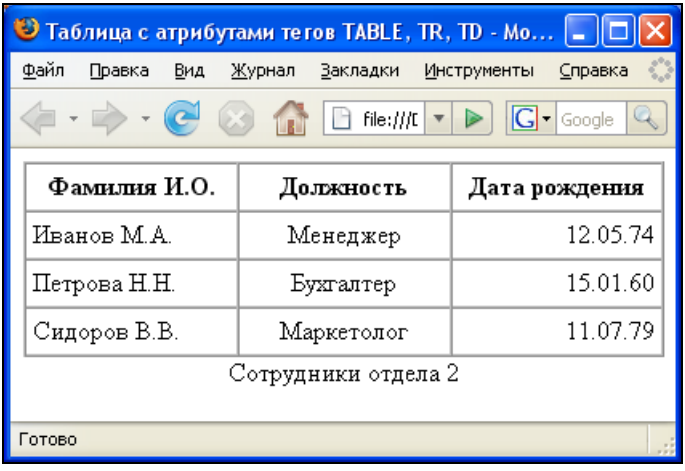

Рис. 8.3. Таблица с атрибутами тегов <table>, <tr> и <td>

Обратите внимание, что для вывода данных второго столбца с выравниванием по центру, а третьего — с выравниванием по правому краю нам пришлось многократно переопределять атрибут `align` соответствующих ячеек, ведь у нас пока нет никаких свойств *столбца* таблицы в целом. Тем не менее, ширина всей таблицы, заданная в теге `<table>` абсолютным значением `400` пикселов, была равномерно распределена между столбцами указанием в тегах их первых ячеек атрибута `width="33%"`. Как видим, сумма процентов в HTML не обязательно равна 100. Задать ширину первого столбца вдвое большей, чем остальные, мы могли, например, указанием атрибута `width="200%"` в ячейке с данными "`Фамилия И.О.`" и `width="100%"` в ячейках с данными "`Должность`" и "`Дата рождения`". Однако все это срабатывает до тех пор, пока определена ширина всей таблицы. В противном случае результат окажется малопредсказуемым.

Важно и то, что нам не понадобилось дублировать атрибут `width="33%"` во всех ячейках строк — полностью распределив пространство первой строки, мы неявно дали браузеру указание сохранить это распределение и для остальных строк.

А что произойдет, если, например, распределить ширину ячеек следующей строки по-другому? В Microsoft Word это не составляет никакого труда — достаточно выделить ячейку в нужной строке и перетащить ее вправо или влево мышью. Увы, с браузером ничего такого не получится. Один раз вычислив ширину столбцов, заданную для ячеек первой строки таблицы, он, в лучшем случае, проигнорирует указания поменять ее в следующих строках. Тем не менее, методом вложения, описанным в *разд. 8.3*, можно создавать таблицы практически любой сложности.

Кроме записи размеров в пискелах и процентах спецификацией предусмотрен еще один способ. Указание *пропорциональной ширины* (например, `width="3*"`) позволяет назначить для ячейки количество свободных частей горизонтального пространства. Если ячейки с абсолютно заданными размерами уже сформированы, браузеры могут генерировать таблицу с учетом деления оставшегося места на части. На практике поддержка этого способа в отношении таблиц весьма ограничена и применять его не рекомендуется.

Конкретные алгоритмы форматирования таблиц могут отличаться в различных браузерах, но общие принципы все же есть.

Как правило, браузер сначала суммирует все имеющиеся указания ширины, а затем уже пытается распределить их исходя из заданной ширины контейнера верхнего уровня либо из имеющегося свободного места, если ширина всей таблицы не указана. По возможности браузер всегда форматирует содержимое таблицы так, чтобы избежать появления горизонтальных полос прокрутки.

При изменении размеров окна ширина столбцов пропорционально уменьшается или увеличивается для обеспечения заданной в процентах ширины таблицы. При этом текстовое содержимое ячеек может переноситься по словам, а высота таблицы увеличиваться или уменьшаться.

Если ширина таблицы задана в пикселах, она выдерживается более строго, а при уменьшении размеров окна до ширины таблицы появляется горизонтальная полоса прокрутки. Если в пикселах задана также высота, браузер сохраняет ее с помощью вертикальной полосы прокрутки. При увеличении размеров окна больше заданной в пикселах ширины таблицы, браузер выравнивает всю таблицу относительно остального содержимого в соответствии с ее атрибутом `align` или его значением по умолчанию `left`.

Если никакие размерные атрибуты не указаны, таблица будет иметь минимально возможные ширину и высоту.

Некоторые браузеры "не признают" указания ширины таблицы больше 100%, хотя в Internet Explorer и Opera это возможно.

Как видим, проектирование таблиц — дело непростое. Кроме того, весьма распространены случаи, когда нужны более сложные таблицы, чем те, что были показаны в наших примерах. Помочь здесь могут атрибуты тегов `<td>` и `<th>`, приведенные в табл. 8.3.

Таблица 8.3. *Дополнительные атрибуты тегов <td> и <th>*

Наименование	Значение
`nowrap`	Запрещает перенос содержимого по строкам внутри ячейки. Применение параметра может привести к появлению горизонтальных полос прокрутки, следует использовать его с осторожностью
`rowspan="число"`	Задает число строк таблицы, занятых ячейкой. Если не указан, значение равно `1`
`colspan="число"`	Задает число столбцов таблицы, занятых ячейкой. Если не указан, значение равно `1`

Атрибут `nowrap`, фактически, *фиксирует* указанные ширину и высоту ячейки. Если ширина и высота не указаны, он просто отключает возможность автоматического разбиения текста ячейки на строки. Очевидно, что применение его имеет смысл, если ширина и, возможно, высота заданы в пикселах. С одной стороны, это необычайно удобно — `nowrap` позволяет четко определить размеры отдельных ячеек, содержащих важную информацию, вид которой не должен быть искажен. С другой стороны, неоправданное применение атри-

бута способно привести к появлению очень широких или слишком высоких ячеек. Поэтому при работе с ним необходимо соблюдать следующие правила:

- фиксируйте с помощью `nowrap` размеры только для *некоторых* ячеек; в таблице должен быть, по крайней мере, один столбец, размеры всех ячеек которого заданы в процентах или не указаны вовсе. За счет этого столбца (лучше всего, если это "заведомо широкий" столбец с основным содержимым страницы) выполняется "подгонка" таблицы под ширину окна в браузере пользователя;
- не применяйте `nowrap` *только* для запрета переноса текста в ячейке; в большинстве случаев этого можно добиться заменой обычных пробелов на жесткие (` `).

Атрибуты `rowspan` и `colspan` позволяют "растянуть" ячейку на указанное число строк или столбцов. Они очень широко применяются для создания общих заголовков нескольких столбцов или строк, а также для формирования любых нестандартных таблиц. Единственное, чего не позволяют сделать эти два атрибута — создать ячейку, "выступающую" за левый или правый край таблицы, чего легко добиться в Microsoft Word. Но если вдуматься, это и не нужно — ведь к "выступающей" влево или вправо ячейке сверху и снизу могут быть добавлены ячейки с пустым содержимым, дополняющие таблицу до классического "прямоугольного" вида. Все удивительные эффекты "нелинейного" расположения информации на странице создаются именно так.

Оба атрибута могут быть указаны у одной ячейки, тогда она займет соответствующее число строк и столбцов. Внешний вид подобной таблицы показан на рис. 8.4, а в листинге 8.4 приведен ее HTML-код. На компакт-диске с примерами соответствующий документ хранится под именем Glava_8\Table4.html.

Листинг 8.4. Таблица с объединенными ячейками

```
<html>
<head>
 <title>Таблица с объединенными ячейками</title>
</head>
<body>
<table border="1" cellpadding="4" cellspacing="2">
  <tr>
    <td colspan="2" rowspan="2"> </td>
    <td colspan="2">Характеристики</td>
  </tr>
  <tr>
```

```
    <td>Средний рост, см</td>
    <td>Средний вес, кг</td>
  </tr>
  <tr align="center">
    <td rowspan="2">Пол</td>
    <td>Мужской</td>
    <td>177</td>
    <td>73</td>
  </tr>
  <tr align="center">
    <td>Женский</td>
    <td>166</td>
    <td>65</td>
  </tr>
</table>
</body></html>
```

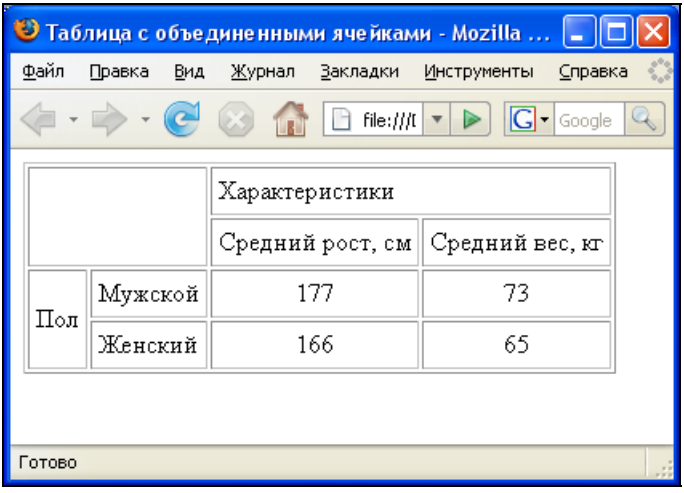

Рис. 8.4. Таблица с объединенными ячейками

Эффект "двойной рамки" здесь возник из-за указания ненулевого расстояния между ячейками (`cellspacing`). Обратите внимание на пустую ячейку, в которой находится неразрывный пробел ` `. Его указание необходимо во всех ячейках, *не содержащих данных* в контейнере `<td>` или `<th>`. В противном случае рамки пустых ячеек будут отображены с искажениями.

Остальные "новшества" этого примера попытайтесь разобрать самостоятельно, он совершенно не сложен для понимания, если не забывать о простом порядке обхода таблицы по ячейкам: слева направо и затем сверху вниз, без пропуска ячеек или строк.

Прежде чем двигаться дальше, скажем несколько слов о разработке ваших собственных таблиц.

Проектирование отдельных таблиц в HTML всегда упрощается, если помнить, что все разделители строк и столбцов могут быть наложены на строго прямоугольную сетку.

Фактически, это означает, что *исходные* ячейки сетки можно *объединять*, но не *разбивать*, примерно так же, как в Microsoft Excel. Нарисовав сетку будущей таблицы на бумаге и объединив на ней нужные ячейки, мы сможем затем описывать таблицу всего тремя тегами `<table>`, `<tr>` и `<td>`, соблюдая простой и естественный порядок обхода ячеек. Невнимательное объединение ячеек способно попортить немало крови начинающему Web-мастеру, создавая самые непредсказуемые искажения вида таблицы. Поэтому предварительное проектирование сложной таблицы "от исходной сетки" совершенно необходимо.

Ну а если рисовать на бумаге не хочется, удобный макет таблицы можно сделать и в Excel, поупражнявшись в объединении и обрамлении ячеек его *рабочего листа*. Есть и еще более удобный путь, но о нем мы поговорим в *разд. 8.5*.

Далее опишем упомянутые в начале раздела теги *группировки строк*.

С точки зрения группировки, строки таблицы могут быть разбиты на три вида:

- строки надзаголовка (элемент `<thead>`);
- строки подзаголовка (элемент `<tfoot>`);
- строки данных (один или несколько элементов `<tbody>`).

Такая группировка была предусмотрена стандартом в предположении, что браузеры при работе с длинными таблицами обеспечат прокрутку строк с данными при сохранении неподвижных строк надзаголовка и подзаголовка, а при печати таких таблиц программы смогут использовать надзаголовок и подзаголовок в качестве колонтитулов страницы. Однако современные браузеры этого не делают и либо просто отображают содержимое `<thead>` и `<tfoot>` в виде строк данных без учета порядка их следования, либо помещают соответствующие строки в начало и конец таблицы.

Теги группировки строк могут включать следующие элементы:

- `<thead>`…`</thead>` (table header) — группу строк надзаголовка таблицы. Его содержимое — один или несколько тегов `<tr>`, которые, в свою очередь, должны состоять из элементов `<th>`.

- `<tfoot>…</tfoot>` (table footer) — группу строк подзаголовка таблицы. Его содержимым также являются один или несколько тегов `<tr>`, которые должны состоять из элементов `<th>`.
- `<tbody>…</tbody>` (table body) — группу строк тела таблицы. Содержимое этого элемента — один или несколько тегов `<tr>`, содержащие элементы `<th>` или `<td>`.

Все три элемента могут иметь атрибуты `align` и `valign` со значениями, указанными в табл. 8.2.

Теги `<thead>`, `<tfoot>` и `<tbody>` должны содержать одно и то же число столбцов (т. е. контейнеров `<td>` или `<th>`).

Таким образом, общая структура таблицы может быть такой, какой она показана в листинге 8.5, а также на примере с компакт-диска Glava_8\TableGroup1.html. Листингу соответствует рис. 8.5.

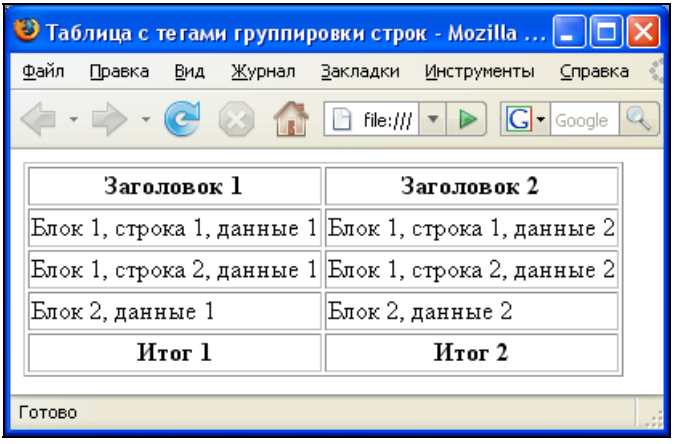

Рис. 8.5. Таблица с тегами группировки строк

Листинг 8.5. Таблица с тегами группировки строк

```
<html>
<head>
 <title>Таблица с тегами группировки строк</title>
</head>
<body>
<table border="1">
 <thead>
```

```
  <tr>
   <th>Заголовок 1</th>
   <th>Заголовок 2</th>
  </tr>
 </thead>
 <tfoot>
  <tr>
   <th>Итог 1</th>
   <th>Итог 2</th>
  </tr>
 </tfoot>
 <tbody>
  <tr>
   <td>Блок 1, строка 1, данные 1</td>
   <td>Блок 1, строка 1, данные 2</td>
  </tr>
  <tr>
   <td>Блок 1, строка 2, данные 1</td>
   <td>Блок 1, строка 2, данные 2</td>
  </tr>
 </tbody>
 <tbody>
  <tr>
   <td>Блок 2, данные 1</td>
   <td>Блок 2, данные 2</td>
  </tr>
 </tbody>
</table>
</body></html>
```

Браузер Safari версии 3.0.1 не смог отобразить содержимое заголовочных ячеек вообще, остальные браузеры вывели таблицу без искажений, но никак не учли разбиение ее на блоки тегами `<tbody>`. Так что при использовании этого способа следует быть осторожным.

До сих пор мы не задействовали никаких средств работы со *столбцами* таблицы, между тем, они предусмотрены в языке HTML и необходимы довольно часто.

Тег *группы столбцов* `<colgroup>…</colgroup>` позволяет указывать информацию о столбцах таблицы. По умолчанию таблица состоит из одной группы столбцов, если в ней нет этого тега. Кроме того, тег *столбца* `<col>` позволяет

присвоить одинаковые атрибуты сразу нескольким столбцам, не указывая их для каждой ячейки.

Содержимым тега `<colgroup>` должны быть нуль или более элементов `<col>`. Он должен располагаться после необязательного элемента `<caption>` и перед необязательным элементом `<thead>`.

Атрибут тега `span="число"` задает число столбцов в группе, по умолчанию его значение равно `1`.

Атрибут `width="размер"` задает ширину *каждого* столбца в группе. Особое значение размера `"0*"` означает, что столбец имеет *минимально возможную* ширину, необходимую для его отображения.

Тег `<col>…</col>` (column) позволяет указать значения атрибутов для конкретного столбца таблицы. Он также должен располагаться после элемента `<caption>` и перед элементом `<thead>`, при этом теги `<col>` могут употребляться как внутри `<colgroup>`, так и самостоятельно. Тег может иметь те же атрибуты, что и `<colgroup>`.

Кроме атрибутов `span` и `width`, к `<colgroup>` и `<col>` применимы атрибуты `align` и `valign` со стандартными значениями, указанными в табл. 8.2.

Таблица, HTML-код которой приведен в листинге 8.6, а внешний вид — на рис. 8.6, занимает 90% текущей ширины окна браузера и состоит из восьми столбцов. Первые 6 столбцов, содержащих названия будних дней недели, имеют фиксированную ширину по 30 пикселов, столбец с данными "`Вс`" занимает 25% оставшегося места (указание `width="1*"`), а последний столбец с данными "`Неделя`" — оставшиеся три четверти свободного места (указание `width="3*"`). Обратите внимание, что представленный на рис. 8.6 браузер Mozilla Firefox версии 2.0 проигнорировал выравнивание данных последнего столбца по правому нижнему краю (`align="right"`, `valign="bottom"`). Из нескольких браузеров с этой задачей справился только Internet Explorer 7.0. Из этого можно сделать вывод, что при группировке столбцов также следует соблюдать осторожность.

Листинг 8.6. Таблица с тегами группировки столбцов

```
<html><head>
 <title>Таблица с тегами группировки столбцов</title>
</head>
<body>
<table width="90%" align="center" border="1" cellspacing="0">
 <colgroup span="6" width="30" align="center" valign="middle">
 </colgroup>
```

```
 <col width="1*" align="center" valign="middle">
 <col width="3*" align="right" valign="bottom">
 <tr>
  <th>Пн</th><th>Вт</th><th>Ср</th><th>Чт</th><th>Пт</th><th>Сб</th>
  <th><font color="red">Вс</font></th>
  <td>Неделя</td>
 </tr>
</table>
</body></html>
```

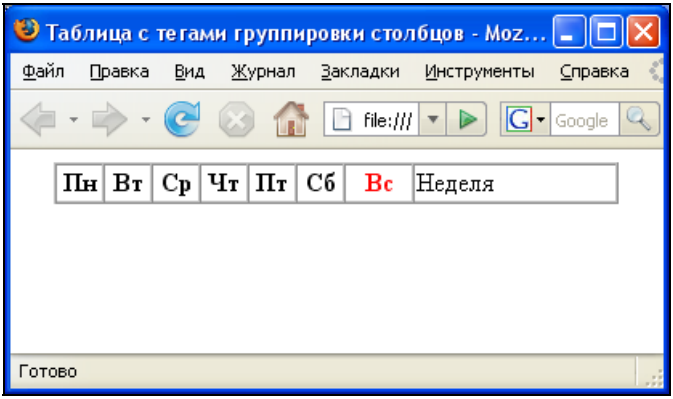

Рис. 8.6. Таблица с тегами группировки столбцов

Этот пример записан на компакт-диске под именем Glava_8\TableGroup2.html.

8.3. Вложение таблиц

Подводя итог сказанному, можно сделать вывод, что существует две основные проблемы работы с таблицами: их двумерность, которую нужно спроектировать "одномерными" тегами, и многократная вложенность табличных элементов друг в друга. Нетрудно оценить и огромное преимущество, достигаемое ценой этой сложности — в ячейки таблиц можно вкладывать разнородное содержимое, в том числе, и другие таблицы, что позволяет создать на странице прямоугольные структуры ячеек любой сложности.

При этом сами по себе правила вложения таблиц в таблицы несложны:

- каждая таблица следующего уровня вложенности полностью размещается внутри контейнерного тега `<th>` или `<td>` внешней таблицы и не создается где-либо за пределами этих тегов;

- таблица каждого уровня может содержать произвольное число вложенных таблиц, каждая из которых расположена в пределах ячейки тега таблицы верхнего уровня;
- каждая таблица должна иметь корректно указанные открывающие и закрывающие теги.

Проектирование вложенной таблицы, ничем не отличается от случая, когда вложений нет. Однако следует учесть, что атрибуты вложенной таблицы, указывающие размеры, будут восприняты браузером *относительно внешней таблицы*. Так, таблица с атрибутом ширины `width="90%"`, вложенная в таблицу с атрибутом `width="600"`, займет 90% не от ширины окна браузера, а от 600 пикселов, составляющих ширину внешней таблицы. Кроме того, следует учесть, что ненулевые атрибуты `cellpadding` и `cellspacing` внешней таблицы также могут потребовать места.

Разумеется, никто не запретит нам попытаться вложить таблицу шириной 100% окна браузера в таблицу шириной 600 пикселов — но результат будет так же на 100% не соответствовать нашим ожиданиям.

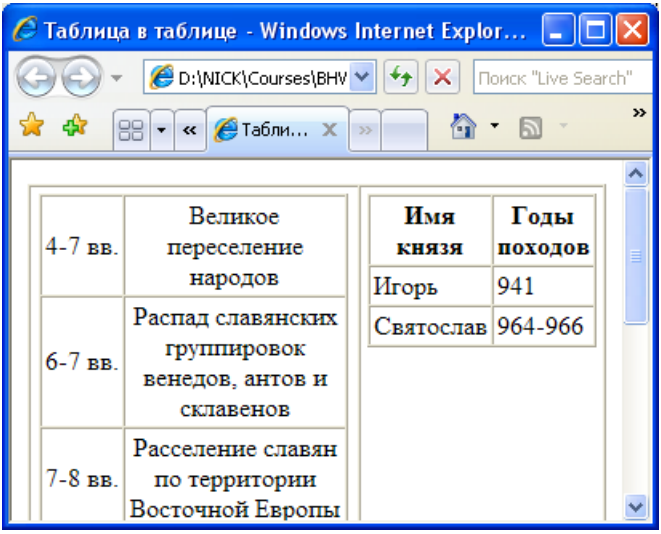

Рис. 8.7. Вложенные таблицы

Зачем нужны вложенные таблицы? Во-первых, вложение таблиц часто позволяет создать удобное и компактное двумерное представление данных. Во-вторых, манипулируя атрибутами вложенных таблиц, мы можем реализовать сколь угодно сложную структуру своего сайта. Подробнее этот важнейший вопрос будет рассмотрен в *разд. 8.4*. Пока же взглянем на рис. 8.7, где

показана таблица, состоящая из двух ячеек. В ее левую ячейку помещена вложенная таблица из шести строк и двух столбцов, содержащая перечень исторических событий. В правой ячейке внешней таблицы также находится вложенная таблица из двух строк и двух столбцов. Теперь посмотрим на листинг 8.7. Обратите внимание, что правая вложенная таблица выровнена по верхнему краю ячейки внешней таблицы за счет указания атрибута `valign="top"` в ее правой ячейке. В левой вложенной таблице центрирование информации в ячейках достигнуто за счет тега `<colgroup>`. Чтобы даты в первом столбце не переносились на другую строку, они заключены в контейнер `<nobr>`. На практике самые внешние таблицы, как правило, служат только для структурирования информации и видимые рамки для них не указываются.

Посмотреть этот пример полностью вы сможете, открыв документ Glava_8\TabInTab.html с компакт-диска.

Листинг 8.7. Вложенные таблицы

```
<html>
<head>
 <title>Таблица в таблице</title>
</head>
<body>
<table width="100%" border="1" cellpadding="4" cellspacing="0">
 <tr>
  <td>
    <table width="90%" align="center" border="1" cellpadding="2"
    cellspacing="0">
      <colgroup span="6" align="center" valign="middle"></colgroup>
      <tr><td><nobr>4-7 вв.</nobr></td><td>Великое переселение
      народов</td></tr>
      <tr><td><nobr>6-7 вв.</nobr></td><td>Распад славянских группировок
      венедов, антов и склавенов</td></tr>
      <tr><td><nobr>7-8 вв.</nobr></td><td>Расселение славян по
      территории Восточной Европы</td></tr>
      <tr><td><nobr>860 г.</nobr></td><td>Первый поход русов на Царьград
      (Константинополь)</td></tr>
      <tr><td><nobr>862 г.</nobr></td><td>Мифическое "призвание
      варягов"</td></tr>
      <tr><td><nobr>882 г.</nobr></td><td>Захват Киева Олегом и основание
       Великого княжества Киевского</td></tr>
```

```
    </table>
  </td>
  <td valign="top">
    <table width="90%" align="center" border="1" cellpadding="2"
    cellspacing="0">
      <tr><th>Имя князя</th><th>Годы походов</th></tr>
      <tr><td>Игорь</td><td>941</td></tr>
      <tr><td>Святослав</td><td>964-966</td></tr>
    </table>
  </td>
 </tr>
</table>
</body></html>
```

8.4. Организация блоков на сайте с помощью таблиц

Итак, основное назначение вложенных таблиц — структурирование информации на вашей Web-странице. Ведь данные, которые потребовали бы для своего отображения вложения одних таблиц в другие, встречаются не так уж часто. А вот "портальный дизайн", при котором страница представляет собой многоуровневую иерархию навигационных, информационных, рекламных и различных других блоков, вы наверняка имели удовольствие наблюдать на самых разных сайтах. В этом разделе мы научимся создавать как раз подобные иерархии блоков. Под *блоком* мы будем понимать организованный с помощью системы таблиц информационный элемент сайта, предназначенный для вывода определенного вида содержимого. Как правило, блок имеет *визуальное обрамление*, *заголовок* и *содержимое*. Почти всегда блоки на странице организованы в два или три столбца. Сделать то и другое нам вполне по силам. Сначала "изобразим" сам блок. Он будет представлять собой таблицу, заключенную в рамку синего цвета. Верхняя строка этой таблицы будет включать в себя заголовок, а нижняя — содержимое. Для создания рамки нам придется вложить одну таблицу в другую — ведь атрибут `bordercolor` из популярных браузеров поддерживает только Internet Explorer! Более экономичный способ реализации рамки блока без вложения таблиц, связан с указанием в открывающей части тега `<table>` стиля `style="border-color: код_цвета"` (см. *главы 9* и *14*).

Внешний вид полученного блока представлен на рис. 8.8, а HTML-код — в листинге 8.8. Посмотреть пример в своем браузере вы сможете, открыв документ Glava_8\Block1.html с компакт-диска.

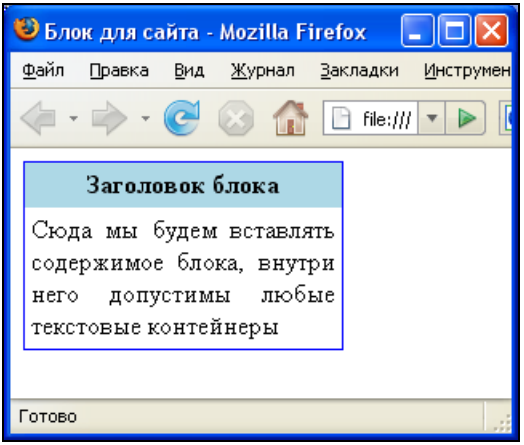

Рис. 8.8. Простой информационный блок для сайта

Листинг 8.8. Простой информационный блок для сайта

```
<html>
<head>
 <title>Блок для сайта</title>
</head>
<body>
<table width="200" border="0" cellpadding="1" cellspacing="0" bgco-
lor="blue">
 <tr>
  <td>
   <table width="100%" border="0" cellpadding="4" cellspacing="0">
    <tr align="center" valign="middle" bgcolor="lightblue">
     <th>Заголовок блока
     </th>
    </tr>
    <tr align="justify" valign="top" bgcolor="white">
     <td>Сюда мы будем вставлять содержимое блока,
      внутри него допустимы любые текстовые контейнеры
     </td>
    </tr>
   </table>
</td></tr></table>
</body></html>
```

Как мы уже знаем, для вложения таблиц достаточно помещать в тег `<td>` одной таблицы полное содержимое другой. Напишем код для "каркаса" несложного сайта. Наш сайт будет состоять из двух столбцов. Узкий левый столбец будет предназначен для вывода блоков с навигационными ссылками, различными краткими аннтотациями, возможно, счетчиками и рекламными баннерами. Широкий правый столбец будет содержать основное наполнение страницы. Неплохо также, если у всех страниц сайта будет общая "шапка" сверху и нижняя информационная строка, куда можно поместить информацию об авторских правах и различные служебные сведения, такие как дата обновления документа. Применив знакомые табличные теги, получаем HTML-код, приведенный в листинге 8.9. Результат его выполнения иллюстрирует рис. 8.9. На компакт-диске этот пример записан под именем Glava_8\Template1.html.

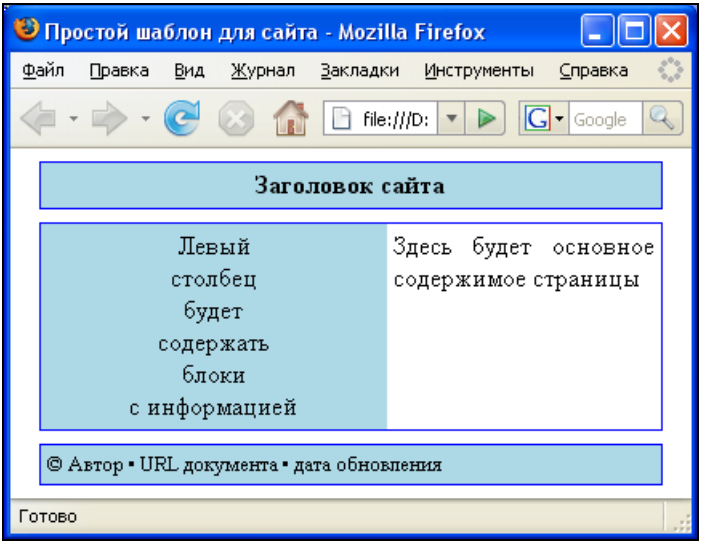

Рис. 8.9. Простой табличный шаблон для сайта

Листинг 8.9. Простой табличный шаблон для сайта

```
<html>
<head>
 <title>Простой шаблон для сайта</title>
</head>
<body>
<!-- Общий титул сайта -->
```

```
<table width="95%" align="center" border="0" cellpadding="1"
cellspacing="0" bgcolor="blue">
<tr><td>
 <table width="100%" align="center" border="0" cellpadding="4"
cellspacing="0" bgcolor="lightblue">
  <tr><th align="center" valign="middle">
   Заголовок сайта
  </th></tr>
</td></tr></table>
</table>
<!-- Таблица-разделитель -->
<table width="95%" align="center" border="0" cellpadding="0" c
ellspacing="0" bgcolor="white">
  <tr><td height="8" align="center" valign="middle" bgcolor="white">
</td></tr></table>
<!-- Основное содержимое, разбитое на 2 столбца -->
<table width="95%" align="center" border="0" cellpadding="1"
cellspacing="0" bgcolor="blue">
<tr><td>
 <table width="100%" align="center" border="0" cellpadding="4"
cellspacing="0">
  <tr>
   <td width="208" align="center" valign="top" bgcolor="lightblue">
    Левый<br>
    столбец<br>
    будет<br>
    содержать<br>
    блоки<br>
    с информацией
   </td>
   <td align="justify" valign="top" bgcolor="white">
    Здесь будет основное содержимое страницы
   </td>
  </tr>
 </table>
</td></tr></table>
<!-- Таблица-разделитель -->
<table width="95%" align="center" border="0" cellpadding="0"
cellspacing="0" bgcolor="white">
  <tr><td height="8" align="center" valign="middle" bgcolor="white">
</td></tr></table>
```

```
<!-- Нижняя информационная строка -->
<table width="95%" align="center" border="0" cellpadding="1"
cellspacing="0" bgcolor="blue">
<tr><td>
 <table width="100%" align="center" border="0" cellpadding="4"
cellspacing="0" bgcolor="lightblue">
  <tr><td align="left" valign="middle"><small>
   &copy; Автор &bull; URL документа &bull; дата обновления
  </small></td></tr>
</td></tr></table>
</table>
</body></html>
```

На рис. 8.9 левый столбец выглядит заметно шире — это следствие очень малого размера окна браузера. Дело в том, что размер левого столбца внешней таблицы зафиксирован атрибутом `nowrap`. Разумеется, это следует учесть при наполнении блоков содержимым — любой объект, требующий для отображения пространства шире 200 пикселов, приведет к искажению задуманного вида страницы. Поэтому, как правило, в подобные блоки помещают только текстовую информацию и небольшие по размерам рисунки или формы.

В листинге основные разделы шаблона отмечены комментариями HTML. Легко сразу же увидеть, что верх и низ страницы оформлены отдельными таблицами. Зачем? Думаю, вы еще не забыли, что таблица отображается большинством браузеров только после полной ее загрузки. "Загонять" всю страницу в одну таблицу означало бы обречь многих пользователей на долгое ожидание перед "пустым" экраном. А быстро загруженный верхний блок, особенно если он красив и хорошо продуман, намного повышает шансы, что пользователь заинтересуется и остальным.

На самом деле, в качестве "верха" куда больше подошел бы графический логотип. Изучив *главу 10*, мы сможем создать и его. Информационный нижний блок также гораздо удобнее делать с помощью JavaScript-технологий, описанных в *главе 15*.

Нам понадобилась и специальная таблица-разделитель, создающая пустое пространство между отдельными таблицами. Почему мы не заполнили пустую ячейку таблицы-разделителя символом ` `?

Дело в том, что пробел — также символ языка, и величина пустого пространства в этом случае будет зависеть от размера шрифта, выбранного у пользователя.

С другой стороны, мы могли бы заменить таблицы-разделители на любой тег, создающий разрыв строки, вроде `<p> </p>` или `<br>`. Правда, размеры

пользовательского шрифта и пустого пространства между таблицами в этом случае опять-таки будут взаимосвязаны.

Существует грамотное решение — прозрачный рисунок размером 1×1 пиксел, растянутый по всему пространству таблицы. Этот способ мы изучим в *разд. 10.7*.

Несмотря на свою простоту, рассмотренный шаблон хорош тем, что корректно реагирует на изменение пользователем размеров основного шрифта, масштабирование страницы и выбор размеров окна браузера в достаточно широких пределах.

Дело за малым — вставить нужное количество блоков в шаблон, заполнить данными постоянные блоки, такие, как навигационное меню, и скопировать файл несколько раз под именами, соответствующими названиям страниц будущего сайта. Подобный пример записан на компакт-диске под именем Glava_8\Template2.html, а фрагмент страницы показан на рис. 8.10. Я не привожу здесь соответствующий листинг из-за ограниченного объема книги, но несколько замечаний все же следует сделать.

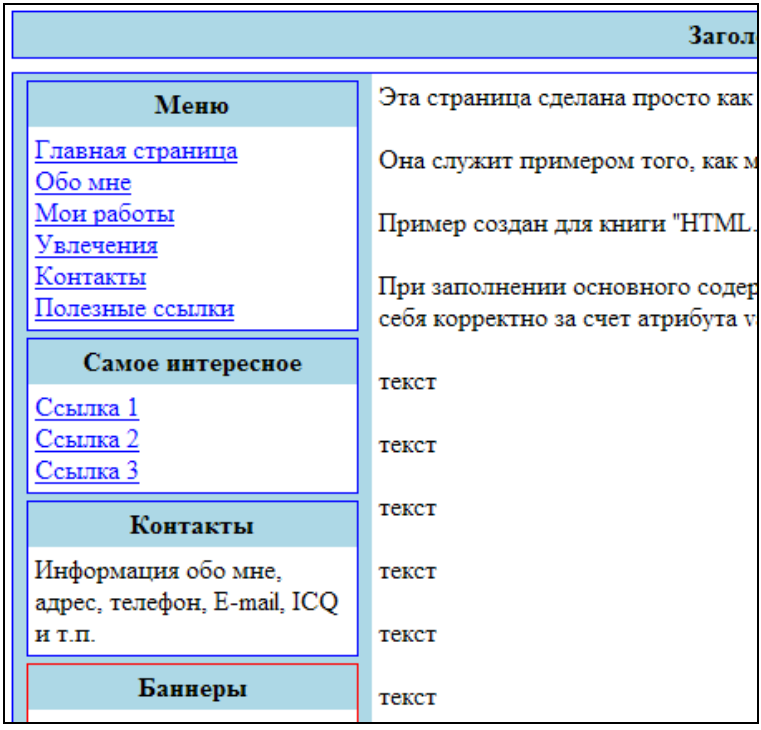

Рис. 8.10. Фрагмент страницы, созданной по шаблону

Между несколькими вставляемыми по вертикали блоками также понадобились таблицы-разделители, в листинге они помечены комментарием "`<!-- Таблица-разделитель для блоков -->`".

Разумеется, ничто не мешает изменить нужные свойства блока после его вставки, как это сделано, например, для блока "Баннеры" (установлено выравнивание по центру и изменен цвет внешней рамки).

Если нам захочется поменять шаблон уже готового сайта, то придется переделывать все файлы. Поэтому шаблон должен быть продуман как можно тщательней, до каждого тега и атрибута. С другой стороны, существуют бесплатные программы, умеющие менять одну строку на другую сразу во всех файлах папки.

В любом случае, прежде чем изменять что-либо на уже работающем сайте, создайте его резервную копию.

8.5. Программное обеспечение для проектирования Web-страниц

Пожалуй, мы достигли того уровня сложности страниц, когда нам может понадобиться автоматизация в написании HTML-кода. К сожалению, размеры и специфика этой книги не позволят нам детально описать хотя бы несколько из многочисленных программ, призванных облегчить труд Web-разработчика, мы лишь кратко обсудим наиболее популярные сегодня *Web-редакторы* для работы с форматом HTML и другими элементами сайта.

Прежде всего, следует сказать, что Web-редакторы делятся на два больших класса.

Текстовые Web-редакторы, по сути дела, представляют собой обычные программы работы с неформатированным текстом, адаптированные к задачам Web-разработки. Это означает, что в отличие от Блокнота Windows, они обеспечат такие функции, как подсветка открывающей и закрывающий части тегов, дополнительные панели инструментов с часто используемыми тегами и ряд других возможностей. Наиболее развитые из этих программ включают функции проверки или оптимизации HTML-кода, добавления скриптов, и даже собственный встроенный браузер. Впрочем, обычно достаточно возможности быстро вызвать браузер, имеющийся в системе.

Таким образом, не избавляя разработчика от необходимости знать язык HTML, текстовые Web-редакторы существенно ускоряют ввод тегов и атрибутов. Собственно, для этого они и пишутся.

Среди наиболее известных приложений этого класса можно выделить программы 1st Page (**www.evrsoft.com**), Bred (**www.astonshell.ru**), TextPad (**www.textpad.com**), Arachnophilia (**www.arachnoid.com/arachnophilia**), CoffeeCup HTML Editor (**www.coffeecup.com**) и Hotdog (**www.sausage.com/products.html**).

Второй класс программ значительно сложнее. Они относятся к так называемым *WYSIWYG-редакторам* (от англ. "What You See Is What You Get", "вы получите то, что видите"). Их суть в том, что мы не работаем непосредственно с HTML-кодом страницы, а размечаем содержимое *визуально*, так, как оно должно выглядеть в результате работы.

Казалось бы, существование таких замечательных редакторов полностью избавляет кого-либо, кроме узких специалистов, от необходимости знать HTML. На самом деле это не так, и, как правило, профессионалы не советуют новичкам "подсаживаться" на WYSIWYG-редакторы.

Во-первых, практически все они генерируют неэффективный и избыточный HTML-код, что приводит к неоправданному увеличению объема страницы и времени ее загрузки. Увеличение размера и сложность кода могут возрастать не на проценты, а в десятки раз даже для несложных проектов. Дело в том, что сам принцип работы WYSIWYG-редактора предполагает максимальное сохранение *видимой* структуры страницы. А ведь изучаемый нами HTML создавался как раз для противоположного — сохранения *логической* структуры документа при достаточно свободном обращении пользователя с внешним видом страницы в браузере. Использование громоздких стилей, не всегда нужное "расширение" возможностей HTML более сложными форматами становятся основными причинами того, что получаемый код оказывается чрезмерно сложным и запутанным.

Во-вторых, работа в визуальных редакторах не дает достаточно возможностей контролировать внутреннюю структуру страницы. В-третьих, многие из WYSIWYG-редакторов предлагают лишь ограниченное число типовых и шаблонных решений страниц. Наконец, в-четвертых, после работы в таких редакторах перейти на классический "чистый HTML" значительно труднее.

Однако во многих случаях WYSIWYG-редакторы полезны. Прежде всего, они позволяют работать быстро, в привычной современному пользователю графической среде и с опорой на готовые решения. Кратко опишем наиболее мощные и популярные на сегодняшний день программные продукты этого класса.

Microsoft FrontPage — это WYSIWYG-редактор, который подходит и новичкам, и опытным Web-дизайнерам. FrontPage содержит средства поддержки ссылок, изображений, таблиц и других элементов. В состав программы входят

шаблоны документов, поэтому для создания несложной страницы достаточно заполнить шаблон текстом и изображениями. Интерфейс FrontPage подобен приложениям Microsoft Office, а процесс редактирования страницы весьма похож на работу в Microsoft Word. Как и в офисной программе PowerPoint, оформление всего сайта можно изменить, выбрав библиотеку готовых тем. FrontPage позволяет интегрировать Web-сайт с базами данных Access и создавать динамические Web-страницы. В программе доступны три режима работы: HTML, WYSIWYG и предварительный просмотр, в процессе работы можно создать диаграмму, отражающую структуру сайта. FrontPage содержит средства проверки корректности ссылок и генерации статических данных. Разработчик может создать список задач, что упрощает управление проектом.

Macromedia Dreamweaver (последние версии программы выпущены компанией Adobe и доступны по адресу **www.adobe.com/products/dreamweaver**) — это профессиональный инструмент, предназначенный для разработки Web-сайтов и управления ими. Dreamweaver ориентирован на опытных разработчиков, которые умеют пользоваться сложными средствами создания и размещения элементов. Интерфейс этого редактора настраивается в соответствии с требованиями пользователя и уровнем его подготовки. Разработчику доступны как обычные средства HTML, так и работа с Dynamic HTML, скриптами, объектами Flash. Применение таблиц для размещения элементов упрощается за счет WYSIWYG-поддержки. Простые в использовании управляющие элементы позволяют включать одну таблицу в другую и автоматически изменять размеры таблиц. Экран программы можно разделять на части и отображать в них как код, так и внешний вид документа. Редактор сообщает о возможных проблемах при отображении документов различными браузерами. Функции управления сайтом позволяют быстро выявлять ошибки в HTML-коде.

Adobe GoLive (**www.adobe.com/products/golive**) представляет собой WYSIWYG-редактор, который содержит инструменты для визуальной разработки Web-страниц и средства управления сайтом. С помощью Adobe GoLive разработчик может создавать статические и динамические Web-сайты, взаимодействующие с базами данных. Характерные особенности этого продукта — позиционирование объектов с точностью до пиксела, возможность "перетаскивания" объектов, а также "связывания" файлов путем указания мышью, защита кода от нежелательных изменений, возможность установки максимального размера файла Web-страницы.

Adobe GoLive предоставляет возможности одновременного просмотра WYSIWYG-представления и HTML-кода в отдельных окнах, определения структуры сайта и схемы навигации с использованием диаграмм, создания приложений электронной коммерции. Редактор также поддерживает прото-

кол WebDAV, позволяющий группе Web-дизайнеров совместно работать над проектом.

Кроме описанных WYSIWYG-редакторов хорошо зарекомендовали себя такие программы, как CuteSite Builder и HotDog Professional.

Отдельно следует сказать о создании Web-страниц средствами программ Microsoft Office. Кроме FrontPage, который входил в версии Office, выпущенные с 1997 по 2002 год, сохранять документы в формате HTML позволяют и многие другие приложения. Так, в новом офисе-2007, Microsoft Word по-прежнему обеспечивает возможность сохранять файлы в форматах **Web-страница** и **Web-страница с фильтром**, приложения PowerPoint и Publisher также могут сохранять документы в HTML.

Однако избыточность и сложность генерируемого HTML-кода при этом очень велики, а его корректность и соответствие стандартам не гарантированы.

В завершении этого небольшого обзора следует добавить, что освоение любого профессионального редактора само по себе весьма трудоемко. И с точки зрения соразмерности средств целям, в большинстве случаев практичней начать именно с изучения HTML, способного, как уже неоднократно подчеркивалось, предоставить мощные, и в то же время доступные средства для реализации ваших творческих идей.

В папке Glava_8\Editors компакт-диска находятся два несложных учебных скрипта, иллюстрирующих описанные подходы к автоматизации создания Web-страницы. Для их работы в браузере должен быть включен JavaScript. Документ editor.html позволяет добавить в текстовое поле распространенные теги и затем открыть полученную страницу для просмотра. Более сложный скрипт, для запуска которого служит файл index.html, дает возможность, имитируя режим WYSIWYG, создать структуру сайта и открыть полученные странички в новых окнах браузера, откуда затем можно просмотреть их HTML-код.

Глава 9

Цвета и управление ими

Выбор различных цветов для оформления элементов Web-страницы способен значительно улучшить внешний вид и "читабельность" документа. В этой главе собраны все необходимые сведения о применении цветов в документах HTML.

9.1. Представление цветов в HTML

Стандартная модель представления цвета в HTML — так называемая *модель RGB*. Согласно ей, любой цвет строится как совокупность интенсивностей трех основных цветов: красного (Red), зеленого (Green) и синего (Blue).

В поддерживаемом большинством современных мониторов видеорежиме Real color на представление одного экранного пиксела отводится 24 бита или 3 байта видеопамяти. Таким образом, интенсивность каждого компонента представлена 1 байтом, а всего возможно 2^{24} (примерно 16,7 млн.) различных цветов. В язык HTML поддержка Real color встроена изначально, так что любой современный браузер "поймет" 24-битовые цвета. Разумеется, если монитор пользователя или выбранное им качество цветопередачи отображают менее 24 бит на пиксел, цвета могут искажаться при выводе, а некоторые из них и вовсе превратятся в смесь "крапинок" из точек различных цветов. Именно так браузер, да и многие другие программы Windows, передают сочетания интенсивностей, недопустимые в том или ином цветовом режиме.

Следует также учитывать, что ни спецификация HTML, ни производители браузеров не гарантируют, что выбранные разработчиком цвета элементов будут переданы идеально точно. Особенно это касается цветов, указанных в элементах `<body>`, `<font>`, и фоновых цветов в таблицах.

Тем не менее, 16,7 млн. цветов — это невообразимое море красок, человеческий глаз способен различить куда меньшее количество оттенков (по разным

данным, всего лишь от нескольких десятков до нескольких тысяч). Но и этого вполне достаточно, чтобы раскрасить страницу всеми мыслимыми и немыслимыми цветами.

Основным способом задания произвольного цвета служит указание его *RGB-кода*. Этот код строится в виде последовательности трех чисел, задающих интенсивности красного, зеленого и синего цветов. Каждое из чисел может принимать значения от 0 до 255 включительно. Понять, почему у интенсивности цвета 256 значений, очень легко. Дело в том, что с помощью одного байта, содержащего 8 битов, можно закодировать именно столько различных чисел, соответствующих возможным интенсивностям из диапазона 0–255.

Сложность лишь в том, что интенсивности эти записываются не в десятичной, а в *шестнадцатеричной* системе счисления. Сделано это вовсе не с целью запутать разработчиков. Наверное, вы знаете, что вся информация в компьютере хранится в виде совокупности двоичных чисел. Шестнадцатеричное число — самая удобная сокращенная запись его двоичного значения. Детали вы можете узнать из любого хорошего учебника по информатике, а пока нам достаточно понимать, что шестнадцатеричное число записывается с использованием шестнадцати различных знаков — привычных нам цифр от 0 до 9 включительно и латинских букв A, B, C, D, E и F вместо недостающих цифр со значениями от 10 до 15.

Перевести число из десятичной записи в шестнадцатеричную и обратно нам поможет всем знакомый Калькулятор Windows. Запустив его из папки стандартных программ меню **Пуск** и выбрав в верхнем меню **Вид** пункт **Инженерный**, мы увидим слева под строкой ввода переключатель систем счисления. Набрав нужное число и выбрав на переключателе значение **Hex**, в строке ввода мы получим шестнадцатеричную запись числа, а переключившись на значение **Dec**, снова вернемся к десятичной записи. Например, легко проверить, что значению полной интенсивности цвета 255 соответствует шестнадцатеричное число FF.

RGB-код цвета записывается в виде `#RRGGBB`, где `RR`, `GG`, `BB` — указанные в форме шестнадцатеричных чисел интенсивности красного, зеленого и синего компонентов соответственно. Спецификация языка требует добавления перед кодом символа "#" (читается "диез"). Так, белому цвету соответствует код `#FFFFFF` (все интенсивности имеют максимальные значения), черному — `#000000` (все интенсивности выключены), красному — `#FF0000` (полностью включена только интенсивность красного компонента), зеленому — `#00FF00`, синему — `#0000FF`.

Указание кодов не всегда удобно, поэтому разработчики стандарта HTML предусмотрели в нем альтернативное и более привычное *наименование* цвета.

Все браузеры поддерживают стандартный цветовой набор из 16 цветов, приведенных в табл. 9.1.

Таблица 9.1. *Названия и коды стандартных цветов HTML*

Цвет	Название в HTML	Код
Черный	Black	`#000000`
Светло-серый	Silver	`#C0C0C0`
Серый	Gray	`#808080`
Белый	White	`#FFFFFF`
Темно-бордовый	Maroon	`#800000`
Красный	Red	`#FF0000`
Пурпурный	Purple	`#800080`
Фиолетовый (фуксин)	Fuchsia	`#FF00FF`
Темно-зеленый	Green	`#008000`
Зеленый	Lime	`#00FF00`
Оливковый	Olive	`#808000`
Желтый	Yellow	`#FFFF00`
Темно-синий	Navy	`#000080`
Синий	Blue	`#0000FF`
Темно-циановый	Teal	`#008080`
Сине-зеленый (циан)	Aqua	`#00FFFF`

Так, например, значения `"#800080"` и `"Purple"` задают пурпурный цвет.

Выбор именно этих цветов в качестве стандартных не случаен. Как легко видеть из табл. 9.1, интенсивности их RGB-компонентов принимают только фиксированные значения `00`, `80` и `FF`, т. е. `0`, `128` и `255` в десятичной системе счисления (кроме цвета `Silver`). Это гарантирует, что даже устройство с невысоким качеством цветопередачи сможет отобразить такие цвета без значительных искажений.

Теоретически и названия, и коды цветов учитывают регистр символов, на практике это не так. Тем не менее, надежней записывать названия и коды в виде, приведенном в табл. 9.1.

Для того чтобы цвета документа корректно отображались различными браузерами и выводились без искажений даже при 256-цветном режиме монитора,

опытные разработчики советуют выбирать только цвета, входящие в так называемый *цветовой куб Netscape* или *безопасную палитру*. За этими терминами скрывается набор цветов, имеющих значения интенсивности всех трех компонентов из ограниченного набора: 00, 33, 66, 99, CC и FF (всего насчитывается 216 таких цветов). Впрочем, качество цветопередачи у большинства современных устройств достаточно высоко, и это требование сейчас теряет актуальность.

Запустив документ Glava_9\colors.html с компакт-диска, вы можете поэкспериментировать с цветами самостоятельно. Для работы этого документа требуется включенный в браузере JavaScript.

Документ предлагает сразу 3 способа работы с цветом:

- выбор цвета из списка по названию;
- выбор цвета из "безопасной палитры" указанием интенсивностей красного, зеленого и синего компонентов;
- выбор произвольного цвета вводом его шестнадцатеричного кода.

Приложение 3 содержит названия и коды 140 стандартных цветов, распознаваемых современными браузерами. Следует иметь в виду, что некоторым цветам могут соответствовать несколько названий, как, например, цвету с кодом `#00ffff` — `Aqua` и `Cyan`. Существуют названия цветов, которые могут не отображаться конкретными браузерами или отображаться по-разному, так что, в общем случае указание кодов цветов *надежнее*, чем названий.

В заключение раздела заметим, что в ряде руководств по HTML можно встретить рекомендацию указывать цвет в виде `"R,G,B"`, где `R`, `G` и `B` — *десятичные* значения интенсивности красного, зеленого и синего компонентов. Этот способ также понимают многие браузеры, но его пока нельзя считать устоявшимся и надежным. Так, браузер Opera 9.02 при указании зеленого фонового цвета страницы в виде "0,255,0" отобразил его как синий.

9.2. Основной цветовой набор документа

Основной цветовой набор HTML-документа указывается через атрибуты тега `<body>` (табл. 9.2).

Значения атрибутов по умолчанию зависят от браузера. Так, Internet Explorer последних версий использует по умолчанию набор атрибутов

```
<body bgcolor="#FFFFFF" text="#000000" link="#0000FF" vlink="#800080">
```

Современные стандарты рекомендуют указывать стиль вместо этих атрибутов.

Таблица 9.2. *Атрибуты тега <body> для указания цвета*

Атрибут	Описание
bgcolor="цвет"	Фоновый цвет страницы
text="цвет"	Цвет основного текста страницы
link="цвет"	Цвет непосещенных ссылок
alink="цвет"	Цвет активной (нажатой в данный момент) ссылки
vlink="цвет"	Цвет посещенных пользователем ссылок

Следует также учесть, что в большинстве браузеров пользователь может переопределить цвета, заданные по умолчанию. Поэтому цветовой набор документа обычно имеет смысл задавать *явно*, независимо от того, используете вы атрибуты из табл. 9.2 или стили. В противном случае, пользователь, установивший у себя черный фоновый цвет, может и не прочитать ваш документ, написанный в предположении, что информация выводится "черным по белому" *у всех*.

Цвет основного шрифта документа можно также поменять с помощью атрибута color="цвет" тега <basefont>, описанного в *разд. 5.2*:

```
<basefont color="green">.
```

9.3. Изменение цвета шрифта

Задавая код цвета любым из описанных в *разд. 9.1* способов, можно изменять цвет *произвольной части текста*, указывая атрибут color="цвет" тега <font>. Например, конструкция

```
<font color="#DAA520 ">золото</font>
```

выведет слово "золото" темно-золотым (goldenrod) цветом.

О разумном применении цветовых выделений мы поговорим подробнее в *разд. 9.5*, пока же приведем пример создания *буквицы*, т. е. особым образом оформленной первой буквы абзаца. На рис. 9.1 разбитый на абзацы текст "украшен" шрифтовым и цветовым выделением первой буквы каждого абзаца с помощью атрибутов тега <font>. HTML-код примера приведен в листинге 9.1.

Обратите внимание на еще одно интересное использование атрибута color в этом примере: красная строка абзаца здесь создана выводом нескольких символов подчеркивания, цвет которых совпадает с фоновым цветом документа. Таким образом, подчеркивания становятся невидимыми, но создают требуемый нам отступ. Все выглядит отлично, но "подделка" быстро обна-

ружится при выделении текста абзаца мышью. Поэтому для создания красной строки следует предпочесть стили или цепочку "жестких пробелов" ` `. На компакт-диске с примерами этот документ записан под именем Glava_9\Letter.html.

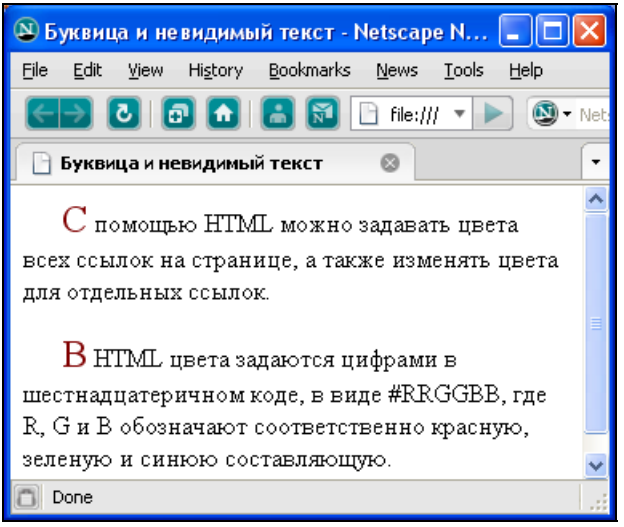

Рис. 9.1. Буквица и невидимый текст

Листинг 9.1. Буквица и невидимый текст

```
<html>
<head>
 <title>Буквица и невидимый текст</title>
</head>
<body bgcolor="#FFFFFF" text="#000000" link="#0000FF" alink="#FF0000"
vlink="#800080">
<p><font color="#FFFFFF">___</font><font size="+2" col-
or="#800000">С</font> помощью HTML можно задавать цвета всех ссылок на
странице, а также изменять цвета для отдельных ссылок.
<p><font color="#FFFFFF">___</font><font size="+2" col-
or="#800000">В</font> HTML цвета задаются цифрами в шестнадцатеричном
коде, в виде #RRGGBB, где R, G и B обозначают соответственно красную,
зеленую и синюю составляющую.
</body></html>
```

Задание цвета *для отдельных ссылок* на странице может понадобиться, например, при их дополнительном выделении в блоках сайта. Так, более светлая,

чем остальные, ссылка лучше привлечет внимание пользователя на темном фоне блока. Возможно также, что вы захотите выделить другим цветом некоторые ссылки по соображениям их важности. Сделав текст ссылки содержимым контейнера `<font>…</font>`, который находится внутри контейнера `<a>…</a>`, вы легко решите поставленную задачу:

```
<a href="menu.html"><font color="#00FF00">Содержание сайта</font></a>.
```

Здесь якорь ссылки "Содержание сайта" выделен зеленым цветом.

Разумеется, теги `<font>` с атрибутом `color` можно вкладывать друг в друга, как и любые другие текстовые теги:

```
<font color="red">M<font color="black">T</font>C</font>.
```

Здесь средняя из трех букв выделена черным цветом, а две другие — красным.

9.4. Изменение цвета в ячейках таблицы

HTML позволяет изменить фоновый цвет всей таблицы, а также ее отдельной строки или ячейки. Для этого достаточно в теге `<table>`, `<tr>`, `<th>` или `<td>` указать атрибут вида `bgcolor="цвет"`.

Как мы помним, атрибут, указанный для контейнера младшего уровня, может переопределить значение атрибута, полученное от родительского контейнера. Именно этим свойством HTML мы пользовались, изучая листинг 8.2 в *разд. 8.2*.

Цвет обрамления таблицы лучше изменять, используя прием с вложением таблиц, рассмотренный в листинге 8.8 из *разд. 8.4*, или применяя стиль.

Основное применение цветового выделения ячеек таблиц — создание различного рода блоков и разделителей. Если цветной прямоугольник нужен только в качестве разделителя, возможно, его удобнее создать, задав параметры `color`, `width` и `size` тега `<hr>`, изученного в *разд. 5.7*.

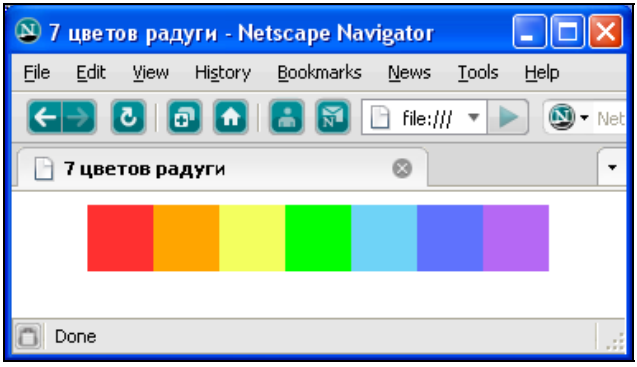

Рис. 9.2. Семь цветов радуги

Цветные ячейки могут служить и для других целей. На рис. 9.2 показана таблица, содержащая в своих ячейках семь цветов радуги. Возможно, подбор их не идеален. Код соответствующего документа приведен в листинге 9.2 (файл Glava_9\Bow.html на компакт-диске). Исправив HTML-код примера по своему усмотрению, вы сможете создать радугу лучше.

Листинг 9.2. Семь цветов радуги

```
<html>
<head>
 <title>7 цветов радуги</title>
</head>
<body bgcolor="white">
<table align="center" border="0" cellpadding="0" cellspacing="0">
 <tr>
  <td width="40" height="40" nowrap bgcolor="#FF3030"> </td>
  <td width="40" height="40" nowrap bgcolor="#FFA500"> </td>
  <td width="40" height="40" nowrap bgcolor="#F3FF5F"> </td>
  <td width="40" height="40" nowrap bgcolor="#00FF00"> </td>
  <td width="40" height="40" nowrap bgcolor="#6FD3F7"> </td>
  <td width="40" height="40" nowrap bgcolor="#5F72FD"> </td>
  <td width="40" height="40" nowrap bgcolor="#B568F4"> </td>
 </tr>
</table>
</body></html>
```

9.5. Создание цветовой модели сайта

Теперь мы знаем о цветах вполне достаточно, чтобы эффективно работать с ними на своих страницах. Осталось обсудить вопрос, *как именно* это делать.

Цветовая гамма — второй после графики элемент сайта по силе воздействия на посетителя. А возможно, и первый, если у пользователя отключен показ рисунков или он не дождался их загрузки. Поэтому цветовое оформление — своего рода *одежда* вашего сайта, по которой его встретят. Страница, оформленная вызывающе пестро и броско, едва ли создаст впечатление об ее владельце, как о серьезном человеке. С другой стороны, скучный черно-белый сайт также не добавит вам популярности.

Как и во всем, следует искать истину посередине.

Если вы не уверены, что создадите нечто оригинальное — лучше оставить все как есть. Цвета, используемые популярными браузерами по умолчанию плюс один-два стандартных цвета для выделения блоков на сайте — решение, приемлемое почти всегда. Однако опыт показывает обратное — почти каждый начинающий Web-мастер стремится поразить посетителей неповторимой цветовой палитрой.

Что ж, эта книга — не о дизайне, а право выражать свой вкус имеет каждый. По подбору цветов существует масса рекомендаций в литературе и Интернете, есть даже целые книги по Web-дизайну, где значительная часть содержимого посвящена этому действительно важному вопросу. Поэтому я ограничусь лишь общими рекомендациями:

- большинство пользователей привыкло читать текст и лучше его воспринимает на белом или хотя бы *светлом* фоне;
- выделение отдельными цветами посещенных и непосещенных ссылок — не каприз разработчиков HTML, а необходимый элемент страницы, позволяющий посетителю лучше ориентироваться на сайте;
- сначала разработайте общий цветовой *стиль*, а уже затем реализуйте его в основном цветовом наборе документа, заголовках таблиц, рамках и шрифтовых выделениях;
- если ваш стиль требует более пяти-шести различных цветов, возможно, стоит поработать над ним еще;
- сохраните несколько страниц, цветовая гамма которых вам нравится, и старайтесь подражать им в собственных работах;
- сохраните также несколько страниц, цветовое оформление которых вызвало у вас отвращение, — думаю, понятно для чего;
- обратитесь за советом к опытным пользователям и просто к людям с хорошим вкусом, ведь увлекшись подбором цветов (и не только их), легко потерять чувство меры;
- найдя яркий и "кричащий" цвет, который вам понравился, уменьшайте его интенсивность раза в два-три. Обычно после этого цвет готов к использованию на странице;
- не следует забывать о "безопасной палитре" и ограниченном наборе стандартных для всех браузеров цветов.

Спроектировать основной цветовой набор сайта вы сможете, запустив пример с именем index.html из папки Glava_8\Editors на компакт-диске. Подбираемые цвета при этом ограничены безопасной палитрой.

Неплохо подойдет для работы с цветами и стандартная программа Paint, входящая в состав всех версий Windows. Выбрав из меню **Палитра** пункт **Изменить палитру…** или просто дважды щелкнув мышью по любому цвету, показанному на панели инструментов **Палитра**, нажмите в появившемся окне диалога кнопку **Определить цвет >>**. Новый цвет можно задавать как мышью, так и указанием интенсивностей красного, зеленого и синего компонентов в соответствующих полях ввода. Нажав кнопку **Добавить в набор**, попробуйте рисовать новым цветом в сочетании с другими цветами вашего набора. Вертикальный графический "бегунок" в правой части окна изменения палитры позволяет также легко управлять интенсивностью цвета. Неудобство программы Paint в том, что она показывает только десятичные коды цветов. Более мощные графические редакторы, такие как Adobe Photoshop, позволяют видеть в окне определения цвета также шестнадцатеричные коды, пригодные непосредственно для HTML-документа.

Глава 10

Графика

Если бы Web-страницы содержали только текст, ссылки и теги для структурирования данных, WWW так и остался бы узкоспециализированным средством для обмена научно-технической информацией. Своей сегодняшней популярностью и "вездесущестью" Интернет во многом обязан *графике*. Анимационные и обычные картинки, фотографии, видеоклипы и другие мультимедийные объекты составляют значительную долю от объема информации, доступной посредством Интернета.

В этой главе мы научимся добавлять на свои страницы графическое содержимое, а также рассмотрим основные принципы и области применения компьютерной графики.

10.1. Общие принципы работы с графикой

Вы наверняка знаете, что любое изображение на экране монитора строится из отдельных экранных точек или *пикселов*. Каждый пиксел можно подсветить одним из цветов, доступных в текущем *видеорежиме*. Видеорежим определяется *разрешающей способностью* монитора, т. е. числом пикселов по горизонтали и вертикали, *тактовой частотой*, с которой обновляется изображение на экране, и количеством *поддерживемых цветов*. Разумется, чем выше все эти показатели, тем мощнее и современнее должны быть сам монитор и установленная в системном блоке видеокарта. В настоящее время мониторы способны поддерживать десятки видеорежимов, и все они отличаются между собой этими тремя характеристиками.

Из *разд. 9.1* мы узнали, как строится код цвета по интенсивностям красного, зеленого и синего цветовых компонентов. Поскольку каждое изображение на экране представляет собой всего лишь набор разноцветных пикселов, любой графический файл должен либо *содержать* информацию о том, какой пиксел

каким цветом подсветить, либо давать возможность *получить* ее из хранящихся в графическом файле данных. По сути дела, мы уже сказали о двух главных видах графики:

- *растровая* (или *точечная*) — основана на том, что в графическом файле хранится информация о цвете каждого пискела изображения;
- *векторная* графика — графический файл содержит *математическую модель* рисунка, состоящую из простых *геометрических примитивов*, таких как отрезки прямых (*вектора*). При этом в файле хранится только информация об *относительных* размерах этих векторов. Когда нужно вывести векторный рисунок на экран или на печать, математические отношения длин векторов пересчитываются в пикселы или другие абсолютные величины, в зависимости от требуемых размеров изображения.

Немного подумав, легко оценить преимущества и недостатки каждого из этих видов графики.

Начнем с векторной. Ее основной плюс — *качество рисунка практически не зависит от его размеров*. При каждом изменении размера модель просто строится заново, с учетом всех взаимных соотношений векторов. Разумеется, неумеренное масштабирование способно испортить внешний вид векторного рисунка, но на самом файле это никак не отразится. Платой за столь похвальные качества и основными недостатками векторной графики являются *ограниченные выразительные средства* векторных редакторов и *сложность ее разработки*. В самом деле, логически спроектировать состоящий из безразмерных векторов объект значительно сложнее, чем нарисовать его "кистью и пером".

Редактор схем в Microsoft Office и поддерживаемый им графический формат WMF (Windows MetaFile) — один из наглядных примеров векторной графики. Значительно более сложными программными средствами для работы с ней являются системы автоматизированного проектирования (CAD), программы создания векторных шрифтов, профессиональные векторные редакторы, такие как CorelDRAW (формат файлов CDR), Abode Illustrator (формат AI) и некоторые другие.

Растровую графику стоит обсудить подробней, т. к. современные браузеры только ее и поддерживают, а в стандартах HTML ничего не сказано о возможности работы с векторными рисунками. Поэтому нам следует хорошо представлять себе виды растровой графики, их достоинства и недостатки.

Основное преимущество растровой графики по сравнению с векторной — *простота ее хранения и кодирования*. Действительно, для вывода растрового рисунка достаточно получить данные о каждой точке и отобразить их на экране. Как мы увидим далее, простота эта во многом кажущаяся, однако

по сравнению с векторной графикой растровая действительно несложна. Кроме того, работая с изображением на уровне отдельных пикселов, растровый графический редактор способен обеспечить сколь угодно сложные инструменты рисования при сохранении высокого качества представления изображения. Отведя по три байта памяти на один экранный пиксел, можно не беспокоиться, что цвет какой-либо точки куда-то пропадет. Кстати, о "трех байтах на пиксел". Много это или мало? Несложный подсчет показывает, что изображение размером 200×100 пикселов, каждая точка которого требует по 3 байта дискового пространства, займет на жестком диске 200 × 100 × 3 = 60 000 байтов или около 60 Кбайт. Примерно такой же объем дисковой или оперативной памяти занимают 20–25 страниц текста. А ведь 200×100 пикселов — это лишь крошечный уголок вашего экрана.

Теперь давайте учтем, что Интернет — очень *медленная* среда по сравнению с вашим быстродействующим компьютером или даже с локальной сетью. При скорости передачи данных, равной 10 Кбайт/с (а это не так уж плохо), растровая картинка размером 1024×768 пискелов (популярное разрешение экрана), отводящая по 3 байта памяти на пиксел, займет около 2300 Кбайт и будет закачиваться 230 с. "Странно — скажете вы, — мне удавалось закачивать обои для Рабочего стола значительно быстрее". Согласен. Только обои эти были *сжатой* графикой. Чуть позже мы обсудим и это, а пока акцентируем внимание еще на одном негативном свойстве точечных рисунков: *при любом изменении размеров растрового рисунка качество изображения только ухудшается*. Представим, что нам захотелось изменить размер нашего рисунка с 200×100 до 400×200 пискелов. Каждый пискел при этом "превратился" в четыре пиксела того же цвета, ведь рисунок увеличился вдвое по горизонтали и вертикали, а значит, все его точки нужно пропорционально "растянуть" до нового размера! Когда изображение представляет собой однотонный прямоугольник, никаких видимых изменений не происходит. Если же имеется тонкая и сложная прорисовка деталей, то они начинают искажаться угловатыми "ступеньками". Подобный ступенчатый эффект наблюдал каждый, кто сильно увеличивал размер картинки в своем любимом графическом редакторе. В нашем примере при увеличении размера ровно в 2 раза ничего страшного с высококачественным рисунком не произойдет. Если же изменения размеров многократны, *непропорциональны* и выполнялись *произвольное число раз*, видимые результаты могут быть просто ужасными. Взгляните на два несложных изображения кошки, приведенных на рис. 10.1. Правое получено однократным уменьшением, а затем увеличением левого изображения вдвое. Даже при этой простой операции потери качества заметны невооруженным глазом. Их и не могло не быть, ведь операция двукратного уменьшения размера исключила информацию о цвете сразу половины пикселов!

Рис. 10.1. Растровый рисунок: слева — исходный; справа — после уменьшения и увеличения вдвое

Классифицировать графику можно не только по типу *кодирования* графической информации, но и по способу ее *хранения*.

Несжатая графика, пример которой представляет всем знакомый формат BMP, полностью и без искажений сохраняет данные о цветовом кодировании каждого пиксела.

Основные преимущества такого подхода — простота обработки графики и высокая скорость ее вывода, основной недостаток — очень большой размер файла при большом количестве используемых цветов. Поэтому несжатая графика подходит для *предварительной обработки* графических файлов или *создания* сложных изображений, но ни в коем случае не для размещения в Интернете, хотя большинство современных браузеров поддерживает формат BMP.

Сжатая графика, к которой относятся распространенные форматы GIF и JPEG, хранит рисунок в виде *архива*, размер которого меньше исходного несжатого изображения. Архивирование выполняется за счет сложных математических алгоритмов, которые, с учетом частоты встречаемости отдельных символов в исходном файле, преобразуют его таким образом, что более распространенные символы кодируются меньшим числом битов. По этому же принципу работают с файлами архиваторы общего назначения, такие как WinZip и WinRAR. Основной плюс подхода очевиден — существенно уменьшается размер рисунка, передаваемого по сети. Очевиден и минус — распаковка требует времени.

Сжатая с потерей качества графика отличается тем, что при упаковке графического файла исключается информация о несущественных или мало различимых глазом цветовых отличиях между точками рисунка. Это дает *дополнительное уменьшение размера*, за счет которого, например, картинка в популярном формате JPEG может быть в 10—20 раз меньше, чем ее оригинал в BMP, но ценой экономии дискового пространства и сетевого трафика оказывается *необратимое ухудшение качества*, присущее этим форматам. Поэтому сжатая с потерей качества графика используется, как правило, для раз-

мещения предварительно обработанных несжатых рисунков на сайте, а преобразование из несжатого формата в сжатый должно выполняться лишь *однократно*.

В *разд. 10.2* мы поговорим о форматах файлов подробнее, а пока продолжим знакомство с теорией. Третий способ классификации графики — по *количеству представляемых цветов*, которое зависит от так называемой *глубины цвета*, т. е. числа битов или байтов, выделяемых для хранения информации о цвете одного пиксела. В табл. 10.1 кратко описаны основные виды графики с точки зрения качества цветопередачи.

Таблица 10.1. *Виды графики по качеству цветопередачи*

Число битов на точку (глубина цвета)	**Видеорежимы и область применения**
1	Монохромная графика (черно-белая)
2	Стандартные 4-цветные режимы видеоадаптеров CGA
4	Стандартные 16-цветные режимы видеоадаптеров EGA и VGA
8	256-цветные режимы видеоадаптеров VGA; базовые режимы SuperVGA
16	Режимы High color (65 536 цветов)
24	Режимы Real color (16,7 млн. цветов)
32	Режимы True color (более 4 млрд. цветов)

В режимах *младше* High color вместе с изображением может быть сохранена *палитра* — таблица данных с информацией о кодах использованных цветов. В старших режимах палитры нет, и один код всегда обозначает один цвет.

Не нужно думать, что режимы младше High Color безнадежно устарели. Разумеется, они непригодны для создания полноцветных фотографий и сложных мультимедийных композиций, но 16- или 256-цветная графика отлично подходит для создания несложных анимаций, рекламных баннеров, графических кнопок, графического маркера списка или пункта меню. Основное преимущество "малоцветных" файлов — небольшой размер, благодаря которому обеспечивается высокая скорость загрузки. А грамотный подбор палитры может компенсировать небольшое количество используемых цветов.

В заключение теоретического раздела обсудим общие правила, о которых не следует забывать, работая с графикой:

- графика должна дополнять вашу информацию, а не скрывать ее отсутствие;

- ключевой фактор для графики — время загрузки. Оно должно быть как можно меньше, а это напрямую зависит от размеров графических файлов. Если на странице находится несколько графических файлов с размерами по 50—100 Кбайт каждый, мало кто оценит их красоты. В *разд. 10.6* мы кратко обсудим более "экономичную" технологию создания коллекций картинок;
- если графика входит в структуру сайта, например, рисунки располагаются в ячейках таблиц, следует указывать их точные размеры. В противном случае загрузка страницы может быть значительно замедлена, ее структура искажена, а части страницы начнут некрасиво "прыгать" в процессе загрузки;
- значения, указанные в атрибутах `width` и `height` тега `<img>`, должны соответствовать реальным размерам рисунка в пикселах. Единственное допустимое, но нежелательное исключение — уменьшение или увеличение изображения в 2, 4 или 8 раз;
- при подготовке иллюстраций для Web-страницы всю предварительную обработку нужно проводить с несжатыми изображениями, например, в формате BMP, а в GIF или JPEG конвертировать только окончательный вариант картинки;
- всегда следует помнить, что отображение графики в браузере пользователя может быть отключено или вовсе отсутствовать. Поэтому необходимо пользоваться атрибутами для создания альтернативных рисунку текстовых подписей, а цветовая модель вашего сайта должна корректно выглядеть как при включенной, так и при отключенной графике;
- не спешите указывать в информационном блоке страницы, что она "оптимизирована для браузера Internet Explorer и разрешения 1024×768". Это производит впечатление не предупредительности, а просто непрофессионализма разработчика. Никто не станет менять разрешение монитора и устанавливать новый браузер, чтобы насладиться прелестями вашего дизайна. Ведь в Интернете существует множество других страниц, не предъявляющих к пользователю столь необоснованных претензий.

10.2. Графические форматы Интернета

В зависимости от способа описания информации в графическом файле, различают множество *форматов* графики.

Сегодня как программами для просмотра рисунков, так и различными специализированными приложениями используется не одна сотня этих форматов.

Однако в стандарте HTML в качестве "широко распознаваемых" упомянуты только GIF, JPEG и PNG.

Формат GIF (Graphics Interchange Format) характеризуется следующими особенностями:

- растровая графика, сжатая без потерь качества;
- число поддерживаемых цветов варьируется от 2 до 256;
- коэффициент сжатия данных по отношению к несжатому рисунку-аналогу находится в пределах от 3 до 5.

Формат GIF поддерживает три эффекта, полезные при создании Web-страниц:

- *чересстрочная загрузка* (interlaced images) — сначала отображается каждая восьмая строка образа, затем каждая четвертая и т. д. В результате, по мере загрузки графического файла браузером, изображение "проявляется" на экране в несколько проходов. Это может быть полезно пользователям, работающим на медленных линиях связи, а также "торопыгам", стремящимся поскорее узнать, что изображено на рисунке, не дожидаясь его полной загрузки;
- *прозрачность* (transparency) — один из цветов обозначается как *прозрачный*, что позволяет фону документа проступать сквозь него. В результате рисунок отображается не прямоугольником, а произвольным контуром, наложенным на фон документа;
- *анимация* (animation) — анимированный файл GIF содержит несколько изображений, которые отображаются браузером, последовательно сменяя друг друга, а также информацию о промежутках времени между сменой кадров и о числе повторений цикла их прокрутки.

Запустив из папки на компакт-диске Glava_10\GifEffects единственный имеющийся там документ index.html, вы можете увидеть эти эффекты формата в действии.

В формате GIF *не следует* хранить полноцветные фотографии и рисунки.

GIF подходит для хранения следующих классов изображений:

- чертежи и рисунки, другие изображения, состоящие из относительно небольшого числа линий и цветов, например, значки, эмблемы, графические кнопки для навигации по сайту, графические маркеры списков и разделители блоков сайта;
- изображения с небольшим количеством однородных цветов. Наилучшее сжатие средствами GIF достигается тогда, когда число цветов в составе изображения невелико, а поверхности закрашены равномерно,

т. е. не применяется градиентное закрашивание, а также не имитируются тени и блики;

- любые изображения с прозрачным фоном, графические навигационные кнопки и другие "украшения" страницы;
- анимация, как правило, предназначенная для создания рекламных заставок-баннеров, небольших "мультфильмов" и т. п.

Для формата JPEG (Joint Photographic Expert Group) характерно следующее:

- растровая графика, сжатая *с потерей качества*;
- параметр "качества JPEG" (число от 0 до 100), как правило, может выбираться пользователем при сохранении файла в этом формате. Одинаковые значения параметра могут по-разному интерпретироваться различными программами, большинство приложений воспринимает их как величину, заданную в процентах. Уменьшение значения параметра приводит к необратимому ухудшению качества рисунка начиная с 20—30. Удовлетворительным в большинстве случаев следует считать однократное сохранение несжатого рисунка в формат JPEG с указанием качества 65—90;
- формат предназначен для хранения полноцветных изображений, в режиме 24 бита на пиксел;
- коэффициент сжатия данных по отношению к несжатому рисунку-аналогу варьируется в зависимости от величины параметра качества; следует считать удовлетворительным сжатие от 4—5 до 15 раз по отношению к исходному рисунку.

Формат *не следует* использовать для решения следующих задач:

- хранение изображений, уже приведенных к 256-цветной или иной палитре младше High color;
- сжатие изображений, содержащих четкие переходы между цветами. Алгоритм JPEG плохо передает резкие границы.

Формат JPEG *предназначен* для работы со следующими классами изображений:

- полноцветные цифровые фотографии и сложные изображения с большим количеством цветов;
- графические данные, в которых применяется градиентная заливка и отслеживается расположение источника освещения;
- фотоснимки, на которых присутствует большое количество оттенков одного цвета, например, снимок неба во время заката, морские пейзажи и т. д.

Своеобразное "развитие" файлов GIF — *прогрессивные* JPEG-файлы, которые также отображаются с постепенной прорисовкой.

Прозрачность в форматах JPEG не поддерживается.

Пример с компакт-диска, записанный в папке Glava_10\JpegQualities под именем index.html, показывает изменение внешнего вида и размера файла одного и того же изображения в зависимости от параметра качества, установленного при сжатии. Обработка исходного несжатого рисунка выполнялась в программе Adobe Photoshop CS2.

Формат PNG (Portable Network Graphics) — новый графический стандарт, разработанный W3C и призванный совместить достоинства GIF и JPEG, он поддерживается всеми современными браузерами, но пока не слишком широко используется разработчиками.

Особенности формата PNG следующие:

- более высокая степень сжатия изображения, чем у GIF;
- поддержка 1-, 2-, 4-, 8-, 16-, 24- и 32-битовых изображений;
- черезстрочный показ изображения;
- несколько уровней прозрачности;
- встроенная гамма-коррекция, позволяющая более точно определять оптимальную яркость изображения.

Проверить, какие графические форматы поддерживает ваш браузер, вы можете, запустив пример Glava_10\BrowserTest\index.html с компакт-диска. Сразу же следует сказать, что поддержка форматов WMBP и PCX браузером маловероятна, а остальные 5 форматов, скорее всего, будут отображены. Заглянув во вложенную папку img, вы сможете сравнить свойства и размер одного и того же рисунка, сохраненного в различных форматах.

10.3. Вставляем картинки или все о теге <*img*>

Тег `<img>` (image) связывает с документом HTML *графический образ*, находящийся во внешнем файле. Он может содержаться в любых текстовых и блочных элементах, кроме <pre>. В простейшем случае, тег имеет вид:

```
<img src="URL">
```

Здесь обязательный атрибут `src` (от англ. "source" — источник) указывает абсолютный или относительный URL-адрес файла, в котором хранится изображение. Этот файл должен быть в одном из трех форматов, поддерживаемых браузерами (GIF, JPEG или PNG). Если рисунок не найден или не загружен из-за ошибок связи, на его месте выведется пустая рамка.

Поскольку тег `<img>` просто определяет место вставки рисунка в документ, закрывающая часть ему не нужна и даже запрещена спецификацией.

Все остальные атрибуты формально не обязательны, но указание атрибутов `alt`, `width` и `height` настоятельно *рекомендуется* для всех рисунков, кроме чисто "декоративных" или служащих фоном.

Атрибут `alt` задает *альтернативный (замещающий) текст*, который выводится браузером вместо рисунка, если вывод графических изображений отключен или не поддерживается.

Браузер Internet Explorer дополнительно выводит содержимое атрибута `alt` в виде всплывающей подсказки при наведении курсора мыши на рисунок. Такое поведение браузера не соответствует спецификациям, и большинство других обозревателей отображают содержимое `alt` на месте рисунка, только если графика отключена. Напомним, что для создания всплывающей подсказки существует общий для всех элементов атрибут `title` (см. *разд. 4.2*). Содержимое атрибута `alt` индексируется современными поисковыми машинами, что служит еще одним аргументом в пользу его применения. С другой стороны, нет гарантий, что браузер при отключенной графике "догадается" разместить альтернативный текст так, чтобы он не испортил структуру страницы. Поэтому замещающий текст должен быть коротким и пояснять только суть рисунка.

Браузеры могут отображать текст из атрибута `alt` *вместо* рисунка в процессе его загрузки.

Спецификацией предусмотрен еще один "описательный" атрибут `longdesc="URL"`, содержащий ссылку на подробное описание рисунка, дополняющее краткие сведения, заданные атрибутом `alt`. Ни один браузер этот атрибут пока не поддерживает.

Атрибуты `width="размер"` и `height="размер"` задают размеры рисунка на экране по горизонтали и вертикали соответственно.

Эти атрибуты наиболее полезны, когда они указывают точные размеры рисунка в пикселах. Их задание позволяет браузеру в процессе загрузки документа сразу же зарезервировать на странице место для рисунка, что несколько ускорит его отображение и позволит избежать некрасивых "скачков" содержимого при загрузке.

Многие браузеры выделяют пространство для размещения рисунка на основе атрибутов `width` и `height` даже тогда, когда они отображают не сам рисунок, а альтернативный текст. Как уже указывалось, это может привести к тому, что этот текст будет виден не полностью.

Из *разд. 8.2* мы уже знаем, что значение типа данных "`размер`" можно указывать как в пикселах, так и в процентах. Опыт показывает, что для рисунка

всегда предпочтительнее задание размеров в пикселах, так как результаты масштабирования графики браузером редко бывают удовлетворительными.

Если указывается только один из атрибутов `width` и `height`, недостающий размер вычисляется автоматически исходя из сохранения пропорций рисунка. Однако это занимает дополнительное время и не рекомендуется.

Уже неоднократно применявшийся нами атрибут `align` допустим и при работе с рисунками. Он определяет выравнивание образа относительно окружающего его содержимого документа. Возможные значения атрибута приведены в табл. 10.2.

Таблица 10.2. *Значения атрибута align тега <img>*

Значение атрибута	Описание
`align="left"`	Рисунок прикрепляется к текущему левому полю документа. Последующий текст обтекает изображение справа
`align="right"`	Рисунок прикрепляется к текущему правому полю документа. Последующий текст обтекает изображение слева
`align="bottom"`	Нижняя граница изображения выравнивается по *базовой линии* текущей строки. Это значение установлено по умолчанию
`align="middle"`	Середина изображения выравнивается по базовой линии текущей строки
`align="top"`	Верх изображения выравнивается по самому высокому элементу текущей строки
`align="texttop"`	Верх изображения выравнивается по самому высокому *текстовому* элементу текущей строки
`align="absmiddle"`	Середина изображения выравнивается по *середине* текущей строки
`align="absbottom"`	Нижняя граница изображения выравнивается по самым нижним элементам букв текущей строки
`align="baseline "`	Используется некоторыми браузерами как `bottom`

Существует важное различие между парой значений `left` и `right` и всеми остальными. Оно состоит в том, что `left` и `right` делают графический объект *плавающим*, выравнивая его относительно левого или правого края документа. При этом браузер может автоматически добавлять перевод строки перед выводом рисунка. Указание всех остальных значений относится к вертикальному выравниванию рисунка относительно окружающего текста. При этом поведение рисунка соответствует положению "в тексте" рисунка, вставленного в документ Microsoft Word.

Базовая линия для текста проходит по нижней части строки, на уровне символа точки или запятой. При этом не учитываются фрагменты букв, выходящие вниз за строку.

При выборе значений из табл. 10.2 нужно иметь в виду следующее:

- только значения `left`, `right`, `bottom`, `middle` и `top` упомянуты в спецификации, все остальные нестандартны или нежелательны;
- браузеры могут по-разному интерпретировать атрибут `align`. Некоторые из них принимают в расчет только текстовую строку, находящуюся перед элементом, другие учитывают текст по обеим сторонам элемента;
- браузеры могут интерпретировать различные значения из табл. 10.2 одинаково, например, `absmiddle` как `middle`, `absbottom` как `bottom`.

Рисунок 10.2 и листинг 10.1 иллюстрируют использование различных значений атрибута `align`. Запустить этот пример в браузере можно, открыв файл Glava_10\ImagesAlign.html с компакт-диска.

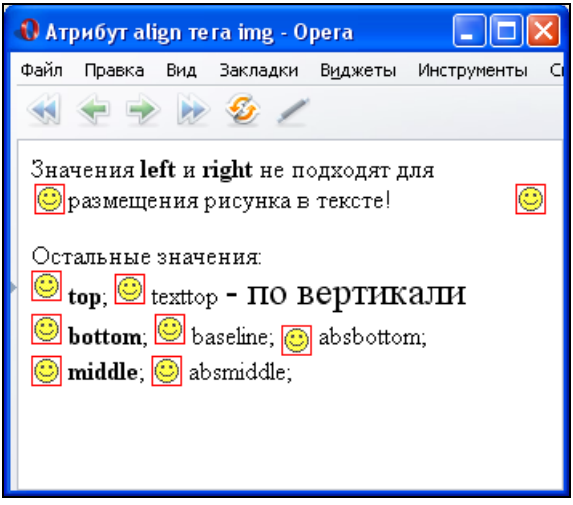

Рис. 10.2. Использование атрибута `align` тега `<img>`

Листинг 10.1. Использование атрибута `align` тега `<img>`

```
<html>
<head>
<title>Атрибут align тега img</title>
</head>
```

```
<body bgcolor="white">
<p>Значения <img src="images/smile.gif" width="19" height="19"
align="left"> <b>left</b> и <img src="images/smile.gif" width="19"
height="19" align="right"> <b>right</b> не подходят для размещения
рисунка в тексте!</p>
<p>Остальные значения:<br>
<img src="images/smile.gif" width="19" height="19" align="top">
<b>top</b>;
<img src="images/smile.gif" width="19" height="19" align="texttop"> text-
top <font size="+2"> - по вертикали</font>
<br><img src="images/smile.gif" width="19" height="19" align="bottom">
<b>bottom</b>;
<img src="images/smile.gif" width="19" height="19" align="baseline">
baseline;
<img src="images/smile.gif" width="19" height="19" align="absbottom">
absbottom;
<br><img src="images/smile.gif" width="19" height="19" align="middle">
<b>middle</b>;
<img src="images/smile.gif" width="19" height="19" align="absmiddle">
absmiddle;
</p></body></html>
```

Все основные атрибуты тега `<img>` описаны в табл. 10.3.

Таблица 10.3. *Основные атрибуты тега <img> и их значения*

Атрибут	**Описание**
`src="URL"`	Абсолютный или относительный URL-адрес рисунка. Обязательный атрибут
`alt="текст"`	Замещающий текст для рисунка (рекомендуется)
`width="размер"`	Ширина рисунка в пикселах (рекомендуется) или в процентах
`height="размер"`	Высота рисунка в пикселах (рекомендуется) или в процентах
`align`	Выравнивание рисунка относительно края документа (значения `left` и `right`); вертикальное выравнивание рисунка относительно текста (значения `top`, `middle`, `bottom`)
`border="пискелы"`	Ширина обрамления рисунка в пискелах
`hspace="пискелы"`	Отступ от изображения слева и справа
`vspace="пискелы"`	Отступ от изображения сверху и снизу
`name="текст"`	Название рисунка для использования в сценариях

Рассмотрим не описанные ранее атрибуты из табл. 10.3.

Атрибут `border` устанавливает обрамление рисунка. Его цвет соответствует цвету текстовых ссылок. Если атрибут не указан, браузер может автоматически

добавлять обрамление рисунка, вставленного в тег ссылки (см. *разд. 10.5*). Запретить обрамлять рисунок можно указанием значения атрибута `border="0"`. Для рисунков, не являющихся ссылками, атрибут не обязателен, хотя может быть полезен для визуального оформления изображений, фон которых отличается от фона страницы.

Атрибуты `hspace` и `vspace` особенно полезны, если рисунок "плавающий". Их игнорирование может привести к тому, что изображение "придвинется" вплотную к тексту.

В настоящее время вместо этих атрибутов рекомендуются *стили* (см. *главу 14*).

Атрибут `name="текст"` содержит название рисунка, на которое можно ссылаться в таблицах стилей и сценариях. В настоящее время его рекомендуется заменять атрибутом `id`, описанным в *разд. 4.2*.

В листинге 10.2 приведен код примера, включающий основные атрибуты тега `<img>`. Результат его выполнения иллюстрирует рис. 10.3. Запустить пример с компакт-диска вы можете, открыв файл Glava_10\Images1.html.

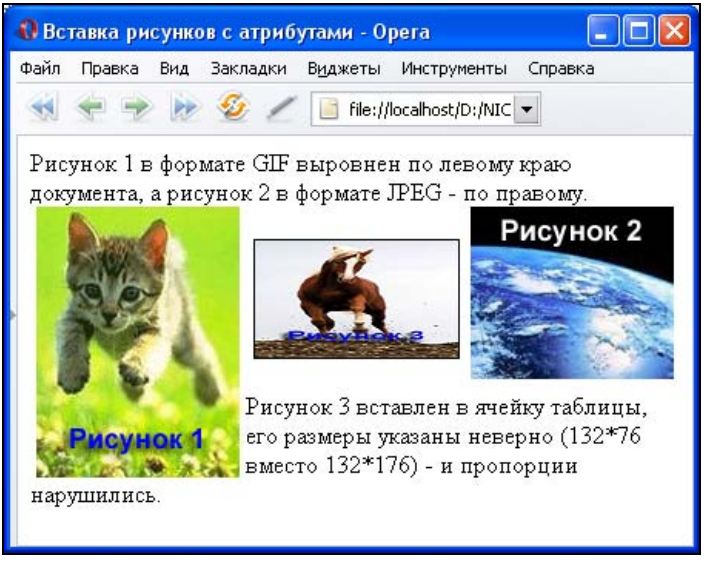

Рис. 10.3. Использование основных атрибутов тега `<img>`

Листинг 10.2. Использование основных атрибутов тега `<img>`

```
<html>
<head>
  <title>Вставка рисунков с атрибутами</title>
```

```
</head>
<body>
<p>Рисунок 1 в формате GIF выровнен по левому краю документа,
а рисунок 2 в формате JPEG - по правому.
<img src="images/ris01.gif" width="132" height="176" align="left"
hspace="4" vspace="0" alt="Рисунок 1">
<img src="images/ris02.jpg" width="132" height="112" align="right"
hspace="4" vspace="0" alt="Рисунок 2">
</p>
<table align="center" border="0">
<tr><td>
 <img src="images/ris03.png" width="132" height="76" align="center" bor-
der="1" hspace="2" vspace="2" alt="Рисунок 3">
</td></tr></table>
<p>Рисунок 3 вставлен в ячейку таблицы, его размеры указаны неверно
(132*76 вместо 132*76) - и пропорции нарушились.
</body></html>
```

Неверно указанный размер рисунка 3 — не главная ошибка в этом документе. Хуже другое — "незакрепленные" рисунки 1 и 2, выровненные по левому и правому краю страницы, были выведены браузером после текста абзаца и заняли часть пространства, в котором формировалось последующее содержимое.

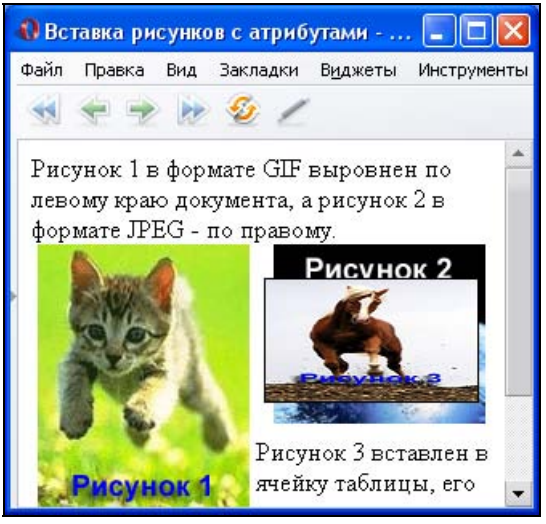

Рис. 10.4. Некорректное наложение рисунков при уменьшении размеров окна

Выровненная по центру таблица с рисунком 3 заняла место между рисунками 1 и 2 и, независимо от ее атрибутов, не будет учитывать наличия этих рисунков на странице. При дальнейшем уменьшении размеров окна изображения могут даже наложиться друг на друга, как показано на рис. 10.4.

Можно сделать вывод, что в отношении рисунков следует с осторожностью использовать атрибуты `align="left"` и `align="right"`, делающие объект "плавающим".

Кроме того, помещение рисунка "в текст" в HTML вообще ненадежно, по возможности его следует избегать, размещая рисунки в ячейках таблиц, структурирующих страницу.

10.4. Использование фоновых рисунков

Фоновое изображение на Web-странице сродни хорошо всем знакомым обоям на Рабочем столе Windows. Не случайно фоновый рисунок обычно называют "обоями" или "текстурой" страницы.

Обои всегда заполняют *все окно* браузера. В настройках Рабочего стола Windows этому варианту соответствует опция **Замостить**. Соответственно, имеет смысл создавать лишь такие обои, цвета граничных пикселов которых совпадают. Существует множество условно-бесплатных программ, предназначенных для этой цели, лучшие из них вполне профессиональны. Так, программа Buttons & Tiles позволяет создавать и текстуры, и графические кнопки, поддерживает сложные алгоритмы генерации узоров и различные эффекты.

Для вставки фонового изображения на Web-страницу существует только один документированный путь — атрибут `background="URL"` тега `<body>`. Указанный URL-адрес может быть как относительным, так и абсолютным. Ни размер, ни другие атрибуты тега `<img>` для фоновых изображений указывать нельзя. Современные стандарты рекомендуют заменять этот атрибут стилевым указанием `background-image: url("URL")`, применимым *ко всем* элементам.

HTML-код страницы, содержащей 4 фоновых рисунка (для всей страницы, таблицы, строки и отдельной ячейки таблицы), приведен в листинге 10.3, а ее внешний вид — на рис. 10.5. Задание атрибута `background` в тегах `<table>`, `<tr>` и `<td>` не является стандартным, но поддерживается всеми современными браузерами.

На компакт-диске этому примеру соответствует файл Glava_10\BackGround1.html.

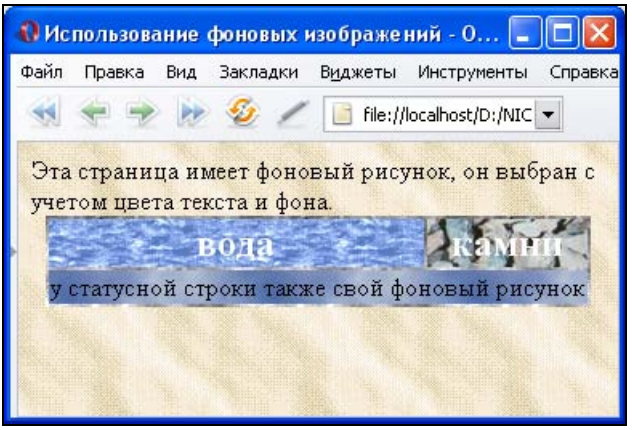

Рис. 10.5. Использование фоновых рисунков

Листинг 10.3. Использование фоновых рисунков

```
<html>
<head>
<title>Использование фоновых изображений</title>
</head>
<body text="black" bgcolor="#F7E7CE" link="blue" vlink="magenta" back-
ground="images/back1.gif">
<p>Эта страница имеет фоновый рисунок, он выбран с учетом
цвета текста и фона.
<table width="95%" align="center" background="images/back2.gif">
<tr align="center">
 <td width="70%" background="images/back3.gif"><font color="white"
size="+2"><b>вода</b></font>
 </td>
 <td width="30%"background="images/back4.jpg"><font color="white"
size="+2"><b>камни</b></font>
 </td>
</tr>
<tr background="images/back5.gif">
 <td colspan="2">у статусной строки также свой фоновый рисунок</td>
</tr>
</table></body></html>
```

Плавные переходы цвета на странице — *градиенты* — часто создают с помощью узкого и длинного фонового рисунка, содержащего нужный эффект.

Посмотреть подобный пример вы можете, запустив файл Glava_10\BackGround2.html с компакт-диска. При реализации этого способа нужно учесть, что если цвета левого и правого краев градиентного рисунка не совпадают, это может испортить внешний вид страницы при горизонтальном разрешении монитора, большем, чем длина фонового рисунка. А разрешение мониторов сегодня достигает уже 1600×1200 пикселов. Градиенты легко и удобно создавать в программе Adobe Photoshop.

При использовании на странице любых фоновых рисунков нужно учесть следующее:

- как и любая графика, фоновая картинка имеет больше шансов загрузиться и быть оцененной, если она будет компактной по размеру. В большинстве случаев для создания фоновых изображений-текстур достаточно формата GIF;
- яркий фон часто выглядит навязчивым и мешает восприятию текста;
- цвета фонового изображения должны составлять контраст по отношению к основному тексту и ссылкам;
- темный фон для больших блоков страницы нежелателен;
- применение фонового изображения не избавляет от необходимости указания фонового цвета атрибутом `bgcolor="цвет"`; скорее, наоборот, выбор фонового цвета, близкого к цвету обоев, поможет увидеть приемлемый вариант страницы и тем пользователям, у которых отключен вывод графики в браузере. Кроме того, такой подход позволит избежать резких "скачков" цвета по мере загрузки страницы.

10.5. Графические средства навигации

Кроме текста, в качестве ссылок можно использовать и рисунки. Изображение в этом случае достаточно разместить в теге `<a href=URL>…</a>`:

```
<a href="document.html"><img src="link.gif" width="100" height="24"></a>.
```

Здесь рисунок с именем link.gif, находящийся во вложенной по отношению к текущей папке images, служит ссылкой на файл document.html из текущей папки.

Атрибут `href` тега `<a>` задает относительный или абсолютный путь к документу, на который указывает ссылка, а атрибут `src` тега `<img>` — путь к графическому файлу.

Вокруг изображения-ссылки автоматически добавляется рамка толщиной 2 пиксела с тем же цветом, что у текстовых ссылок. Чтобы убрать рамку, следует у тега `<img>` установить атрибут `border="0"`:

```
<a href="document.html" target="_blank"><img src="link.gif" width="100"
height="24" border="0"></a>.
```

Здесь картинка с именем link.gif лишена рамки, а файл document.html открывается в новом окне.

Чтобы убрать рамку для всех изображений, которые являются ссылками, можно также воспользоваться стилями. Для этого применяется стилевой параметр `border` со значением `none`.

Как правило, ссылки в виде рисунков или фото целесообразны в следующих случаях:

- навигация по сайту с помощью графических кнопок;
- организация коллекций картинок и фотогалерей; при этом быстро загружаемая уменьшенная копия картинки служит ссылкой на основную картинку;
- указание логотипов и эмблем в качестве ссылок на сайт производителя программного обеспечения, рекламодателя или иной организации, например:

```
<a href="http://www.microsoft.com/ie" target="_blank">
<img src="logos/ie.gif" width="88" height="31"></a>;
```

- обмен графическими кнопками сайтов и рекламными блоками.

Первые два вопроса будут рассмотрены в этой главе, а остальные кратко описаны в *разд. 15.5*.

Графическая навигация по сайту обладает рядом неоспоримых достоинств:

- надписи на графических кнопках будут прочитаны пользователем независимо от выбранных языка, шрифта и кодировки;
- независимо от настроек браузера навигационный блок будет иметь точно известные размеры в пикселах;
- при создании текстовых элементов средствами графики возможны любые шрифты и эффекты.

Эти преимущества оттеняются не менее вескими недостатками:

- время загрузки страницы и трафик существенно увеличиваются;
- меню с трудом поддается модификации, требующей рисования новых картинок.

Компромиссом может быть работа с профессиональными графическими редакторами, создающими оптимизированные и компактные файлы формата GIF, такими как Gif Animator и Adobe Photoshop.

Код несложного графического меню, созданного с помощью набора картинок размером 120×24 пиксела, приведен в листинге 10.4, а внешний вид меню иллюстрирует рис. 10.6. Короткие альтернативные подписи, сделанные с помощью атрибута `alt`, обеспечат правильное функционирование этого меню

для тех пользователей, у которых отключена графика. На компакт-диске этот пример записан под именем Glava_10\GraphMenu.html.

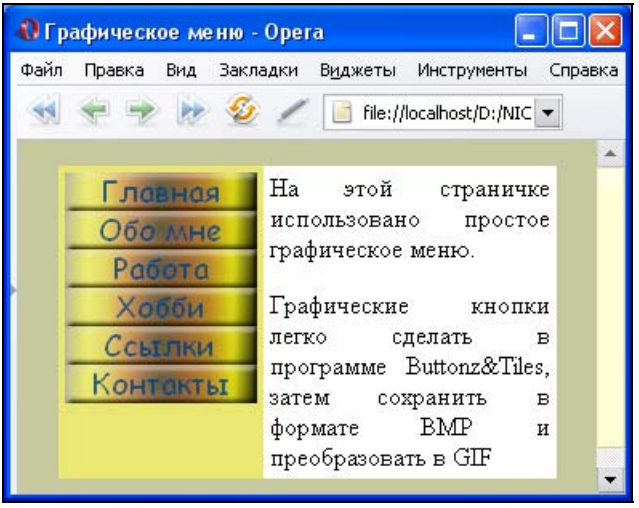

Рис. 10.6. Графическое меню

Листинг 10.4. Графическое меню

```
<html>
<head>
  <title>Графическое меню</title>
</head>
<body text="black" bgcolor="#C6C89E" link="blue" vlink="magenta">
<!-- документ будет содержать 2 столбца -->
<table width="95%" align="center" border="0" cellpadding="0" cellspacing="8">
<tr><td>
 <table width="100%" align="center" border="0" cellpadding="4" cellspacing="0">
  <tr>
   <td width="120" align="center" valign="top" bgcolor="#EAE872">
    <!-- графическое меню-->
     <a href="index.html"><img src="images/button1.gif" width="120"
      height="24" border="0" alt="Главная"></a>
     <a href="about.html"><img src="images/button2.gif" width="120"
      height="24" border="0" alt="Обо мне">
     <a href="work.html"><img src="images/button3.gif" width="120"
```

```
      height="24" border="0" alt="Работа">
     <a href="hobby.html"><img src="images/button4.gif" width="120"
      height="24" border="0" alt="Хобби">
     <a href="links.html"><img src="images/button5.gif" width="120"
      height="24" border="0" alt="Ссылки">
     <a href="contact.html"><img src="images/button6.gif"
      width="120" height="24" border="0" alt="Контакты">
   </td>
   <td align="justify" valign="top" bgcolor="white">
    <p>На этой страничке использовано простое графическое меню.
    <p>Графические кнопки легко сделать в программе Buttonz&Tiles,
     затем сохранить в формате BMP и преобразовать в GIF</p>
   </td>
  </tr>
 </table>
</td></tr></table>
</body></html>
```

Создание сложных меню значительно более трудоемко, помочь здесь может подключение апплетов на языке Java и скриптов на JavaScript, рассмотренное в *главе 15*.

Для организации графической навигации по сайту применяются также картированные изображения, описанные в *главе 11*.

10.6. Организация коллекций картинок

Возможно, вы обратили внимание, что рисунки почти всех примеров главы загружаются из папки images, вложенной в папку Glava_10. Это сделано не только для удобства, но и помогает браузеру *кэшировать* файлы, т. е. не загружать их повторно без необходимости. Разумеется, папка для рисунков не обязана быть единственной и называться images. Сейчас мы поговорим о том, как организовать хранение картинок, если их много.

Качество картинки прямо пропорционально ее размеру. Нам одинаково не хотелось бы ни жертвовать красотой рисунков, конвертируя их в JPEG низкого качества или уменьшая размеры, ни перегружать страницу фотоальбома "тяжелой" графикой, загрузка которой по Сети потребует много времени. Поэтому общепринятый путь создания коллекций картинок таков:

- коллекция исходных картинок размещается в отдельной папке, возможно, в ней есть и вложенные папки для отдельных категорий рисунков; обычно

исходные изображения хранятся в формате JPEG со значением параметра качества от 65 до 90;

- в HTML-код страницы фотоальбома включаются только *эскизы* или миниатюры, сделанные с основных фотографий их пропорциональным уменьшением и, возможно, конвертированием в другой графический формат. Картинки-эскизы одновременно являются ссылками на основные изображения, открываемыми в новом или текущем окне. Возможно, эскизы снабжаются дополнительными описаниями.

Встречается как последовательное описание тегами `<img>` серии картинок, встроенных в текст, так и размещение их в ячейках таблицы. Разумеется, в последнем случае браузеру потребуется дополнительное время на формирование таблицы, что замедлит и без того небыстрый процесс загрузки графики.

Для создания миниатюрных версий картинок удобнее всего пользоваться режимом пакетной обработки популярного графического браузера ACDSee. Нужная команда называется **Batch Resize Images** и вызывается в окне программы нажатием комбинации клавиш <Ctrl>+<R>. Между прочим, нажатием комбинации <Ctrl>+<F> можно вызвать мощный и удобный мастер конвертирования файлов. Как правило, серия изображений, полученных со сканера или цифровой камеры, имеет одинаковые размеры. Задав на первом шаге требуемый размер миниатюры, мы буквально одним нажатием кнопки получим серию уменьшенных картинок, которые затем преобразуются в формат GIF Мастером конвертирования. Разумеется, при указании размеров следует позаботиться о сохранении пропорций исходных рисунков, а также проверить, есть ли у вас копии всех изображений, на тот случай, если выбрана опция удаления исходных файлов.

Средствами HTML нельзя организовать *предварительную загрузку* изображений, но это легко сделать, используя JavaScript. В Интернете доступно множество скриптов, предназначенных для упрощения задачи размещения в документе большого количества картинок. Пример подобного несложного скрипта приведен в файле PhotoGallery.html из папки Glava_15 компакт-диска.

Более подробно этапы технологической обработки изображений рассмотрены в *разд. 10.8*, а в завершение этого раздела приведем пример, фрагмент кода которого содержит листинг 10.5, а внешний вид показан на рис. 10.7. В этом примере коллекция картинок содержит две категории: "Цветы" и "Животные". Для большей компактности в листинге приведена только первая из них. Полный текст примера записан на компакт-диске под именем Glava_10\PhotoAlbum\index.html. Разумеется, при создании ваших собственных коллекций рисунков именно такая организация дерева папок необяза-

тельна. Также заметим, что, используя практические навыки по созданию меню и шаблонов, эту работу можно значительно упростить, даже делая сайт в невизуальном редакторе HTML.

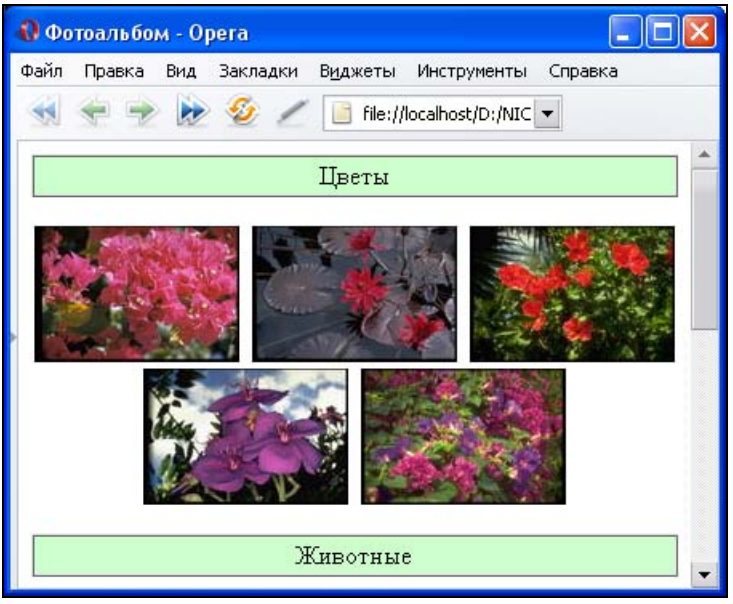

Рис. 10.7. Организация коллекции картинок

Листинг 10.5. Организация коллекции картинок

```
<html>
<head>
<title>Фотоальбом</title>
</head>
<body bgcolor="white">
<div align="center">
<!-- таблица-заголовок категории -->
<table width="100%" border="1" bgcolor="#008000" cellpadding="0"
cellspacing="0">
<tr><td>
 <table width="100%" border="0" cellpadding="2" cellspacing="0">
  <tr><td align="center" bgcolor="#CCFFCC">
   Цветы
 </td></tr></table>
```

```
</td></tr></table>
<!-- эскизы со ссылками на основные картинки -->
<p>
 <a href="flowers/01.jpg" target="_blank"><img src="thumb/flowers/01.jpg"
width="128" height="85" border="0" alt="Рис.1" hspace="2" vspace="2"></a>
 <a href="flowers/02.jpg" target="_blank"><img src="thumb/flowers/02.jpg"
width="128" height="85" border="0" alt="Рис.2" hspace="2" vspace="2"></a>
 <a href="flowers/03.jpg" target="_blank"><img src="thumb/flowers/03.jpg"
width="128" height="85" border="0" alt="Рис.3" hspace="2" vspace="2"></a>
 <a href="flowers/04.jpg" target="_blank"><img src="thumb/flowers/04.jpg"
width="128" height="85" border="0" alt="Рис.4" hspace="2" vspace="2"></a>
 <a href="flowers/05.jpg" target="_blank"><img src="thumb/flowers/05.jpg"
width="128" height="85" border="0" alt="Рис.5" hspace="2" vspace="2"></a>
</p>
<!-- аналогично для других категорий -->
</div></body></html>
```

10.7. Организация блоков с использованием графики

Используя и развивая технологию блочно-табличного создания шаблонов сайта, описанную в *разд. 8.4*, можно выделить несколько основных применений графики в структуре сайта:

- заголовки и тела блоков с фоновой графикой;
- графические логотипы, рамки и другие элементы интерфейса;
- пустые рисунки с прозрачным фоном для заполнения пространства в таблицах или между таблицами;
- графические маркеры списков и горизонтальные разделители.

Приведем примеры реализации этих возможностей.

Изменим пример из листинга 8.8 так, чтобы ячейка с заголовком блока содержала фоновую картинку, подключенную так, как показано в листинге 10.6.

Листинг 10.6. Блок, содержащий в заголовке фоновое изображение

```
<table width="200" border="0" cellpadding="1" cellspacing="0" bgco-
lor="#452900">
 <tr>
  <td>
   <table width="198" border="0" cellpadding="4" cellspacing="0">
    <tr align="center" bgcolor="#CC7A06">
```

```
     <th width="198" height="24" nowrap background="images/blockbk.jpg">
<font color="#FFFFFF">
      Заголовок блока
     </font></th>
    </tr>
    <tr align="justify" valign="top" bgcolor="white">
     <td>Содержимое блока</td>
    </tr>
   </table>
</td></tr></table>
```

Абсолютная ширина внешней таблицы блока, заданная равной 200 пикселов, потребовала вставки картинки шириной 198 пикселов — мы ведь не забыли про рамку шириной в 1 пиксел вокруг блока. Цвета блока также были изменены для большей "гармонии" с фоном заголовка. Посмотреть этот пример в работе вы можете, открыв файл Glava_10\BlockGraph.html с компакт-диска.

Всем хорошо знакомый *логотип* сайта также можно вставить в ячейку таблицы-заготовки. Попробуйте изменить с этой целью шаблон Template1.html из папки Glava_8. Как минимум, следует позаботиться о том, чтобы края рисунка с логотипом были того же цвета, что и фон титульной таблицы сайта, а сам рисунок был центрирован по горизонтали и вертикали в ее единственной ячейке.

Применяя графику в табличных блоках, можно создавать все более сложные элементы интерфейса. К примеру, нетрудно организовать "резиновую" рамку с закругленными краями, которая растягивалась бы под размер содержимого. Для этого воспользуемся свойством браузера при необходимости масштабировать картинки. Масштабировать мы будем маленькие рисунки, растягивая их в одну из сторон. Для организации подобного блока нам понадобится таблица размером 3×3 с картинками в каждой ячейке, кроме центральной, а также 6 небольших изображений (4 закругленных уголка рамки и ее вертикальная и горизонтальная стороны).

Углы этой рамки растягиваться не будут, боковые стороны растягиваются только по одной координате, а середина, заполненная цветом, принимает произвольные размеры, изменяясь в обоих направлениях. Из экономии места я привожу здесь только ссылку на пример с компакт-диска Glava_10\Ramka.html, реализующий описанный подход в чуть усложненном варианте: в рамке реализован трехмерный эффект, что потребовало большего количества картинок. Разобрав этот пример, вы сможете создать графический блок по собственному вкусу. Сделаем еще пару замечаний:

- "безопасный" способ заполнения пустых ячеек таблиц (например, служащих только для подключения фоновых рисунков) — размещение в ячейке

такой таблицы прозрачного изображения в формате GIF размером 1×1 пиксел. Подобные примеры вы найдете в документе Ramka.html, а сам пустой файл GIF с именем 1.gif и размером 49 байтов — в папке Glava10\img;

- при растягивании элемента рамки по горизонтали допускается в теге `<img>` указывать атрибут `width="100%"`, при растягивании по вертикали надежнее делать часть рамки фоновым изображением ячейки таблицы, содержащей пустой прозрачный рисунок.

Подобных элементов интерфейса можно придумать очень много, но важно, прежде всего, понять принцип: встраивая картинки в ячейки или делая их фоном таблиц, можно организовать блоки любой сложности.

Наконец, пример Glava_10\Ramka2.html демонстрирует использование графических разделителей и маркеров списков. Код этого небольшого примера приведен в листинге 10.7, а внешний вид — на рис. 10.8. Отметим, что размер файла bk2.gif, служащего альтернативным тегу `<hr>` горизонтальным разделителем, составляет всего 52 байта, а веселый маркер-смайлик больше ненамного — всего 111 байтов (файл smile.gif). Эти файлы вы также найдете в папке Glava_10\img. Такой оптимизации легко достигнуть, пользуясь мастером сохранения Web-графики в последних версиях Adobe Photoshop.

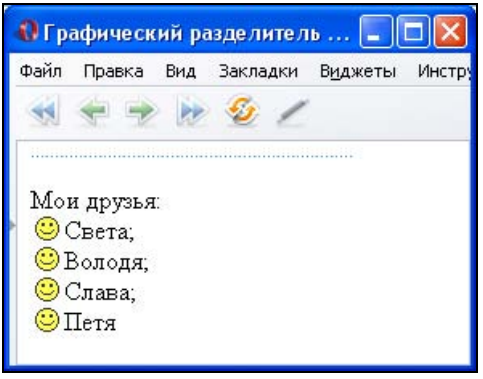

Рис. 10.8. Графический разделитель и маркер списка

Листинг 10.7. Графический разделитель и маркер списка

```
<html>
<head>
  <title>Графический разделитель и маркер списка</title>
```

```
</head>
<body bgcolor="#FFFFFF" text="#000000">
<!-- Графический разделитель -->
<table border=0 width="200" cellpadding="0" cellspacing="0">
 <tr><td height="3" background="img/bk2.gif" nowrap>
  <img src="img/1.gif" align="left" width="1" height="1">
</td></tr></table>
<!-- Список с графическими маркерами -->
<p>Мои друзья:
<br><img src="img/smile.gif" width="17" height="17" align="top"
      hspace="2">Света;
<br><img src="img/smile.gif" width="17" height="17" align="top"
      hspace="2">Володя;
<br><img src="img/smile.gif" width="17" height="17" align="top"
      hspace="2">Слава;
<br><img src="img/smile.gif" width="17" height="17" align="top"
      hspace="2">Петя
</p></body></html>
```

Для всех рисунков, служащих разделителями, заполнителями или маркерами, нецелесообразно указывать атрибуты `alt` или `title`.

10.8. Программное обеспечение для работы с графикой

Объем этой книги не позволяет детально описать все разнообразие современных графических программ. Однако использование мощных возможностей графических редакторов — вопрос опыта. Прежде всего имеет смысл поговорить о принципах подготовки графики и выбора программного обеспечения для этих целей.

Как уже упоминалось, всю *предварительную обработку* изображения следует производить с файлом, сохраняемым в формате без потерь качества. Если файл получен со сканера или цифровой камеры, его желательно сохранить в несжатом формате (BMP). Если файл разрабатывается самостоятельно, его оригинал следует хранить в виде проекта одного из профессиональных графических редакторов. Обычно это Adobe PhotoShop (формат PSD) или CorelDRAW (CDR).

Предварительная обработка несжатого изображения может включать в себя операции *кадрирования* (обрезки краев) и *масштабирования* для уменьшения

линейных размеров картинки. Оптимальным представляется последовательное использование обоих методов. Например, для обработки высококачественного несжатого изображения размером 5120×3840 пикселов целесообразно сначала обрезать 10% изображения по краям, а затем уменьшить его масштаб до 25% от размера после обрезки. Размер полученного варианта картинки составит 0,9×0,25 = 0,225 от исходного или 1152×864 пиксела. Этот вариант будет размещаться в Интернете как основное изображение. Затем на его основе можно подготовить эскиз некоторого стандартного размера (обычно ширина эскизов не превышает 200 пикселов). При уменьшении картинки коэффициент масштабирования по возможности должен быть кратен целому числу (например, 25%). С другой стороны, если это возможно, следует приводить картинки к некоторым стандартным размерам, таким как 1024×768, 640×480 или 512×384 и заботиться о сохранении "естественного" соотношения ширины и высоты (4:3 или 3:4).

Собственные изображения следует создавать в графическом редакторе, поддерживающем многослойную графику и "плавающие" настраиваемые объекты, которые можно редактировать неоднократно. Независимо от того, разрабатывается растровая, векторная или комбинированная картинка, редактор должен давать возможность сохранения и оптимизации изображения для Web.

Оптимизация графики для размещения на Web-странице — это поиск компромисса между ее качеством и объемом файла. По сути дела, оптимизация сводится, во-первых, к выбору одного из двух форматов (GIF или JPEG), а во-вторых, к его настройке.

При выборе формата руководствуйтесь следующим:

- используйте формат GIF, если изображение без видимых потерь качества переводится в 128-цветовую гамму с включенной опцией "dithering" (*точечное размытие*). В противном случае лучше сохранять изображение в JPEG-формате;
- подобрать минимальное количество цветов GIF можно вручную. При этом для качественной передачи одноцветного изображения на одноцветном фоне достаточно 5—8 цветов, для двух-трехцветного изображения — 15—25 цветов);
- по возможности избегайте включения опции "dithering", увеличивающей размер файла. Это требуется только для изображений, имеющих значительные цветовые или тоновые переходы (такие, как тень или внешний контур другого цвета). После применения эффекта точечного размытия убеждайтесь, что фон не стал "клетчатым";

- если в первую очередь важно уменьшение размера файла, коэффициент качества JPEG должен составлять до 80–90, если важнее сохранение высокого качества, то 92–94.

После определения формата и настроек качества, возможно применение к масштабированному изображению *эффектов*, доступных из панелей инструментов графического редактора. Добавляйте эффекты только с помощью профессиональных редакторов, оптимизирующих картинку для Web. После этого обработанное изображение *однократно* сохраняется в выбранном Web-формате.

Если изображение имеет дополнительное *внешнее* обрамление, оно наносится в последнюю очередь непосредственно перед сохранением рисунка для Web. Если к редактированию файла JPEG пришлось вернуться повторно, сохраняйте его с качеством 100, чтобы избежать вторичного сжатия и дополнительных потерь качества.

Среди приложений, ориентированных на работу с растровой графикой, наиболее популярными заслуженно считаются Adobe Photoshop, Corel PhotoPaint и Macromedia Fireworks.

В состав пакета Adobe Photoshop входит специализированный редактор Web-графики Adobe ImageReady. Как и Macromedia Fireworks, это приложение имеет расширенные средства для оптимизации графики, создания графических кнопок меню, фоновой графики и рамок. В обоих приложениях присутствует возможность создания анимации GIF. С последней задачей отлично справляется и специализированный редактор Ulead GIF Animator.

Как правило, развитые приложения *векторной* графики, позволяя создавать изображения в форматах, не зависящих от разрешения, содержат средства конвертирования в растровые форматы, поддерживаемые в Интернете. Среди этих программ наиболее популярны Adobe Illustrator, Macromedia Freehand и CorelDRAW. Освоение любого из перечисленных инструментов позволит вам работать с графикой профессионально.

Глава 11

Картированные изображения

Сегодня многие Web-страницы располагают интересной разновидностью меню — *картированными изображениями* — рисунками, отдельные части которых реагируют на нажатие кнопки мыши. Обычно эти чувствительные части изображения связаны с HTML-документами, т. е. представляют собой *ссылки*.

В качестве графического формата для изображения-карты, как правило, выбирается GIF с чересстрочной загрузкой, который поддерживается всеми браузерами и позволяет выводить изображение, постепенно уточняя детали. Как и всегда при работе с графикой, следует позаботиться о том, чтобы размер файла-изображения не был слишком большим.

Изображение-карта может обслуживаться как *сервером*, так и *на стороне клиента*, т. е. браузером. Мы рассмотрим только последний вариант, поскольку для его реализации не потребуется иных средств, кроме тегов HTML.

11.1. Преимущества и недостатки картирования

Проектирование и использование картированных изображений дает следующие преимущества:

- для изготовления карты требуется всего один графический файл и один блок информации о том, какие области изображения чувствительны к нажатию кнопки мыши. Таким образом, карта — перспективный, красивый и достаточно удобный в модификации вариант основного меню сайта;
- карты — наиболее подходящее средство представления любых *пространственных отношений* между группой объектов. Например, невозможно придумать лучший способ размещения географической карты с описанием

каждого региона на своей Web-странице. В этом случае для перехода к файлу, содержащему информацию о конкретном регионе, пользователю достаточно щелкнуть мышью по его контуру на карте.

Недостатки карт, увы, являются продолжением их достоинств:

- не следует забывать о пользователях, браузер которых не поддерживает графику или она отключена: если на странице, помимо картированного изображения, нет других средств навигации, они не смогут увидеть результаты вашей работы. Поэтому наряду с картированным изображением Web-мастер обычно разрабатывает текстовый вариант меню;
- неудачно спроектированное изображение-карта может сбить пользователя с толку. Если карта такова, что активные области-ссылки не определяются интуитивно, пользователь сможет узнать о них только одним способом: перемещая в пределах изображения курсор мыши и отслеживая появляющиеся адреса ссылок в статусной строке браузера, что весьма неудобно;
- информация о посещенных и непосещенных на карте ссылках не выделяется ни одним браузером.

Тем не менее, разумное применение этой технологии способно существенно улучшить внешний вид ваших Web-страниц и удобство работы с ними.

11.2. Конфигурация клиентского изображения-карты

Для определения *областей картинки*, чувствительных к нажатию кнопки мыши, их следует описать в контейнерном теге `<map>`, имеющем вид

```
<map name="текст">...</map>.
```

Здесь значение атрибута `name` представляет собой наименование карты, назначенное Web-мастером. Для надежности следует давать картам имена, не содержащие пробелов, начинающиеся с буквы и состоящие только из латинских букв и цифр.

Располагать тег `<map>` можно в любом месте документа, обычно это делают либо в начале страницы после тега `<body>`, либо непосредственно перед тегом изображения `<img>`, к которому относится карта.

Внутри элемента `<map>` *каждая чувствительная область изображения* описывается в теге `<area>`. Его общий вид следующий:

```
<area shape="форма" coords="список координат" href="URL" alt="Описание">
```

Закрывать тег `<area>` не нужно.

Рассмотрим подробнее назначение отдельных атрибутов.

Атрибут `shape` (форма) может принимать одно из значений, описанных в табл. 11.1.

Таблица 11.1. *Значения атрибута shape*

Атрибут	**Вид области, чувствительной к нажатию мыши**
`shape="rect"`	Прямоугольник (значение по умолчанию)
`shape="circle"`	Круг
`shape="poly"`	Произвольный замкнутый многоугольник
`shape="default"`	Все изображение. В этом случае не используется атрибут `coord`

При пересечении двух чувствительных областей изображения действует значение, определенное первым встретившимся тегом `<area>`. Таким образом, *порядок следования* тегов `<area>` внутри тега `<map>` имеет значение, что является для HTML скорее исключением.

Значение атрибута `coords` (список координат) зависит от выбранного значения атрибута `shape` и включает в себя координаты в пикселах, перечисленные через запятую. Как обычно, сначала указывается `X`-, а затем `Y`-координата и при этом считается, что ось Y расположена *сверху вниз от левого верхнего угла рисунка*. Левый верхний угол рисунка всегда имеет координаты (0,0).

Для прямоугольника указываются координаты левого верхнего и правого нижнего углов, например

```
<area shape="rect" coords="0,0,200,100">.
```

Здесь чувствительная прямоугольная область — верхний левый угол рисунка шириной 200 и высотой 100 пикселов.

Для круга указываются `X`- и `Y`-координаты центра и радиус в пикселах, например

```
<area shape="circle" coords="100,200,50" href="1.html">.
```

Центр чувствительной области расположен в точке, отстоящей на 100 пикселов от левого края по оси X и на 200 пискелов от верхнего края по оси Y. Радиус окружности при этом составляет 50 пикселов, а щелчок мышью по ней открывает документ `1.html` из текущей папки.

Для многоугольника перечисляются пары `X`- и `Y`-координат вершин. Для надежности многоугольник должен быть *замкнутым*, т. е. первая пара вершин

должна совпадать с последней. Если многоугольник не замкнут, при интерпретации данных для этой формы браузер автоматически соединит первую точку с последней. Многоугольник не обязан быть выпуклым. Жесткого количественного ограничения на число вершин нет. По крайней мере, многоугольник, имеющий 100–200 вершин, будет уверенно обработан всеми современными браузерами. Это дает практически неограниченные возможности указания сколь угодно сложных чувствительных областей изображения. Например, тег

```
<area shape="poly" coords="0,300,100,300,100,150">
```

определяет чувствительную к нажатию кнопки мыши область в виде прямоугольного треугольника, катеты которого ориентированы по оси X на уровне 300 пикселов от верхнего края рисунка и по оси Y на расстоянии 100 пикселов от его левого края.

Хорошо знакомые нам атрибуты `href` и `alt` имеют то же применение, что в теге `<img>`. Правила их задания также ничем не отличаются: значениями атрибута `href` могут быть как относительные, так и абсолютные URL, а в атрибуте `alt` следует указать короткий, но содержательный текст, сообщающий о ссылке. Текст, заданный значением атрибута `alt`, выводится в качестве подсказки при попадании курсора мыши на активную область не всеми браузерами. Надежнее использовать атрибут `title`.

Для *исключения* части картинки из карты вместо атрибута `href` нужно задать специальное значение `nohref`, указывающее, что область *не является* ссылкой. Например, конструкция тегов

```
<area shape="circle" coords="100,100,50" nohref>
<area shape="circle" coords="100,100,100" href="http://www.ring.com"
alt="Кольцо сайтов">
```

создаст чувствительную область в виде кольца толщиной 50 пикселов. Центр этого кольца находится в точке с координатами 100,100. Эта точка представляет собой центр двух окружностей с координатами 50 и 100 пикселов, а исключение внутренней окружности из карты возникает за счет того, что у первого тега `<area>` указан атрибут `nohref`.

Для связывания изображения с созданной картой достаточно в теге `<img>`, с помощью которого вставлено изображение, указать атрибут вида

```
usemap="#имя_карты"
```

Значение "`имя_карты`" должно быть предварительно определено в теге `<map>`. Например, тег

```
<img src="1.jpg" width="526" height="400" border="1" usemap="#a1">
```

подключает карту с наименованием `a1`, описанную в текущем документе. На самом деле значение атрибута `usemap` — это *закладка*, возможно, нахо-

дящаяся в другом документе. Так, запись `usemap="maps/maps.html#a1"` заставит браузер искать карту `a1`, т. е. тег `<map>` с соответствующим значением атрибута `name`, в файле maps.html, находящемся во вложенной папке maps.

Помещение элементов `<map>` в другой документ позволяет собрать все карты в одном месте, что может оказаться удобным при сложном картировании, но потребует лишних обращений к сетевым ресурсам со стороны браузера.

Атрибут `usemap` имеет *приоритет* перед ссылкой, определяемой тегом `<a>`. Если изображение 1.jpg из предыдущего примера сделать ссылкой, заключив в тег `<a href=…>…</a>`, то она будет проигнорирована браузерами.

При помещении в документ рисунка-карты не следует указывать его размер в процентах, иначе на мониторах с различным разрешением экрана карта будет накладываться на изображение неправильно.

Приведем пример картирования изображения. В листинге 11.1 показан HTML-код документа, реализующий рассмотренный ранее пример с кольцом. Результат выполнения этого кода иллюстрирует рис. 11.1. Использование желтого цвета для заливки кольца в файле GIF с прозрачным фоном и тот же цвет фона документа позволили создать область, выделенную только красным контуром шириной в 1 пиксел. На компакт-диске этот пример записан в файле Glava_11\Map1.html.

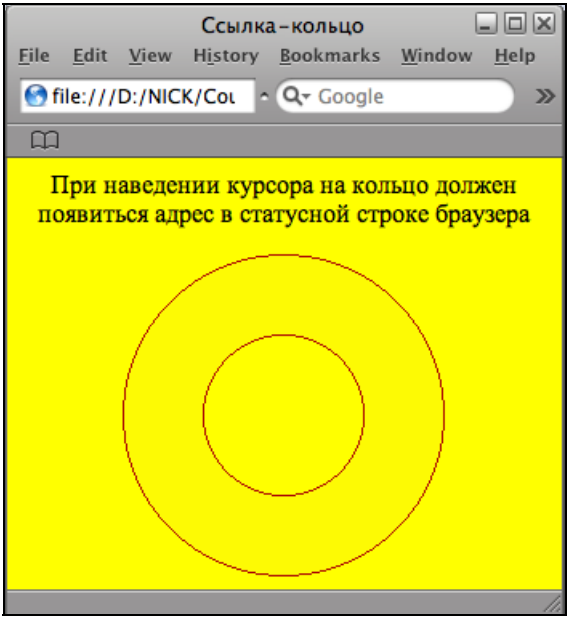

Рис. 11.1. Изображение, картированное кольцом

Листинг 11.1. Изображение, картированное кольцом

```
<html>
<head>
 <title>Ссылка-кольцо</title>
</head>
<body bgcolor="yellow">
<div align="center">
<p>При наведении курсора на кольцо должен появиться адрес
в статусной строке браузера</p>
<map name="map1">
 <area shape="circle" coords="100,100,50" nohref>
 <area shape="circle" coords="100,100,100" href="http://www.ring.com"
  title="Кольцо сайтов">
</map>
<img src="images/map1.gif" width="201" height="201" border="0" use-
map="#map1">
</div></body></html>
```

В завершение раздела заметим, что "ручная" разметка карт-изображений — дело нелегкое. В простейшем случае определить координаты опорных точек и спланировать карту поможет стандартная программа Paint, в статусной строке которой показываются координаты выбранной в данной момент точки изображения, а при отметке части изображения инструментом **Выделение** дополнительно выводятся размеры выделенной области.

Существует и ряд специализированных решений для создания сложных изображений-карт. Так, популярны программы MapEdit, Map THIS!, Coffee Cup Image Mapper, Sausage Image Mapper и некоторые другие. Информацию о них легко найти в Интернете.

Приложения Adobe ImageReady и CorelDRAW последних версий также содержат все необходимое для создания карт любой сложности.

Глава 12

Фреймы

Фреймами или *кадрами* называют независимые окна внутри основного окна браузера, в которых могут одновременно отображаться разные документы. Фреймы удобны при создании страниц, которые должны иметь как динамическое, так и статическое содержимое. Например, узкий левый фрейм может содержать оглавление сайта, а широкий правый будет предназначен для вывода информации. Возможны и любые другие конфигурации.

Принципиальное отличие фреймов от таблиц и других способов структурирования документа состоит в том, что содержимое отдельных фреймов можно прокручивать или изменять независимо от других. Таким образом, фрейм — полноценное "подокно" внутри основного окна браузера.

В этой главе мы рассмотрим теорию и практику работы с фреймами.

12.1. Применение фреймов: за и против

Создание фреймов удобно и оправдано в следующих случаях:

- при необходимости разбить Web-страницу на "навигационную" и "содержательную" части. Это может быть полезно, например, при выводе в одном фрейме оглавления или списка терминов, а в другом — текста статей или описаний терминов;
- для расположения в окне браузера статического окна с информацией, которая должна быть видна постоянно;
- для представления любой другой информации, которую удобнее расположить в нескольких смежных окнах, каждое из которых может просматриваться независимо;
- для проектирования структуры окон, с которой будут работать интерактивные Web-приложения, такие как сетевые игры, чаты или доски объявлений.

Если подумать, можно найти и другие удобства. Так, при создании навигационного меню во фрейме необходимость многократного дублирования ссылок отпадает сама собой — ведь документ, содержащий меню, будет единственным. Кроме того, постоянное присутствие во фрейме навигационного меню и других повторяющихся элементов страницы уменьшит объем передаваемых по сети данных.

Тем не менее, использование фреймов наталкивается на целый ряд серьезных возражений:

- исторически сложилось так, что поддержка фреймов браузерами различается;
- при перемещении по страницам "фреймового" сайта с помощью кнопок браузера **Вперед** и **Назад**, а также при выборе ссылки, ведущей в другой фрейм, в адресной строке браузера ничего не меняется;
- браузеры не всегда корректно открывают исходный текст страницы с фреймами и сохраняют закладки на фреймы в Избранном;
- пользователь, нашедший ваш документ, содержащийся во фрейме, с помощью поисковой машины, увидит его в полном окне, а не во фрейме, что может полностью лишить его навигации.

Говоря обобщенно, страница, построенная на фреймах, объединяя *несколько* HTML-документов, перестает быть как информационной, так и навигационной единицей отображения информации, а ее URL-адрес — унифицированным идентификатором ресурса. Поместив URL такой страницы, например, в папку "Избранное", после перехода по ссылке мы увидим не текущее состояние каждого фрейма, а начальное отображение фреймового документа. Фактически, некоторые Web-страницы оказываются лишенными собственного адреса, что в корне противоречит самой концепции WWW. Противоречат фреймы и концепции языка HTML, согласно которой каждый его элемент описывает некоторое *значение*, а не просто способ отображения.

Применение "плавающего" или вложенного в страницу фрейма `<iframe>`, описанного в *разд. 12.4*, сохраняет подчиненность объектов главной странице, но оставляет нерешенной проблему с адресной строкой браузера.

Можно сделать вывод, что фреймы подходят для публикации лишь такой информации, "содержательная" часть которой может быть отделена от "навигационной" без нарушения целостности сайта. Указанные примеры со словарными статьями и оглавлением книги во фрейме — как раз из этой серии.

12.2. Описание коллекций фреймов

Итак, фрейм — это прямоугольное "подокно" внутри основного окна браузера. Для создания прямоугольной сетки фреймов служит тег `<frameset>` ("frame set" — "множество рамок"). Он *заменяет* тег `<body>` при описании набора фреймов. Таким образом, главный документ страницы с фреймами "ничего не делает", а только описывает структуру фреймов. Документы, помещаемые в отдельные фреймы, строятся по обычным правилам, в том числе, любой ранее разработанный документ может быть помещен во фрейм.

Каждый фрейм может представлять собой отдельное окно или быть разделенным еще на несколько фреймов вложенным тегом `<frameset>`. Это очень напоминает вложение таблиц — каждый тег `<frameset>` создает только простую прямоугольную сетку окон, но внутри него можно поместить произвольное число других тегов `<frameset>`, каждый из которых описывает вложенную прямоугольную сетку. Разница лишь в том, что таблицы в HTML описываются построчно и по отдельным ячейкам, а в открывающей части тега `<frameset>` сразу же указано число строк и столбцов. Порядок описания отдельных фреймов в тегах `<frame>`, вложенных в тег `<frameset>`, тот же, что для ячеек таблиц: сначала слева направо, а затем сверху вниз.

Рассмотрим эти два тега подробнее. У элемента `<frameset>` есть два главных атрибута: `rows="список величин"` и `cols="список величин"`, описывающие, соответственно, *строки* и *столбцы* таблицы фреймов.

Строка "`список величин`" представляет собой разделенный запятыми перечень значений в *пикселах*, *процентах* или *относительных величинах*.

Например, тег

```
<frameset rows="100,240,140">
```

создает сетку из трех кадров, *высота* которых равна 100, 240 и 140 пикселам соответственно. Если высота окна браузера не равна 100 + 240 + 140 = 480 пискелов, то высоты фреймов будут пропорционально пересчитаны.

Следующий тег, имеющий вид

```
<frameset cols="25%,75%">,
```

создает 2 кадра *шириной* 25 и 75 процентов от ширины окна браузера.

Наконец, *относительный размер* имеет общий вид `i*`, где `i` — целое положительное число. При распределении пространства браузер сначала выделяет место для размеров, заданных в пикселах и процентах, а затем разделяет оставшееся пространство между элементами с относительными размерами. При этом элементу с указанным размером `3*` будет выделено втрое больше про-

странства, чем элементу с размером `1*`. Значение `*` эквивалентно `1*` и часто означает указание заполнить оставшееся пространство.

Таким образом, тег

```
<frameset cols="*,2*">
```

создаст 2 кадра с шириной 1/3 и 2/3 от текущей ширины окна браузера.

Эти способы можно использовать и *совместно* — например, тег

```
<frameset cols="128,*">
```

инструктирует браузер оставить 128 пикселов слева под первый фрейм, а все остальное пространство отвести под второй. При совместном определении атрибутов `rows` и `cols` в одном теге, таком как

```
<frameset rows="*,2*" cols="*,2*">
```

получается сетка кадров размерностью 2×2.

Кроме двух указанных, тег поддерживает только общие атрибуты и события `onload` и `onunload` (см. *разд. 4.2*). Тег является блочным и закрывается.

Содержимое тега `<frameset>` — описания отдельных кадров, каждое из которых выполнено в собственном теге `<frame>`. Подобно `<img>`, тег `<frame>` описывает только *наполнение* для отдельного кадра и не содержит контейнеров, поэтому не нуждается в закрывающем теге.

Число тегов `<frame>` должно быть равно числу кадров, определенных в теге `<frameset>`, при этом считается, что кадры описываются слева направо и сверху вниз.

В табл. 12.1 приведены основные атрибуты тега `<frame>`.

Таблица 12.1. *Основные атрибуты тега* `<frame>`

Атрибут	**Описание**
`src="URL"`	Задает абсолютный или относительный адрес документа, отображаемого в данном кадре. Документ не обязательно является документом HTML
`name="строка"`	Указывает *наименование* фрейма. Это необходимо сделать, если предполагается ссылаться из одних фреймов на другие
`scrolling`	Управляет *полосами прокрутки* фрейма. Допустимы три значения атрибута: "`yes`" (полосы есть всегда), "`no`" (никогда) и "`auto`" (если необходимо; значение по умолчанию)
`noresize`	Запрещает пользователю изменять *размеры кадра*. Атрибут `noresize`, указанный для данного фрейма, влияет также и на все фреймы, смежные с ним

Таблица 12.1 *(окончание)*

Атрибут	Описание
`frameborder`	Включает или выключает *отображение обрамления* кадров. Допустимы значения `1` (обрамление отображается, по умолчанию) или `0` (не отображается). В случае `1` обрамление имеет трехмерную форму, иначе оно невидимо. Указание `0` может привести к тому, что пользователь в любом случае не сможет менять размеры кадра
`marginwidth="пикселы"`	Ширина левого и правого поля фрейма в пикселах
`magrinheight="пискелы"`	Ширина верхнего и нижнего поля фрейма в пикселах

Вследствие исторической "войны браузеров" между Internet Explorer и Netscape Navigator, современные обозреватели "понимают" атрибут `frameborder` также в виде `frameborder="yes"` или `frameborder="no"`, однако этот способ нестандартный. Иногда атрибут `frameborder` указывают непосредственно в теге `<frameset>`, для управления отображением всех видимых рамок сразу. Хотя этого также нет в спецификации, все проверенные браузеры справились с задачей.

В исходных текстах страниц, написанных "под Inrernet Explorer" можно встретить атрибут `framespacing="пикселы"`, указывающий *ширину промежутка* между смежными кадрами в пискелах. Этот атрибут нестандартный, не работает в остальных браузерах и не рекомендуется к использованию.

Кроме тегов `<frame>`, внутри `<frameset>` может находиться блочный тег `<noframes>...</noframes>`, определяющий содержимое, которое будет выводиться браузерами, не поддерживающими фреймов. В настоящее время трудно встретить такой браузер или поисковую машину, однако добавлять `<noframes>` все-таки следует, по меньшей мере, помещая туда краткое описание и ссылку на карту сайта (см. *разд. 6.5*).

При отсутствии документов, адресуемых атрибутом `src` тега `<frame>`, браузер Internet Explorer может не показать даже структуру фреймов, остальные браузеры, как правило, покажут в каждом фрейме стандартное сообщение об ошибке загрузки.

В листинге 12.1 приведен код простого документа с фреймами, содержащего необходимый минимум атрибутов. Его внешний вид иллюстрирует рис. 12.1. Выполнить этот пример в своем браузере вы можете, открыв документ Glava_12\Frames1.html с компакт-диска.

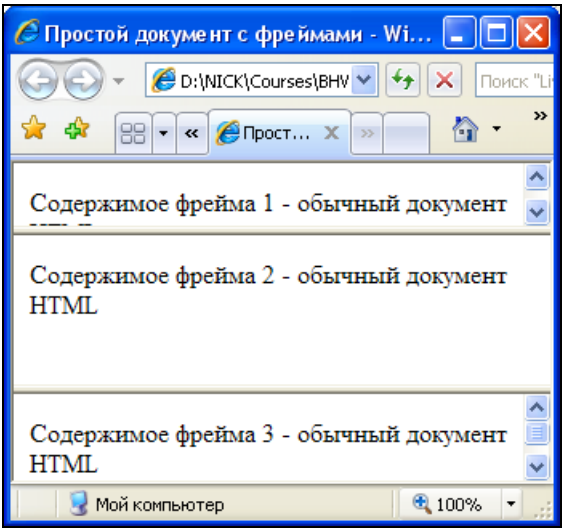

Рис. 12.1. Простой документ с фреймами

Листинг 12.1. Простой документ с фреймами

```
<html>
<head>
 <title>Простой документ с фреймами</title>
</head>
<frameset rows="100,240,140">
 <frame name="frame1" src="frames/frame1.html">
 <frame name="frame2" src="frames/frame2.html">
 <frame name="frame3" src="frames/frame3.html">
 <noframes>
  <p>Извините, Ваш браузер не поддерживает фреймы.
  <p>Вместо них Вы можете воспользоваться следующими ссылками:
  <a href="frames/frame1.html">фрейм 1</a>,
  <a href="frames/frame2.html">фрейм 2</a>,
  <a href="frames/frame3.html">фрейм 3</a>
 </noframes>
</frameset></html>
```

Для создания более сложных конфигураций кадров теги `<frameset>` могут вкладываться друг в друга таким образом, что внутренний тег `<frameset>`

занимает место любого из тегов `<frame>`. Пример вложенных фреймов приведен в листинге 12.2. На компакт-диске с примерами ему соответствует файл Glava_12\Frames2.html.

Листинг 12.2. Вложение фреймов

```
<html>
<head>
 <title>Документ с вложенными фреймами</title>
</head>
<frameset rows="40,*" frameborder="1">
 <frame src="reclama.html" name="top" scrolling="no" noresize>
 <frameset cols="128,*" frameborder="1">
  <frame src="menu.html" name="menu" scrolling="auto">
  <frame src="index.html" name="main" scrolling="yes">
 </frameset>
<noframes>
 Извините, ваш браузер не поддерживает кадры!
</noframes>
</frameset></html>
```

Здесь верхний кадр высотой 40 пикселов может служить, например, для вывода рекламы и связан с файлом reclama.html. Этот фрейм займет всю доступную ширину окна. Остальная часть окна с помощью вложенного элемента `<frameset>` разбита на две колонки. Левая имеет ширину 128 пикселов и может быть приспособлена для вывода меню (документ menu.html). Правая колонка занимает основную часть окна и содержит третий кадр, предназначенный для вывода основного содержимого страницы (файл index.html).

Чтобы пользователь не смог увидеть фрейм вне структурного документа, в теге `<body>` каждого отображаемого во фрейме файла достаточно указать следующий обработчик события `onload`:

```
onload="if (top.location==this.location) top.location='Frames2.html';".
```

Именно этот код включен во все три документа, отображаемые во фреймах, в чем вы можете легко убедиться, попытавшись открыть их отдельно от фреймовой структуры. При использовании этого приема в собственных проектах достаточно вместо имени файла Frames2.html вписать требуемый вам URL структурного документа.

Следует заметить, что сохранение страницы из фрейма — не такое простое дело. Во всех браузерах выбор стандартной команды просмотра HTML-кода

из меню **Вид** приведет к тому, что будет открыто содержимое только структурного документа, содержащего описание фреймов.

В некоторых браузерах в контекстном меню фрейма, вызываемом правой кнопкой мыши на свободном пространстве нужного кадра, содержится дополнительное подменю **Фрейм**, позволяющее показать, открыть или сохранить только содержимое текущего кадра.

В Internet Explorer в контекстном меню фрейма непосредственно содержится команда **Просмотр HTML-кода**, открывающая содержимое текущего фрейма в Блокноте. Затем это содержимое можно сохранить как HTML-документ (см. *разд. 4.3*).

12.3. Связь между фреймами и навигация с их помощью

В отличие от других способов структурирования Web-страницы, между фреймами существует прямое взаимодействие — в том смысле, что выбор ссылки в одном из фреймов может обновить содержимое другого. Для этого каждый фрейм имеет уникальное имя, отличающее его от других. Именно это имя задается атрибутом `name` тега `<frame>`. Для надежности следует давать фреймам имена только из латинских букв и цифр, без пробелов, а также соблюдать регистр символов при записи этих имен.

Для создания *ссылки из одного фрейма в другой* достаточно указать в теге ссылки `<a href=…>…</a>` атрибут вида `target="имя фрейма"`. Например, для ссылки из кадра `menu` в кадр `main` в листинге 12.2 достаточно записать в документе `menu.html` ссылку вида

```
<a href="URL нового документа" target="main">...</a>.
```

При щелчке по этой ссылке содержимое правого кадра, т. е. документ `index.html`, будет заменено на новый документ.

Если все документы должны открываться в определенном фрейме, это можно сделать с помощью тега `<base>`, рассмотренного в *разд. 4.5*:

```
<base target="main">.
```

Теперь все ссылки будут по умолчанию открываться во фрейме `main`, а для открытия ссылки в другом фрейме можно воспользоваться в ее открывающей части атрибутом `target`:

```
<a href="reclama2.html" target="top">обновить рекламу</a>.
```

При работе с кадрами можно также указывать атрибут `target` в одном из специальных видов, описанных в табл. 12.2. Указание символа подчеркивания

перед специальными именами обязательно. В противном случае имя фрейма `top` в листинге 12.2 можно было бы легко перепутать со специальным именем `_top`.

Таблица 12.2. *Специальные значения атрибута target*

Значение атрибута	Описание
`target="_self"`	Загрузить адресуемый документ в тот же кадр, откуда делается ссылка. Может использоваться для обхода умолчания, заданного тегом `<base>`
`target="_parent"`	Загрузить документ в *родительский* для данного кадр; если такого нет — результат действия аналогичен `_top`
`target="_top"`	Загрузить документ в полное окно, разрушая все кадры. Если документ уже располагается в полном окне, результат действия аналогичен `_self`
`target="_blank"`	Действует из фрейма так же, как из главного документа, т. е. открывает адресуемый документ в новом окне или вкладке браузера, не затрагивая существующих фреймов текущего окна

Чтобы понять, что такое *родительский фрейм*, проведем небольшой мысленный эксперимент. Представьте, что главный файл index.html описывает два фрейма с именами frame1 и frame2, каждый из которых в свою очередь содержит описание двух фреймов. Для frame1 назовем эти фреймы второго уровня frame11 и frame12, а для frame2, соответственно, frame21 и frame22. Теперь для frame1 и frame2 "родителем" служит главное окно, т. е. результат действия значений `_parent` и `_top` аналогичен, для фреймов же второго уровня frame11 и frame12 "родителем" является frame1, а для frame21 и frame22 — frame2. Указав в ссылке из frame12 атрибут `target="_parent"`, мы выведем адресуемый документ в окно, которое заменит только кадры frame11 и frame12, а указав `target="_top"`, мы разрушим все четыре кадра. Разобраться в этом детальней вам поможет файл Glava_12\Targets\index.html с компакт-диска, открыв который в своем браузере вы наглядно увидите действие всех значений атрибута.

Чтобы не зависеть от конкретных имен фреймов, указание специальных имен *предпочтительнее*, чем имен фреймов, назначенных Web-мастером.

Атрибут `target` также может быть указан у тега `<area>`, если ссылка с чувствительной к нажатию мыши области рисунка ведет в новый фрейм (см. *главу 11*).

12.4. Плавающие фреймы

В отличие от рассмотренных ранее *внешних* фреймов, *плавающий* или *внутренний* фрейм описывается в тексте обычного HTML-документа, что снимает часть спорных вопросов, обсуждаемых в *разд. 12.1*. На сегодняшний день плавающий фрейм — стандартный элемент HTML и использование его оправдано. Концепция внутреннего фрейма близка к встраиванию таблиц или изображений с той разницей, что в определенное место HTML-документа добавляется содержимое другого документа.

Концепция плавающего фрейма реализована тегом `<iframe>`, имеющим точно такие же атрибуты, как и `<frame>`, кроме `noresize`. Указание `noresize` в теге `<iframe>` бессмысленно, т. к. размер плавающего фрейма в любом случае не может быть изменен пользователем.

Кроме атрибутов, описанных в табл. 12.1, в теге `<iframe>` могут присутствовать знакомые нам атрибуты `align`, `width` и `height`.

Атрибут `align` выполняет свою обычную роль, задавая выравнивание фрейма относительно содержимого вмещающего его элемента. Он может принимать следующие значения:

- `top` — вертикальное выравнивание по верхней части контекста;
- `middle` — вертикальное выравнивание по центру контекста;
- `bottom` — вертикальное выравнивание по нижней части контекста;
- `left` — выравнивание по левому полю, содержимое обтекает фрейм справа;
- `right` — выравнивание по правому полю, содержимое обтекает фрейм слева.

Атрибуты `width="размер"` и `height="размер"` задают ширину и высоту плавающего фрейма в пискелах, процентах или относительных размерах, таких как `2*`.

Тег `<iframe>` *контейнерный*, т. е. должен закрываться. Его содержимым может быть альтернативная информация для браузеров, не поддерживающих данный тег. Пример HTML-кода "плавающего" фрейма для вывода отдельного документа приведен в листинге 12.3, а его внешний вид — на рис. 12.2. На компакт-диске этот пример записан в файле Glava_12\Iframe\index.html.

Листинг 12.3. Использование тега `<iframe>`

```
<html>
<head>
  <title>Документ с тегом iframe</title>
```

```
</head>
<body>
<p>Основное содержимое документа располагается
  <iframe name="links" src="links.html" frameborder="1" align="left"
   width="170" height="100">
   <p><i>Извините, Ваш браузер не поддерживает плавающие фреймы.
   <br>Это окно адресует документ
   <a href="links.html" target="_blank">links.html</a></i></p>
  </iframe> здесь. Документ в теге <tt>&lt;iframe&gt;</tt> с атрибутом
выравнивания <tt>align="left"</tt> ведет себя примерно так же,
как рисунок, вставленный тегом <tt>&lt;img&gt;</tt>
<p><a href="new.html" target="links">Ссылка для обновления iframe</a>
</body></html>
```

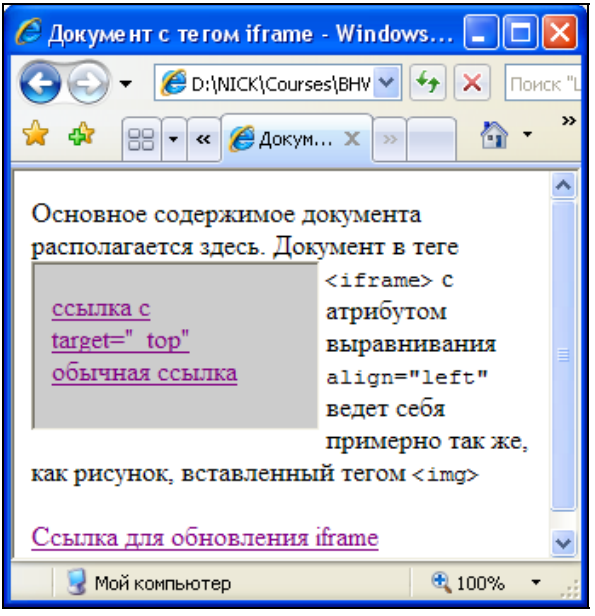

Рис. 12.2. Использование тега `<iframe>`

Здесь содержимое встроенного документа находится в файле links.html, сохраненного в текущей папке. При выборе ссылки для обновления `<iframe>` содержимое плавающего фрейма будет заменено документом new.html.

12.5. Программное обеспечение для работы с фреймами

Все популярные редакторы HTML, описанные в *разд. 8.5*, поддерживают работу с фреймами.

В Microsoft Word последних версий создать фреймы можно, применяя инструмент **Рамки** меню **Формат**. При этом документ автоматически приводится к виду **Веб-документ**. Создание фреймов в Word может быть оправдано исключительно для автоматического добавления во фрейм оглавления документа, элементы которого помечены соответствующими заголовочными стилями.

Из независимых программ, предназначенных для работы с фреймами, стоит отметить специализированные редакторы FrameGang и Frame-It.

ЧАСТЬ III

Разработка интерактивных документов средствами HTML

Освоив *часть II*, мы узнали почти все, что можно, о разработке *статических* документов HTML, содержащих размеченную с помощью тегов информацию.

Средства, изученные в *части III* книги, позволят сделать ваш сайт *динамическим*, т. е. более интерактивным, гибким и управляемым. *Глава 13* посвящена разработке пользовательских форм для ввода и передачи любых данных. В *главе 14* описаны стили — мощное и универсальное средство управления свойствами элементов HTML. Наконец, *глава 15* научит вас подключать к своей Web-странице разнообразное внешнее содержимое: от звуковых файлов и видео до внешних программ и информационных блоков.

Глава 13

Формы: создаем обратную связь

Прежде всего, нам следует научиться разрабатывать собственные *интерфейсы*, с помощью которых Web-страница сможет собирать вводимые пользователем данные.

Все мы, работая с современными операционными системами, привыкли взаимодействовать с ними через окна диалога, содержащие формы. Особо не задумываясь, мы работаем с кнопками, выпадающими списками и меню, переключателями и полями ввода. Почти все эти компоненты имеются в HTML, и использовать их так же легко, как остальные теги.

13.1. Область применения форм

Основное назначение форм в HTML — пересылка введенных пользователем данных Web-серверу, который занимается их обработкой. В *главе 2* мы уже говорили об архитектуре "клиент-сервер" и особенностях программирования для WWW, настала пора изучить имеющиеся в HTML средства для организации взаимодействия с пользователем.

Даже не владея в совершенстве языками серверного и клиентского программирования, с помощью форм можно решить целый ряд важных задач:

- организовать на Web-странице всевозможные анкеты, заявки, тесты, голосования и опросы, результаты которых передаются вам по электронной почте;
- подключить внешние серверные программы, получающие данные посредством форм; например, таким способом можно организовать у себя на сайте поиск информации или форму для отправки сообщения прямо со страницы;

- установить на странице клиентские приложения, требующие размещения на ней форм, служащих для ввода исходных данных пользователем (соответствующий пример приводится в *разд. 13.3*).

Кроме того, некоторые компоненты форм, такие, например, как выпадающие списки, удобны для представления пользователю информации в привычном для него виде.

13.2. Теги для организации форм

Для создания *интерактивной формы* служит контейнерный тег `<form>…</form>`, содержащий теги ее элементов. Кроме этого, в него могут помещаться программы-сценарии и любые блочные элементы. Документ может включать в себя любое количество форм, но они не могут быть вложены одна в другую.

Основные атрибуты формы приведены в табл. 13.1.

Таблица 13.1. *Основные атрибуты тега <form>*

Атрибут	Описание
`action="URL"`	Задает URL программы, обрабатывающей данные формы. В качестве `action` возможно также задание адреса электронной почты вида `mailto:name@server.com`. Атрибут является обязательным
`method`	Принимает одно из двух значений: `get`, (по умолчанию) предписывает посылать информацию формы вместе с URL, `post` — отдельно от URL. Значение `post` используется в случае отправки данных по электронной почте или при необходимости передавать значительный объем информации
`enctype="текст"`	Определяет способ кодирования данных формы. В случае отправки данных в виде неформатированного текста указывается `enctype="text/plain"`
`name="текст"`	Указывает имя формы для использования в сценариях

К формам применим также атрибут `target` со значениями, описанными в табл. 12.2. Например, после обработки данных, переданных из формы, находящейся в одном фрейме, результирующий документ может быть открыт в другом фрейме:

```
<form action="url" target="фрейм">.
```

Тег формы поддерживает обработку двух нестандартных событий:

- `onsubmit` происходит перед пересылкой формы для обработки;
- `onreset` происходит перед сбросом всех значений формы.

Способ обработки формы зависит от значений атрибутов `method` и `enctype`.

В общем случае эта обработка выполняется следующим образом.

1. Набор данных формы создается в виде последовательности пар "`имя элемента=значение`" для всех компонентов, которые удалось обработать. При этом имя элемента задано атрибутом `name`, а значение определено вводом или выбором пользователя.
2. Полученный набор данных кодируется в соответствии с типом файла, заданным атрибутом `enctype`.
3. Закодированные данные передаются серверу в соответствии с указанием атрибута `method`:
   - если `method="get"`, то к URL, заданному атрибутом `action`, добавляется символ `?` и набор данных формы в виде пар "`имя элемента=значение`", разделенных символом `&` и закодированых в соответствии с типом файла `application/x-www-form-urlencoded`. Полученная таким образом строка передается Web-серверу. В этом случае допустимый объем данных ограничен;
   - если `method="post"`, то браузер пересылает данные по адресу URL, заданному в атрибуте `action`, формируя тело сообщения в соответствии с типом файла, указанным в `enctype` (обычно это тип `text/plain` или же `multipart/form-data`, если из формы выполняется загрузка файлов на сервер).

Например, заголовок формы поисковой машины Яndex по адресу **http://www.ya.ru**, имеет вид

```
<form name="Web" method="get" action="http://www.yandex.ru/yandsearch">.
```

После ввода пользователем запроса `test` в единственное поле формы, имеющее атрибут `name="text"`, данные будут отправлены в URL запроса строкой вида

```
http://www.yandex.ru/yandsearch?text=test.
```

Заголовок формы, отправляющей информацию на адрес E-mail, может иметь вид

```
<form action="mailto:pers@mail.ru?subject=Internet-Test" method="post"
enctype="text/plain" name="Q" onsubmit="return Validate()">.
```

Здесь форма имеет наименование `Q`, в качестве `action` указана отправка сообщения по E-mail на адрес **pers@mail.ru** с темой `Internet-Test`, атрибут `enctype` предписывает отправлять данные как неформатированный текст, а атрибут `onsubmit` связывает отправку формы с функцией `Validate()`, написанной на JavaScript. Подобные функции, отвечающие за проверку правиль-

ности ввода, широко применяются для предотвращения отправки ошибочных данных.

Основной тег для организации *элемента* формы — унарный тег `<input>`. В зависимости от значения своего атрибута `type`, он предназначен для сбора информации различными способами, включая текстовые поля, поля для ввода пароля, переключатели, флажки, кнопки для отправки данных и очистки формы. Рассмотрим основные возможности этого тега.

Создать однострочное поле ввода можно тегом вида

```
<input type="text" name="текст" maxlength="число" size="число"
value="текст">.
```

Атрибут `maxlength` ограничивает максимальную длину *вводимого* текста, а `size` показывает максимальное число *отображаемых* символов. Атрибут `value` указывает исходное значение поля ввода. Например, тег вида

```
Введите Ваше имя: <input type="text" name="name" maxlength="20" size="10"
value="">
```

создает поле ввода с разрешением вводить до 20 символов и отображаемой длиной 10 символов. Изначально после пусто, а для внешних программ оно называется `name`.

Текстовое поле с защитой вводимых символов (т. е. с заменой их на звездочки, как принято при вводе паролей) можно создать, если вместо атрибута `type="text"` указать `type="password"`. Остальные атрибуты поля ввода пароля те же самые.

Следует иметь в виду, что поле `password` само по себе *не обеспечивает* защиту передаваемых данных и при передаче данных методом `get` они будут отображаться в адресной строке браузера.

Для ввода строки текста существует также устаревший тег `<isindex prompt="текст">`. Использовать его не следует.

Флажок выбора, имеющий состояния "включено" и "выключено", создается тегом вида

```
<input type="checkbox" name="текст" value="текст" checked>.
```

Атрибут `value` определяет значение, возвращаемое программе обработки данных при выборе пользователем флажка, необязательный атрибут `checked`, если он указан, делает флажок выбранным по умолчанию. Например, тег

```
Вы согласны с условиями договора? <input type="checkbox" name="accept"
value="yes" checked>
```

создаст включенный по умолчанию флажок, данные от которого поступят программе обработки в виде "`accept=yes`". Если флажок не был включен, значение атрибута `value` не передается.

Переключатели подобны флажкам, но отличаются тем, что если несколько переключателей имеют одно и то же значение атрибута `name`, то выбрать можно только один из них. В операционной системе Windows такие переключатели называют *радиокнопками*. Опции `value` и `checked` радиокнопки имеют те же значения, что и у флажка:

```
Выберите Вашу возрастную категорию:
<br><input type="radio" name="age" value="till 18">До 18
<br><input type="radio" name="age" value="till 60" checked>От 18 до 60
<br><input type="radio" name="age" value="after 60" >За 60
```

Здесь формируется набор из трех радиокнопок, вторая из них выбрана по умолчанию. Если она останется выбранной, программа обработки получит данные от переключателя в виде "`age=till 60`". С помощью тега `<br>` каждая радиокнопка выведена на экране с новой строки.

В наших примерах мы используем обычный текст для создания подписей к полям ввода и переключателям. В стандарте HTML существует и специальный тег `<label>`, создающий *текстовую метку* для элемента формы. Его содержимым могут быть любые текстовые элементы, кроме другой метки. Он может содержаться в любых блочных и текстовых элементах, кроме кнопки `<button>`.

Каждый элемент `<label>` связывается только с одним элементом формы. Это может быть сделано двумя способами:

- в атрибуте `for` элемента `<label>` указывается значение атрибута `id` соответствующего ему управляющего элемента, например, `<label for="fname">Введите Ваше имя</label><input type="text" name="firstname" id="fname">`;
- управляющий элемент становится содержимым тега `<label>`, например `<label>Введите Ваше имя <input type="text" name="firstname"></label>`.

Существует три типа *кнопок*, задаваемых тегом `<input>`.

Сначала приведем пример создания кнопки для очистки формы:

```
<input type="reset" value="текст">.
```

Тип кнопки назначается атрибутом `type="reset"`, а значение атрибута `value` позволяет указать надпись для нее. После очистки все элементы формы принимают значения, которые они имели по умолчанию.

Кнопку для отправки данных формы приложению-обработчику можно создать так:

```
<input type="submit" value="текст">.
```

Здесь тип кнопки задается атрибутом `type="submit"`, атрибут `value` также позволяет указать надпись на ней.

Для создания произвольной кнопки общего назначения вместо `reset` или `submit` указывают значение `button`. В основном это требуется при написании скриптов, работающих на стороне клиента. Например, кнопка, созданная тегом `<input type="button" value="Вывести" onclick="Look()">`, подписана словом "Вывести", а при нажатии на нее вызывается функция сценария с именем `Look()`.

Распространенной комбинацией кнопок является следующее указание:

```
<input type="submit" value="Отправить">
<input type="reset" value="Очистить">.
```

Кнопкам также можно назначать атрибут `name`, что позволит программе-обработчику определить, какая из кнопок отправки данных была нажата, если их несколько.

Альтернативой служит создание кнопки тегом вида

```
<button type="тип кнопки" name="текст" value="текст">…</button >,
```

где параметр `type` принимает одно из значений "`submit`", "`reset`" или "`button`". В отличие от `<input>`, тег `<button>` контейнерный и может содержать любые текстовые и блочные элементы, кроме других элементов форм. Это позволяет, например, комбинировать на одной кнопке текст и графику. Сам тег `<button>` также может находиться в любых текстовых и блочных элементах, кроме другого элемента `<button>`.

Особый вид кнопок представляет собой *графическая кнопка пересылки*, создаваемая тегом вида

```
<input type="image" src="URL" alt="альтернативный текст надписи">.
```

При щелчке мышью по такой кнопке происходит пересылка формы, причем координаты курсора мыши в момент щелчка передаются серверу в виде `name.x=x` и `name.y=y`, где `name` — имя кнопки, а `x` и `y` — координаты в пикселах относительно левого верхнего угла картинки. С помощью атрибута `usemap` можно также указать на клиентскую карту ссылок. Его значением должно быть имя закладки, заданное атрибутом `name` соответствующего тега `<map>` (см. *главу 11*).

Управляющий элемент `<input type="file">` позволяет пользователю *выбрать файл* для передачи его содержимого серверу при пересылке формы. Из дополнительных атрибутов обычно действуют только `size` и `maxlength` — как ограничения на длину поля ввода, но не содержимого файла. Браузером этот тег отображается в виде поля ввода с системной кнопкой **Обзор**, позво-

ляющей вызвать диалог выбора файла. Управлять видом кнопки из HTML нельзя.

Форма, содержащая элемент выбора файлов, должна иметь атрибуты `method="post"` и `enctype="multipart/form-data"`.

Из *общих атрибутов* тега `<input>` отметим атрибут `readonly`, имеющий смысл для полей ввода текста и паролей. Он разрешает только чтение, но не изменение элемента и целесообразен для *вывода* данных. Существует также атрибут `disabled`, *запрещающий* данный управляющий элемент. Это означает, что компонент не может получить фокус и передать данные. Атрибут может использоваться программами для временного отключения нужного компонента формы.

Тег `<input>`, как и рассмотренные далее альтернативные средства, поддерживает обработку следующих нестандартных событий:

- `onfocus` — происходит, когда компонент формы получает фокус;
- `onblur` — происходит, когда компонент теряет фокус;
- `onselect` — происходит, когда выбран текст (для текстовых объектов);
- `onchange` — происходит, когда компонент теряет фокус, и его значение изменилось с момента предыдущего получения фокуса (для изменяемых пользователем элементов).

Увидеть поля ввода, переключатели и кнопки "в действии" вы можете, открыв файл Glava_13\Form1.html с компакт-диска. Изучите исходный код этого примера.

Кроме тега `<input>`, существуют и другие возможности для создания интерфейсных элементов.

Не всегда удобно вводить данные только в одну строку. При добавлении сообщений в гостевую книгу или форум естественней выглядит *многострочное поле ввода*. Оно создается тегом вида

```
<textarea name="текст" cols="число" rows="число">...</textarea>.
```

Атрибуты `cols` и `rows` указывают видимое число строк и столбцов в поле. В отличие от других тегов ввода, текст, написанный между открывающей и закрывающей частью тега `<textarea>`, становится его значением по умолчанию.

Атрибуты `cols` и `rows` являются скорее рекомендацией, чем ограничением. Как правило, пользователь может ввести больше текста, чем позволяют указанные атрибуты, а браузер при необходимости обеспечивает прокрутку поля ввода. В зависимости от конкретного обозревателя возможна как горизонтальная и вертикальная прокрутка, так и только вертикальная с автоматическим разрывом слишком длинных строк. Теоретически браузеры могут обес-

печивать ввод текста любого размера, на практике этот объем обычно ограничен и составляет у современных обозревателей не менее 32 Кбайт. Разумеется, серверная программа обработки данных может "отказаться" обрабатывать слишком длинный текст. К тегу применимы атрибуты `name`, `readonly` и `disabled`. Некоторые браузеры поддерживают также нестандартный атрибут `wrap`, управляющий автоматической разбивкой передаваемого текста на строки. Он может принимать значения `off` (выключен), `virtual` (текст распределяется по полю ввода, но передается на сервер как одна строка) и `physical` (текст передается так, как отображается).

Контейнерный тег `<select>…</select>` создает компонент формы, предназначенный для *выбора значений из списка*. В зависимости от указания атрибута `size`, этот элемент отображается в виде многострочного списка или выпадающего меню. Его содержимым могут быть только элементы `<optgroup>` и `<option>`, определяющие элементы списка. Тег `<select>` может содержаться в любых блочных и текстовых тегах, кроме `<button>`.

Применение `<select>` удобно, когда требуется выбрать одно или несколько значений из большого количества вариантов. Те же возможности, запрограммированные с помощью радиокнопок и переключателей, потребовали бы значительно больше места на экране.

Атрибут `size="число"` задает число видимых строк списка. Для создания выпадающего меню, как правило, указывают значение `size="1"`.

Атрибут `multiple` разрешает выбор сразу нескольких значений из списка. Это можно сделать с помощью мыши, удерживая нажатой клавишу <Ctrl>. Все выбранные элементы в этом случае передаются на сервер отдельными парами "`имя=значение`", где строка "`имя`" задается атрибутом `name` тега `<select>`, а строка "`значение`" — атрибутом `value` тега `<option>`. При указании атрибута `multiple` целесообразно задавать значение `size`, большее единицы.

Тег может также содержать атрибуты `name` и `disabled`.

Отдельный элемент списка создается тегом `<option>`, закрывать который не обязательно. В открывающей части тега могут быть указаны следующие атрибуты:

- `value="текст"` — значение, возвращаемое на сервер при выборе элемента;
- `selected` — определяет, что элемент выбран заранее; для списка с указанным атрибутом `multiple` таких пунктов может быть несколько;
- `label="текст"` — задает текстовую метку элемента списка. Поддержка браузерами этого атрибута пока ограничена, и вместо него вслед за тегом `<option>` следует записывать обычный текст.

Теги `<option>` можно группировать текстовым тегом `<optgroup>...</optgroup>`. Атрибут `label="текст"` элемента `<optgroup>` задает текстовую метку группы, которая может выводиться браузером. Современные браузеры поддерживают этот тег, позволяющий сделать список более наглядным.

Создание списка значений и многострочного поля ввода иллюстрирует код листинга 13.1, а их внешний вид — рис. 13.1. На компакт-диске примеру соответствует документ Glava_13\Form2.html. Пример выпадающего списка-меню приведен в *разд. 13.3*.

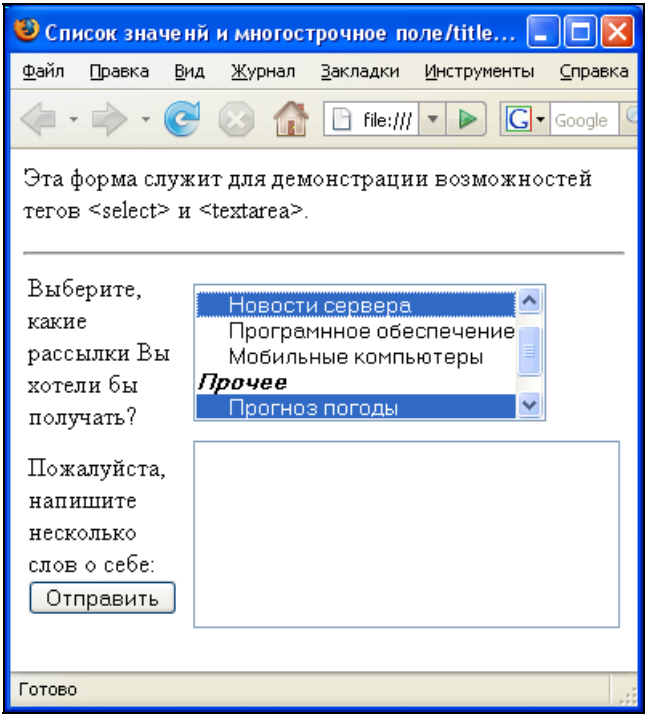

Рис. 13.1. Список значений и многострочное поле

Листинг 13.1. Список значений и многострочное поле

```
<html><head>
  <title>Список значений и многострочное поле</title>
</head>
<body bgcolor="#FFFFFF" text="#000000">
<p>Эта форма служит для демонстрации
```

```
возможностей тегов &lt;select&gt; и &lt;textarea&gt;.
<hr>
<form action="" enctype="text/plain" method="post">
<table border="0"><tr><td>
  Выберите, какие рассылки Вы хотели бы получать?
 </td><td>
  <select name="news" size="5" multiple>
   <optgroup label="Технические новости">
    <option value="server" selected>Новости сервера
    <option value="soft">Программное обеспечение
    <option value="mobile">Мобильные компьютеры
   </optgroup>
   <optgroup label="Прочее">
    <option value="weather" selected>Прогноз погоды
    <option value="tv">Программа ТВ
   </optgroup>
  </select>
 </td></tr><tr><td>
  Пожалуйста, напишите несколько слов о себе:
  <br><input type="submit" value="Отправить">
 </td><td>
  <textarea name="msg" cols="30" rows="6"></textarea>
 </td></tr>
</table></form>
</body></html>
```

Для выравнивания компонентов здесь использована таблица с невидимыми границами, встроенная в форму. Обратите внимание, что содержимое метки тега `<optgroup>` также занимает одну строку списка. Как правило, браузеры автоматически добавляют к спискам вертикальные полосы прокрутки и подбирают ширину списка по самой длинной его строке.

Группировать можно не только строки списков, но и любые компоненты формы. Визуально сгруппированные элементы заключаются в рамку, цвет которой выбирается браузером. В верхней части рамки может быть выведена текстовая подпись. Группировка компонентов и разбиение их на отдельные логические блоки облегчают пользователю заполнение формы и делают ее более наглядной. Существуют всего два тега группировки:

- `<fieldset>…</fieldset>` — *определяет* группу элементов. Внутри этого тега помещаются любые компоненты формы;

- `<legend>…</legend>` — задает *заголовок* группы. Он может содержать любые текстовые элементы и употребляется только первым тегом внутри `<fieldset>`.

Эти теги поддерживаются современными браузерами, пример их использования рассмотрен в *разд. 13.3*.

В заключение опишем еще один компонент формы, который не отображается на экране, но без которого трудно наладить гибкое взаимодействие с серверными программами:

`<input type="hidden" name="текст" value="текст">`.

Этот тег позволяет передать обработчику формы произвольную строку данных вида "`имя=значение`" и применяется, в основном, в двух целях:

- для передачи серверным программам дополнительных параметров, влияющих на режим их работы;
- для передачи информации от одной Web-страницы к другой, если пользователь заполняет последовательно несколько форм.

Сам по себе тег `<hidden>` не обеспечивает какой-либо защиты данных, и его значение может быть легко получено из исходного кода страницы. Необходимость его использования связана с тем, что сервер обрабатывает каждый последующий запрос независимо от предыдущего и без дополнительной передачи данных с предыдущего шага работы не будет "знать", как с ним связан текущий шаг.

13.3. Отправка данных по электронной почте и внешнему приложению

Как уже описывалось ранее, для отправки данных по указанному в заголовке формы адресу E-mail достаточно задать в открывающей части тега `<form>` атрибут вида `action="mailto:адрес E-mail"`. При желании можно автоматически добавить тему письма, добавив после адреса строку вида "`?subject=тема письма`". В заголовке формы должен быть указан метод отправки данных `method="post"`. Дальнейшие события зависят от того, установлена ли на машине пользователя почтовая программа-клиент. Если это так, то информация будет отправлена, причем данные из полей формы придут в виде строк текста `имя=значение`, где `имя` задано атрибутом `name`, а `значение` — атрибутом `value` или его аналогом. Пустые или незаполненные поля формы не помешают отправке письма. Для обработки этой ситуации и предотвращения пересылки неправильных данных следует включить в документ функцию про-

верки, написанную на JavaScript или другом языке сценариев. Если на машине не установлена программа электронной почты, операционная система предупредит об этом в окне диалога. Если почтовая программа установлена, но нет активной учетной записи, обычно вызывается программа-мастер для ее настройки. Современные браузеры, как правило, дополнительно предупреждают пользователя о попытке формы отправить данные по E-mail. Это делается из соображений безопасности, отключить вывод такого предупреждения нельзя.

Листинг 13.2 содержит HTML-код формы для "обратной связи" с пользователем посредством E-mail, а ее внешний вид иллюстрирует рис. 13.2. Перед отправкой формы функция `Check()`, написанная на JavaScript, выполняет проверку корректности данных. Не вдаваясь в детали этого языка, заметим, что вы можете использовать предлагаемую форму в качестве образца на своих Web-страницах. Исходный код примера, который можно открыть в своем браузере, находится на компакт-диске в документе Glava_13\MailForm.html.

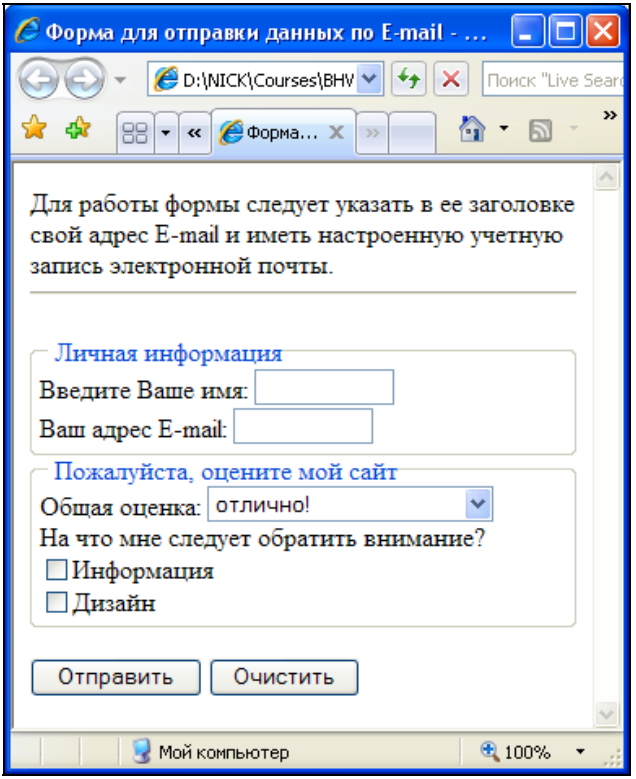

Рис. 13.2. Форма для отправки данных по E-mail

Листинг 13.2. Форма для отправки данных по E-mail

```
<html><head>
  <title>Форма для отправки данных по E-mail</title>
<script language="javascript">
 //Функция для проверки формы
 function Check() {
  if (document.form1.name.value=="" ||
      document.form1.mail.value==="") {
   window.alert ("Пожалуйста, заполните текстовые поля перед отправкой");
   return false;
  }
  return true;
 }
</script>
</head>
<body bgcolor="#FFFFFF" text="#000000">
<p>Для работы формы следует указать в ее заголовке
свой адрес E-mail и иметь настроенную учетную запись
электронной почты.
<hr>
<form name="form1" action="mailto:pers@mail.ru?subject=Data from page"
method="post" enctype="text/plain">
 <fieldset>
  <legend>Личная информация</legend>
  <label>
   Введите Ваше имя:
   <input type="text" name="name" maxlength="20" size="10" value="">
  </label>
  <label>
   <br>Ваш адрес E-mail:
   <input type="text" name="mail" maxlength="20" size="10" value="">
  </label>
 </fieldset>
 <fieldset>
  <legend>Пожалуйста, оцените мой сайт</legend>
  <label>
   Общая оценка:
   <select name="ball" size="1">
    <option value="5" selected>отлично!</option>
    <option value="4">хорошо для начала</option>
```

```
    <option value="3">нужно еще поработать</option>
   </select>
  </label>
  <label>
   <br>На что мне следует обратить внимание?
   <br><input type="checkbox" name="information" value="yes">Информация
   <br><input type="checkbox" name="design" value="yes">Дизайн
  </label>
 </fieldset>
 <br><input type="submit" value="Отправить" onclick="return Check()">
 <input type="reset" value="Очистить">
</form></body></html>
```

Для визуального выравнивания полей формы и подписей мы могли бы поместить ее элементы в ячейки таблицы. Обратите внимание, что тег `<label>` никак не структурирует свое содержимое, и переход к новой строке реализован с помощью `<br>`. Некоторые браузеры до сих пор игнорируют атрибут `label` тега `<option>`, поэтому он также не задействован.

Отправка данных *внешнему приложению*, расположенному на сервере Интернета, требует вставки на страницу формы, с которой приложение принимает данные.

В Интернете достаточно много открытых Web-программ, разрешающих обращаться к ним с данными пользовательской формы. При этом информация может отправляться как методом `get`, так и `post`. Обычно на странице помощи открытого приложения приведен полный код пользовательской формы и требования к ее размещению у себя на сайте. Как правило, разработчики требуют полной неизменности кода формы, но на практике можно вносить в него незначительные изменения, касающиеся, например, стилевого оформления элементов (см. *главу 14*). Естественно, атрибуты `name` и типы интерфейсных элементов формы должны быть указаны корректно.

В качестве примера мы рассмотрим организацию поиска на сайте с помощью популярной поисковой машины Яndex. Для успеха поиска вы должны корректно указать URL главной страницы вашего сайта, и он должен быть хотя бы раз *проиндексирован* с помощью Яndex. Информацию об индексировании сайтов можно получить из *разд. 16.4*. Листинг 13.3 содержит код примера поисковой формы, а рис. 13.3 иллюстрирует ее внешний вид. Информация о URL сайта, по которому выполняется поиск, и его названии передается через скрытые поля с атрибутом type="`hidden`". Вторая пара скрытых полей служит для того, чтобы вернуться на свой сайт из результатов поиска. Открыть этот пример в браузере вы можете, запустив документ Glava_13\SearchForm.html с компакт-диска.

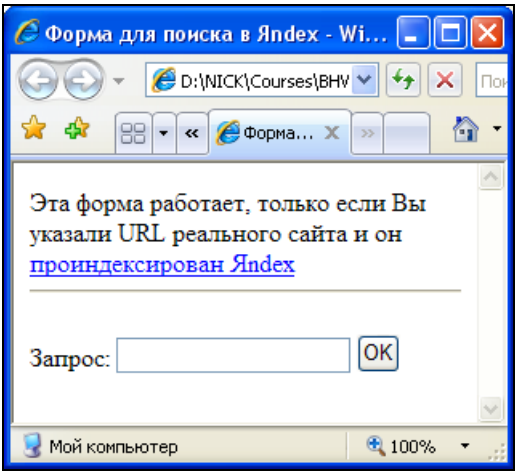

Рис. 13.3. Форма для поиска по сайту с помощью Яndex

Листинг 13.3. Форма для поиска по сайту с помощью Яndex

```
<html>
<head>
 <title>Форма для поиска в Яndex</title>
</head>
<body bgcolor="#FFFFFF" text="#000000">
<p>Эта форма работает, только если Вы указали
URL реального сайта и он <a href="http://Webmaster.yandex.ru/"
target="_blank">проиндексирован Яndex</a>
<hr>
<form name="Web" method="get" action="http://www.yandex.ru/yandsearch">
Запрос: <input type="text" name="text" size="20" value="" max-
length="100">
<input type="submit" value="OK">
<!-- замените URL и название сайта на свои -->
<input type="hidden" name="serverurl" value="http://www.narod.ru">
<input type="hidden" name="server_name" value="Мой сайт">
<!-- 2 строчки ниже позволяют вернуться на сайт из результатов поиска -->
<input type="hidden" name="referrer1" value="http://www.narod.ru">
<input type="hidden" name="referrer2" value="Мой сайт">
</form>
</body></html>
```

13.4. Дизайн форм

Мы уже научились создавать формы, теперь имеет смысл кратко обсудить, как сделать их более удобными и привлекательными. Вот несколько рекомендаций.

- Главная идея формы — простота и понятность для пользователя. Человека не так уж легко заставить что-то заполнять, даже если он — ваш потенциальный клиент, готовый потратить пару минут на составление заказа. Поэтому самое главное должно "бросаться в глаза" с первого взгляда. Хорошо, если перед формой приведена краткая аннотация, описывающая ее назначение и пользу, которую можно извлечь, заполнив ее.
- По возможности, форма должна быть короткой. Если этого достичь нельзя, разбивайте ее на небольшие блоки, хотя бы горизонтальными чертами `<hr>`.
- Форма должна быть узнаваемой. Плохо, когда кнопок отправки данных и отмены нет внизу, на их привычном месте. Формы — не тот элемент сайта, который должен удивлять нестандартными решениями.
- Выделяйте ключевые метки и заголовки тегами форматирования. Это улучшит восприятие и позволит пользователю избежать ошибок.
- Разделяйте элементы формы визуально. Для этого подойдет как невидимая таблица, так и теги абзаца или перевода строки. Неплохо, если элементы формы сгруппированы в два-три логически понятных блока. Не следует размещать на одной строке более одного интерфейсного элемента и метки к нему.
- Исключайте лишнее. Так ли вам нужна ссылка на домашнюю страницу пользователя? А может быть, можно обойтись без "Комментариев"?
- Исключайте повторное заполнение. К сожалению, средствами HTML это реализовать непросто, можно лишь использовать сценарий для проверки данных, как это сделано в листинге 13.2. Но ваша серверная программа должна быть в состоянии *вернуть* введенные пользователем данные обратно в форму, даже если в некоторых полях обнаружены ошибки. Ничто так не раздражает, как необходимость повторно заполнять девять полей из-за одного заполненного неправильно.
- Не запрашивайте заведомо приватную информацию — для этого есть защищенные соединения, но не протокол HTTP.
- Чтобы приблизить форму к общему дизайну страницы, применяйте стили, не нарушая однозначность восприятия элементов. Поле ввода, у которого почти не видно границ, способно смутить даже опытного пользователя.

Глава 14

Таблицы стилей — управляем сайтом легко

Современные текстовые процессоры, например, Microsoft Word, позволяют пользователю определять *стиль*, т. е. набор правил оформления и форматирования, который может применяться к различным частям документа. С другой стороны, в стандартном HTML для присвоения элементу определенных свойств (цветов, размеров, положения на странице) их приходится каждый раз описывать заново, даже если в документе находится 10 или 100 таких элементов.

Каскадные таблицы стилей (Cascading Style Sheets, сокращенно CSS) предназначены для хранения информации о стилях Web-страниц. Механизм CSS позволяет отделить *форматирование* HTML-документа от его *содержания* и, таким образом, экономично и удобно использовать однажды созданные правила для оформления любого количества документов.

В настоящее время CSS — одна из самых перспективных технологий, облегчающих и автоматизирующих разработку Web-сайтов. В этой главе мы научимся создавать стили и применять их к своим страницам. Стандарты CSS стремительно развиваются, поэтому мы рассмотрим только устоявшееся "ядро" языка, одинаково хорошо понимаемое всеми современными браузерами.

14.1. Общие свойства таблиц стилей

Итак, CSS можно определить как набор стилевых правил для дополнительного форматирования тегов HTML.

Хотя таблицы стилей часто сравнивают со стилевыми указаниями Microsoft Word, между ними нет прямой аналогии. Прежде всего, это связано с тем, что установки CSS действуют на *все* элементы, заключенные внутри тега, которому присвоен стиль. Так, описав некоторый стиль для тега `<body>`, мы автоматически присваиваем его всему содержимому Web-страницы. В объектно-

ориентированном программировании (ООП) такое поведение объектов называется *наследованием*. С другой стороны, стилевые указания обладают и свойством, называемым в ООП *полиморфизмом* — вложенные элементы могут *переопределять* свойства элементов, в которые они вложены, формируя таким образом собственные стили.

Чтобы лучше понять структуру документа с точки зрения таблиц стилей, рассмотрим хорошо нам знакомый простейший пример:

```
<html><head>
    <title>Заголовок окна</title>
</head><body>
    <h1>Заголовок первого уровня</h1>
    <p>Абзац текста</p>
</body></html>
```

Теперь представим себе этот документ в виде *дерева* с единственным *корнем* — элементом `<html>` и двумя *дочерними элементами* — вложенными в `<html>` тегами `<head>` и `<body>`. Элемент `<head>` имеет дочерний элемент `<title>`, для которого он сам является *родителем*. Для элемента `<html>` элемент `<title>` будет *потомком*, а сам элемент `<html>` — *предком* `<title>`. Элемент `<body>` имеет дочерние элементы `<h1>` и `<p>`, являющиеся между собой *сестрами* (или братьями, если кому-то так больше нравится). При этом тег `<h1>` — *предшествующий* элемент, а `<p>` — *следующий*.

Как видим, все достаточно просто. Важно, что каждый тег нашего документа имеет только одного родителя, за исключением корневого элемента `<html>`, у которого родителя нет. Любой элемент в этой иерархии может либо *определить* собственный стиль, либо *унаследовать* его от родителя или от более отдаленного предка. Легко увидеть, что при этом возможны достаточно сложные наложения правил и некоторые из них могут противоречить друг другу. В CSS предусмотрена и такая возможность, ведь таблицы стилей не случайно названы *каскадными*.

Это означает, что каждому правилу приписан определенный *вес*. Если к конкретному элементу применимы несколько правил, используется то из них, которое имеет больший вес. В результате происходит накопление свойств элементов в соответствии с правилами наследования, и, тем самым, образуется *каскад свойств*, распространяющийся от предков к потомкам. Не вдаваясь в детали механизма наследования, заметим, что наибольший вес имеют стилевые правила, заданные *явно* для текущего элемента, затем учитывается значение атрибута, которое может быть *унаследовано* от родительского элемента или от вышестоящих предков, наконец, в последнюю очередь действует значение по умолчанию.

Так, если для элемента `<body>` в стиле определен синий цвет, а для элемента `<p>` — зеленый, то цвет выделенного текста в конструкции тегов

```
<p>Стили – это <strong>важно</strong>!</p>
```

зависит от того, был ли в стиле задан цвет для тега `<strong>`. Если это так, заданный цвет будет применен, в противном случае слово `важно` будет выведено тем же *зеленым* цветом, что и весь абзац, поскольку тег `<p>` — потомок `<body>`, переопределивший его атрибут цвета с синего на зеленый.

Независимо от способа задания, таблица стилей состоит из набора *операторов*. При этом каждый оператор представляет собой либо *директиву*, либо *правило*. Операторы могут разделяться пробелами или символом перевода строки.

Директива начинается с символа `@` (at) и следующего за ним без пробела ключевого слова. Назначение директив — описания шрифтов, кодировок символа, печатных страниц и другого *внешнего* по отношению к стилевому форматированию содержимого.

Правило состоит из имени-*селектора* и *блока деклараций*, заключенного в фигурные скобки, например:

```
p {
  font-size: 12pt;
  font-family: Arial;
  font-style: normal
}
```

Здесь `p` — это *селектор*, за которым следует блок из трех *деклараций*, каждая из которых состоит из *названия свойства* (например, `font-size`), символа двоеточия и *значения свойства* (например, `12pt`). Декларации должны разделяться символом точки с запятой, построчное их расположение не имеет значения, но удобнее, когда на одной текстовой строке находится только одна.

Согласно определению CSS, все его элементы не зависят от регистра символов.

Подобно документам HTML, таблицы стилей могут содержать *комментарии*, которые служат для пояснения операторов. Комментарий начинается с символов `/*` и заканчивается символами `*/`. Вложенные комментарии не допускаются.

14.2. Способы определения стилей

Существует три основных способа применения таблицы стилей к документу:

1. *Связывание* (linking) — можно связать HTML-документ с таблицей стилей, хранящейся в отдельном файле.
2. *Внедрение* (embedding) — можно встроить таблицу стилей в документ с помощью тега `<style>`, указанного в заголовке документа `<head>`.

3. *Задание стиля для отдельного элемента* (inline) — можно определять элементы стиля "на лету", т. е. указывать их в тегах документа, например, в теге абзаца `<p>`.

Наиболее перспективен первый способ, называемый также *внешними* таблицами стилей. Очевидно, что он позволяет применить однажды созданный и сохраненный в файле стиль ко всем страницам вашего сайта. Хранить таблицу стилей следует в текстовом файле с расширением css, который можно создать при помощи любого текстового редактора, например, Блокнота Windows. При сохранении такого файла достаточно поменять ему тип с txt на css по алгоритму, описанному в *разд. 4.3*.

Для связывания таблицы стилей с документом HTML в последнем записывается тег следующего вида:

```
<link rel="stylesheet" type="text/css" href="URL">.
```

Здесь атрибут `URL`, как всегда, обозначает абсолютный или относительный адрес стилевого файла, а остальные атрибуты служат для того, чтобы сообщить браузеру, что на странице используется CSS. Обычно этот тег располагают в теге `<head>` или между тегами `<head>` и `<body>`.

К внешним таблицам стилей следует отнести и директиву `@import`, позволяющую включать в таблицу стилей другие ранее подготовленные таблицы. Она должна содержать абсолютный или относительный URL импортируемой таблицы и может задаваться в одном из следующих видов:

```
@import "mystyle.css";
@import url(mystyle.css);
```

В случае, когда разработанные стили действуют только в одном документе, первый способ можно заменить вторым, называемым *внутренними* таблицами стилей.

При этом указывать тег `<link>` не нужно, а таблица стилей будет описана непосредственно внутри документа в теге `<style>`. Общий вид этого описания показан в листинге 14.1.

Листинг 14.1. Общий вид таблицы стилей

```
<style type="text/css">
<!--
  операторы таблицы стилей
-->
</style>
```

Операторы таблицы заключены в тег HTML-комментария `<!--...-->` для того, чтобы "спрятать" стили от старых браузеров, которые могут показать их просто как неформатированный текст страницы. В настоящее время комментирование потеряло актуальность, т. к. браузеров, не использующих стилей, почти не осталось.

Стилевой файл CSS также имеет общий вид, показанный в листинге 14.1.

Наконец, для определения стиля конкретного тега достаточно указать в его открывающей части атрибут `style` со значением, включающим одну или несколько деклараций CSS:

```
<h1 style="color: blue">Заголовок</h1>.
```

Здесь с помощью свойства CSS с названием `color` мы определили цвет *текущего* заголовка первого уровня как синий.

Этот метод указания стиля поддерживается всеми дочерними тегами тега `<body>`.

Перечисленные три способа определения стиля упорядочены иерархически, что имеет значение при разрешении одинаковых правил, заданных на разном уровне:

- наивысший приоритет имеет оперативное указание стиля в конкретном теге;
- вторым по старшинству является указание стиля в теге `<style>`;
- низший приоритет имеет указание в стилевом файле.

Отсюда очевидно, что третий способ обычно служит лишь для "разового" переопределения свойств какого-либо тега, указанных в стилевом файле. При злоупотреблении оперативным определением стиля теряется основное преимущество CSS — разделение содержания и форматирования.

Приведем также пример, иллюстрирующий поддержку наследования в CSS:

```
<div style="color: blue">
   <h1>Это заголовок внутри раздела</h1>
   <p>Это абзац внутри раздела</p>
</div>
```

Здесь теги `<h1>` и `<p>` унаследуют синий цвет шрифта от тега `<div>`.

14.3. Виды селекторов. Определение классов

Мы уже знаем, что таблица стилей состоит из деклараций, имеющих общий вид:

```
селектор { свойство1: значение1; … ; свойствоN: значениеN }.
```

Чаще всего используется так называемый *селектор типа*, представляющий собой наименование тега HTML без символов < и > (например, `H1`, `P`, `TD`):

```
h1 { color: blue; font-size: 12pt; text-line: center }.
```

Если это определение сделано в теге `<style>` или в стилевом файле, то все заголовки первого уровня, определенные тегом `<h1>`, будут выведены синим шрифтом размером 12 пунктов с выравниванием по центру. Для прочих свойств будут действовать значения по умолчанию.

Существует также *универсальный селектор* `*`, относящийся ко всем элементам документа, но обычно его успешно заменяет `body`.

Если вы хотите определить одно и то же свойство для нескольких тегов HTML, то можно указать для них отдельные декларации:

```
P { font-size: 12pt }
UL { font-size: 12pt }
```

или более компактно перечислить селекторы через запятую:

```
P, UL { font-size: 12pt }.
```

Подобный прием называется *группированием* селекторов.

Иногда возникает необходимость определения двух и более деклараций стиля для одного тега: например, для тега элемента списка `<li>` может понадобиться один вид оформления, если он находится в нумерованном списке и другой — в маркированном. Такие указания легко сделать с помощью *контекстных определений*, задав точную последовательность тегов, для которой будет применен стиль:

```
OL LI { list-type-style: decimal }
UL LI { list-type-style: square }
```

Обратите внимание, что названия элементов *не разделены* запятыми. В противном случае всем тегам списка будет присвоен один и тот же стиль.

Контекстное определение может относиться не только к элементам-родителям, но и к более отдаленным предкам:

```
div * i { color: blue; }.
```

Данное правило будет применяться ко всем элементам `<i>`, которые находятся внутри любых элементов (универсальный селектор), непосредственно заключенных в тег `<div>`.

Селекторы контекстного определения называют также *селекторами потомков*.

Таблицы стилей позволяют также *создавать классы* — совокупности определений, каждое из которых может использоваться в нужном месте страницы. Например, вы можете определить два варианта стиля заголовка `<h1>`.

Определение вариантов отличается от указания стиля лишь тем, что к названию тега добавляется наименование класса, отделенное точкой:

```
H1.blue { color: blue }
H1.red { color: red }.
```

Теперь, включая в документ тег `<h1>`, можно указать в нем конкретный стиль при помощи атрибута `class`:

```
<h1 class=red>Красный заголовок</h1>
<h1 class=blue>Синий заголовок</h1>.
```

Рекомендуется выбирать названия классов, отражающие их назначение.

Классы могут формироваться и без привязки к конкретному тегу, в этом случае их имя начинается с символа точки, например:

```
.normal {
  font-family: Times New Roman; font-size: 12pt; color: maroon
}
```

Здесь описан класс для форматирования текста с именем `normal`. Его действие может распространяться на любые текстовые элементы страниц, для чего достаточно указать в нужном теге атрибут `class="normal"`:

```
<font class="normal">текст</font>.
```

Обратите внимание, что в атрибуте `class` HTML имя класса CSS указывается *без точки*.

CSS позволяет создавать *подмножества* значений атрибута `class`, например правило

```
P.normal.under { text-decoration: underline }
```

будет применено к элементам с атрибутом `class="normal under other"`, но не применено к элементам с атрибутом `class="normal other"`.

Селектор идентификатора в CSS имеет вид `#id`, где после символа `#` записано значение атрибута `id` некоторого элемента. Например, после декларации

```
#p1 { letter-spacing: 0.3em }
```

правило стиля будет применено к элементу с атрибутом `id="p1"`:

```
<p id="p1">Текст разрежен</p>.
```

Несмотря на то, что `id` описывается в HTML как уникальный атрибут тега, все современные браузеры покажут форматирование любого числа элементов, помеченных одинаковым `id` и связанных с соответствующим селектором идентификатора.

В спецификации CSS описаны и другие классы селекторов, но вопрос полной поддержки их браузерами пока не решен.

Основные виды селекторов стиля представлены на рис. 14.1. Внимательный анализ и чтение комментариев листинга 14.2, соответствующего рис. 14.1, помогут вам закрепить полученные знания. На компакт-диске с примерами этот документ записан под именем Glava_14\Selectors.html.

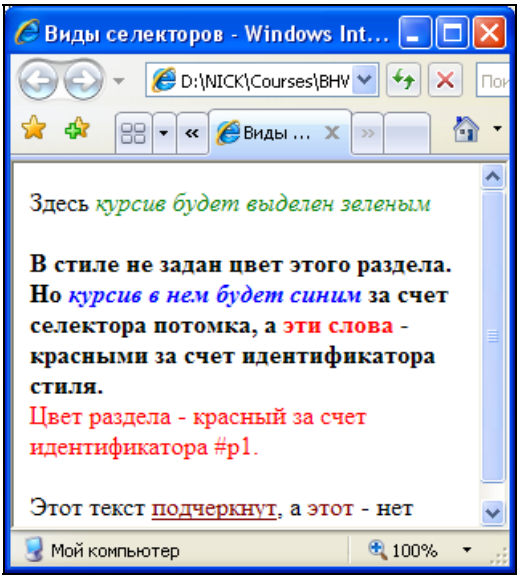

Рис. 14.1. Виды селекторов CSS

Листинг 14.2. Виды селекторов CSS

```
<html><head>
<style type="text/css">
 p { color: black; } /* черный цвет абзацев */
 p i { color : green; }   /* зеленый цвет для курсива в абзаце */
 div * i { color: blue; } /* ... и синий - в разделе внутри любого
элемента */
 #p1 { color: red } /* красный цвет для атрибута id="p1" */
 font.green { color: green; } /* два селектора класса */
 font.red { color: red; }     /* для элемента font */
 .normal { /* класс без привязки к тегу*/
  font-family: Times New Roman; font-size: 12pt; color: maroon
 }
 .normal.under { text-decoration: underline } /* подмножество класса normal */
```

```
</style>
<title>Виды селекторов</title>
</head>
<body>
 <p>Здесь <i>курсив будет выделен зеленым</i></p>
 <div><b>В стиле не задан цвет этого раздела.<br>
  Но <i>курсив в нем будет синим</i> за счет селектора потомка,
  а <span id="p1">эти слова</span> - красными за счет идентификатора
стиля.
 </b></div>
 <div id="p1">Цвет раздела - красный за счет идентификатора #p1.
 <p>Этот текст <font class="normal under">подчеркнут</font>,
 а <font class="normal">этот</font> - нет</p>
 </div></body></html>
```

Все рассмотренные селекторы основывались на *положении элементов в иерархии документа*. Для некоторых целей форматирования такого подхода недостаточно. Прежде всего, это относится к отображению *различных состояний* элемента и выделению *отдельных частей его содержимого*. Так, гиперссылка может иметь состояния "не посещена", "выделена" и "посещена", в абзацах текста может понадобиться выделить первую строку или букву и т. п.

Чтобы задействовать возможности отображения, не описываемые в терминах иерархии элементов, в CSS были включены понятия *псевдоэлементов* и *псевдоклассов*.

Разница между этими понятиями в том, что псевдоклассы применяются к *различным типам* элементов, а псевдоэлементы — к *частям* элементов, таким, как первая строка или первая буква. Селекторы псевдоклассов и псевдоэлементов записываются в виде `элемент:псевдокласс`, например, `A:link`.

Не все существующие селекторы этих видов поддерживаются современными браузерами. Основные псевдоклассы и псевдоэлементы приведены в табл. 14.1.

Таблица 14.1. *Основные псевдоклассы и псевдоэлементы*

Элемент	Описание
`:link`	Псевдокласс для непосещенной гиперссылки (применим к элементу `<a>`)
`:visited`	Псевдокласс для посещенной гиперссылки (применим к элементу `<a>`)

Таблица 14.1 *(окончание)*

Элемент	Описание
`:hover`	Псевдокласс применяется к элементу, когда пользователь навел на него курсор мыши, но не активировал щелчком
`:active`	Псевдокласс применяется к элементу, когда пользователь активировал его щелчком мыши
`:focus`	Псевдокласс применяется к элементу, когда он получает фокус ввода
`:first-line`	Псевдоэлемент применим к любому блочному элементу и задает стиль отображения его первой строки
`:first-letter`	Псевдоэлемент применим к любому блочному элементу и задает стиль отображения его первой буквы
`:before`	Псевдоэлемент используется для вставки автоматически генерируемого содержимого перед указанным элементом
`:after`	Псевдоэлемент служит для вставки автоматически генерируемого содержимого после указанного элемента

Два последних псевдоэлемента *не поддерживаются* Internet Explorer, что ограничивает их применение. Для генерации содержимого эти псевдоэлементы используют свойство `content`, а способ вставки задается значением свойства `display` (`block`, `inline` или `marker`). Например, для автоматического вывода текстовой метки в конце каждой Web-страницы, к которой применен стиль, можно указать такую декларацию:

```
BODY:after {
 content: "[конец документа]"; display: block; text-align: center;
}
```

Код примера, содержащий псевдоклассы и псевдоэлементы, приведен в листинге 14.3, а результат его исполнения — на рис. 14.2. На компакт-диске с примерами листингу 14.3 соответствует документ Glava_14\PseudoClass.html.

Листинг 14.3. Псевдоклассы и псевдоэлементы

```
<html><head>
<title>Псевдоклассы и псевдоэлементы</title>
<style type="text/css">
P:first-letter { /* Буквица для абзацев */
 font-size: X-LARGE; color: red; font-weight: 600
}
```

```
.normal:first-line {  /* Псевдоэлемент для выделения 1-й строки */
 text-transform: uppercase
}
p:hover { color:blue }  /* Выделяем активный абзац синим, */
a:hover { color:red  }  /* а активную ссылку -  красным   */
a:link { color: #00FF00 }   /* также меняем цвета обычной */
a:visited { color: #006600; /* и посещенной ссылок        */
  text-decoration: none } /* убираем у посещенной ссылки подчеркивание*/
</style>
</head>
<body bgcolor="#FFFFFF" text="#000000">
<p class="normal">Первая строка этого абзаца будет выведена большими буквами.
<p>И у этого, и у предыдущего абзаца есть буквица.
<br>Ссылки <a href="After.html">1</a>, <a href="Selectors.html">2</a>
</body></html>
```

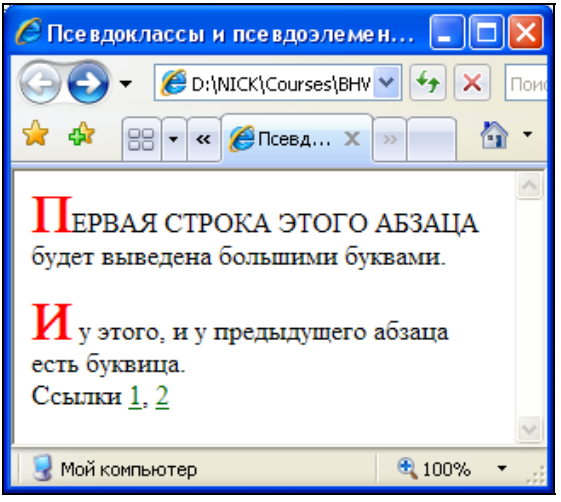

Рис. 14.2. Псевдоклассы и псевдоэлементы

При работе со стилями нам часто придется указывать *размеры* различных объектов. Все размеры в CSS задаются в виде числа, после которого следует *единица измерения*. Определенные в стандарте единицы измерения приведены в табл. 14.2.

Первые пять размеров в табл. 14.2 — *абсолютные*. Они применимы только тогда, когда известны точные физические характеристики устройства отображения, например, диагональ дисплея. Указание абсолютных размеров

в стиле может лишить пользователя возможности управлять размерами шрифта с помощью браузера, а также привести к тому, что при "слишком большом" или "слишком маленьком" разрешении монитора документ станет неудобочитаемым.

Таблица 14.2. *Единицы измерения в CSS*

Обозначение	Описание
`in`	Дюймы (1 дюйм = 2,54 см = 25,4 мм = 72 точки = 6 пик)
`cm`	Сантиметры (1 см = 10 мм = 0,39 дюйма = 2,36 пики = = 28,35 точки)
`mm`	Миллиметры (1 мм = 0,1 см = 0,039 дюйма = 0,24 пики = = 2,84 точки)
`pt`	Точки (1 точка = 1/12 пики = 1/72 дюйма = 0,035 см = = 0,35 мм)
`pc`	Пики (1 пика = 12 точек = 1/6 дюйма = 0,423 см = 4,23 мм)
`em`	Размер шрифта, равный размеру его наибольшей буквы
`ex`	Высота строчных букв шрифта
`px`	Пикселы
`%`	Проценты

Оставшиеся четыре размера — *относительные*, о них следует сказать подробней.

Единицы `em` и `ex` основаны на размере шрифта того элемента, к которому относится использующая их декларация. Величина `em` задает размер шрифта, равный размеру его наибольшей буквы, обычно это буква "M", откуда и произошло сокращение `em`. Величина `ex` интерпретируется как высота строчных букв шрифта, обычно это высота буквы "x", откуда произошло сокращение `ex`.

Указание размера в пикселах зависит как от физических размеров экрана дисплея, так и от его разрешения. Так, пиксел на экране с разрешением 640×480 всегда *больше*, чем на том же экране при установленном разрешении 1280×1024.

Процентные значения указывают на величину элемента в процентах от другой величины. Всюду, где CSS допускает проценты, в описании свойств указывается, какая величина является базовой для расчета значений в процентах. Обычно это соответствующее свойство родительского элемента.

14.4. Свойства шрифта, текста и цветов

Перейдем к более детальному рассмотрению свойств CSS. Основные свойства, применимые для управления шрифтами, приведены в табл. 14.3.

Таблица 14.3. *Свойства шрифта в CSS*

Свойство	Описание
`font-family`	Используется для указания шрифта или шрифтового семейства (см. табл. 14.4), которым будет отображаться документ или его часть. Задает список имен шрифтов и/или семейств, элементы списка разделяются запятыми, наименования шрифтов располагаются в порядке предпочтения
`font-style`	Задает *стиль шрифта* для отображения содержимого. Допустимы значения `normal` (обычный шрифт), `italic` (курсив), `oblique` (наклонный шрифт)
`font-variant`	Задает *вариант шрифта* для отображения содержимого. Допустимы значения `normal` (обычные буквы), `small-caps` (малые прописные)
`font-weight`	Определяет толщину линий (насыщенность) шрифта, имеет значения `lighter`, `normal`, `bold` и `bolder`. Можно задавать также числами `100`, `200`, `...`, `900`. Значение `400` примерно соответствует нормальной насыщенности, а 700 — начертанию, создаваемому тегом `<b>`
`font-size`	Задает *размер шрифта* для отображения содержимого элемента. Размеры могут быть абсолютными и относительными величинами, либо одним из значений `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large` (от наименьшего к наибольшему), либо одним из относительных указаний `smaller` (меньше шрифта родительского элемента) и `larger` (больше шрифта родительского элемента)
`font`	Является сокращением для свойств `font-style`, `font-variant`, `font-weight`, `font-size`, `font-family`, `line-height` и позволяет задать все свойства шрифта одновременно, например `P { font: bolder 12pt Arial, serif }`

В стандартах существуют также следующие свойства:

- `font-stretch` — для определения *выключки* шрифта, т. е. интервала между символами при отображении содержимого;
- `font-size-adjust` — для определения *аспекта* шрифта, т. е. соотношения размеров прописных и строчных букв.

В настоящее время они не поддерживаются.

Свойства, перечисленные в табл. 14.3, применимы ко всем элементам и наследуются дочерними элементами. Указание размеров в процентах допустимо только для свойства `font-size`. Приведем пример:

```
P { font-family: "Times New Roman", serif}.
```

Изучив в *разд. 5.2* тег `<font>` с атрибутом `face`, мы уже знакомы с приемом последовательного указания нескольких шрифтов, желаемых для отображения. Здесь в качестве приоритетного шрифта абзацев указан Times New Roman, при его отсутствии документ будет отображаться любым подходящим шрифтом *семейства* Serif. *Семейства шрифтов* или *родовые шрифты* были введены в стандарт на тот случай, если на компьютере-клиенте не установлен ни один из шрифтов, заданных автором. Тогда браузер использует родовой шрифт, начертание которого *напоминает* авторский. В CSS определены семейства шрифтов, описанные в табл. 14.4.

Таблица 14.4. *Семейства шрифтов в CSS*

Семейство	Описание
`serif`	Пропорциональные шрифты с засечками на буквах. Примеры: Times New Roman, Bodoni, Garamond
`sans-serif`	Пропорциональные шрифты без засечек на буквах. Примеры: Arial, Verdana, Helvetica, Tahoma
`cursive`	Каллиграфические шрифты, стилизованные под рукописный текст с типичными для него соединениями между буквами. Примеры: Script, Zapf-Chancery
`fantasy`	Шрифты декоративного характера. Примеры: Western, Comic
`monospace`	Моноширинные или телетайпные шрифты с одинаковой шириной печатаемых символов. Примеры: Courier New, Prestige, Everson Mono

Как и любые значения атрибутов, состоящие из нескольких слов, подобные имена шрифтов нужно заключать в кавычки.

Обратите внимание, что если такое определение, как `font-family: "Times New Roman"` будет сделано при указании шрифта внутри тега, то у атрибута `style` и свойства шрифта `font-family` должны быть *разные* типы кавычек:

```
<p style="font-family: 'Times New Roman'; color: black ">…</p>.
```

В противном случае, "запутавшись" во вложенных кавычках, браузер может отобразить документ с ошибками. Приведем еще один пример:

```
P { font-size: 12pt }

.little { font-size: 80% }.
```

Здесь для отображения текста абзацев установлен шрифт размером 12 пунктов, а также описан класс `little`, размер шрифта которого составляет 80% от базового.

В стандарте CSS имеется также директива `@font-face`, предназначенная для загрузки внешнего шрифта из Интернета. Например, указание ее в виде

```
@font-face { font-family: MyFont; src: http://www.fonts.com/MyFont.eot }
```

приведет к загрузке шрифта с указанного URL. При этом шрифт должен храниться на сервере в формате Embedded OpenType, являющемся расширением стандартного формата TrueType.

Рассмотрим набор свойств, определяющих различные параметры отображения текста (табл. 14.5).

Таблица 14.5. *Свойства текста в CSS*

Свойство	**Описание**
`text-indent`	Определяет отступ первой строки при отображении блочных элементов. Отступ может быть отрицательным и задается либо абсолютным значением, либо в процентах от ширины вмещающего блока
`text-align`	Задает выравнивание текста при отображении блочных элементов. Может принимать значения `left`, `right`, `center`, `justify`, действие которых соответствует табл. 5.1
`text-decoration`	Задает украшение текста при отображении элементов. Может принимать значения `none` (обычный текст), `underline` (подчеркнутый), `overline` (надчеркнутый), `line-through` (перечеркнутый), `blink` (мерцающий). Значение `blink` не поддерживается Internet Explorer
`text-transform`	Определяет преобразование текста при отображении элементов. Может принимать значения `none` (без преобразования), `capitalize` (делать первую букву каждого слова прописной), `uppercase` (выводить прописными буквами), `lowercase` (выводить строчными буквами)
`letter-spacing`	Задает интервал между буквами при отображении текста. Значение `normal` означает стандартный интервал, абсолютный или относительный размер устанавливает интервал *в дополнение* к стандартному. Это значение может быть отрицательным
`word-spacing`	Задает интервал между словами при отображении текста. Значение `normal` означает стандартный интервал, абсолютный или относительный размер устанавливает интервал *в дополнение* к стандартному. Это значение может быть отрицательным

Таблица 14.5 *(окончание)*

Свойство	Описание
`white-space`	Определят правила обработки пробелов при отображении блочного элемента. Может принимать значения `normal` (пробелы и разрывы строк отображаются обычным образом), `pre` (отображение как в форматированном тексте), `nowrap` (пробелы отображаются обычным образом, но разрывы строк запрещены)
`line-height`	Задает высоту строки текста. Для блочного элемента устанавливает минимальную высоту входящих в него строчных блоков, для текстового элемента — точную высоту его строчного блока. Высота не может быть отрицательной и задается значением normal (по умолчанию) или в виде размера
`vertical-align`	Устанавливает положение элемента по вертикали относительно содержащего его строчного блока. Применимо к текстовым элементам и ячейкам таблиц. Принимает одно из значений: `baseline` (выравнивание по базовой линии строчного блока), `sub` (расположение в позиции нижних индексов), `super` (расположение в позиции верхних индексов), `top` (выравнивание по верху строчного блока), `text-top` (выравнивание по верхней линии шрифта строчного блока), `middle` (выравнивание по центру строчного блока), `bottom` (выравнивание по низу строчного блока), `text-bottom` (выравнивание по нижней линии шрифта строчного блока). Не меняет размер шрифта. Указание вместо одного из этих значений абсолютного или относительного размера означает увеличение (для положительного значения) или уменьшение (для отрицательного) отступа по вертикали на заданную величину. Величина в процентах вычисляется от значения `line-height`

Распространенные браузеры пока не поддерживают свойство `text-shadow`, создающее тень текста.

Свойства `text-decoration` и `vertical-align` не наследуются дочерними элементами, остальные свойства наследуемы. Приведем несколько примеров.

Свойство `text-indent` идеально подходит для создания "красной строки", принятой в русскоязычных текстах:

```
P { text-indent: 2em }
```

Во многих браузерах принято горизонтальное выравнивание текста по левому краю. Для приведения текста к более "читабельному" виду определим выравнивание по ширине в качестве основного способа для всего документа:

```
body { text-align: justify }
```

Свойства CSS, предназначенные для работы с цветом и фоном элементов, приведены в табл. 14.6.

Таблица 14.6. *Свойства цветов и фона*

Свойство	Описание
`color`	Задает цвет текста, содержащегося в элементе при отображении. Принимает значения наименования или кода цвета (см. *разд. 9.1*)
`background-color`	Устанавливает цвет фона элемента при отображении. Принимает значения наименования или кода цвета. Значение `transparent`, принятое по умолчанию, соответствует прозрачному фону
`background-image`	Определяет графический образ фона элемента при отображении, заданный относительным или абсолютным адресом в виде `url(URL-адрес)`. Значение `none`, принятое по умолчанию, означает отсутствие фонового образа
`background-repeat`	Если определено свойство `background-image`, то задает повтор фонового образа при отображении. Возможные значения: `repeat` (образ повторяется по горизонтали и вертикали), `repeat-x` (образ повторяется только по горизонтали), `repeat-y` (образ повторяется только по вертикали), `no-repeat` (образ не повторяется, отображается одна его копия)
`background-attachment`	Если задано свойство `background-image`, то устанавливает прокрутку фонового образа при отображении. Возможные значения: `scroll` (образ прокручивается вместе с документом), `fixed` (образ зафиксирован и не прокручивается)
`background-position`	Если задано свойство `background-image`, то определяет позицию фонового образа при отображении. Задается в виде `RX RY`, где `RX`, `RY` — положительные или отрицательные размеры, соответствующие горизонтальному и вертикальному смещению образа относительно области заполнения. Вместо значения `RX` может быть указано одно из значений `left`, `center`, `right`, вместо `RY` — одно из значений `top`, `center`, `bottom`
`background`	Является сокращением для остальных свойств фона, позволяя задавать их в одной декларации, например, `table{ background: url("img/bk1.gif") gray 50% 50% repeat fixed }`

Свойство `color` применимо ко всем элементам и наследуется. Свойства фона также применимы ко всем блочным и текстовым элементам, но не наследуются элементами-потомками.

На рис. 14.3 и в листинге 14.4 приведен пример использования изученных в этом разделе свойств CSS. Открыть этот пример в своем браузере вы можете, запустив документ Glava_14\FontProperties.html с компакт-диска.

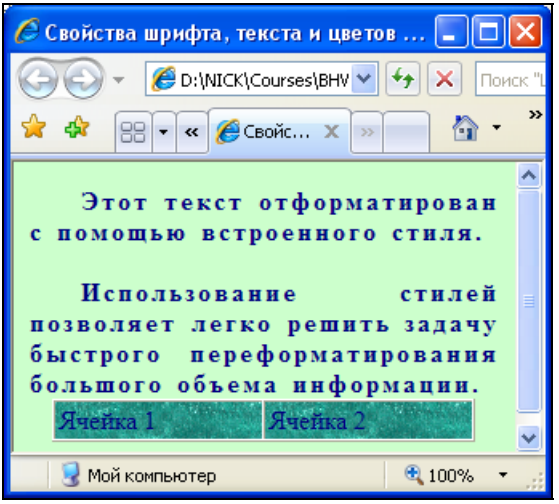

Рис. 14.3. Свойства шрифта, текста и цветов

Листинг 14.4. Свойства шрифта, текста и цветов

```
<html><head>
 <title>Свойства шрифта, текста и цветов</title>
<style type="text/css">
 body { /* основные свойства документа */
  text-indent: 2em;
  text-align: justify;
  letter-spacing: 0.1em;
  word-spacing: 0.1em;
  color: #000080;
  font-weight: 800;
  background-color: #CCFFCC
 }
```

```
 table{ /* свойства фона для таблиц */
  background: url("img/bk1.gif") #CCFFCC left top repeat fixed }
</style></head>
<body>
 <p>Этот текст отформатирован с помощью встроенного стиля.
 <p>Использование стилей позволяет легко решить задачу
 быстрого переформатирования большого объема информации.
<table width=90% align=center border=1 cellpadding=2 cellspacing=0>
<tr><td> Ячейка 1</td><td>
 Ячейка 2 </td></tr></table>
</body></html>
```

14.5. Свойства заполнителей, границ и рамок

Для любого элемента HTML в CSS создается *объемлющий его прямоугольник*, "устройство" которого схематично представлено на рис. 14.4, откуда видно, что помимо содержимого каждый элемент содержит *заполнитель* или *вмещающий блок*, *рамку* и *прозрачную границу*, образующие его внешние слои.

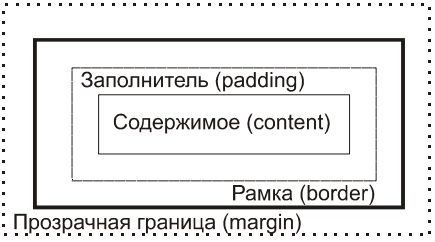

Рис. 14.4. Общая схема элемента HTML в CSS

Каждый из этих четырех вложенных прямоугольников распадается еще на четыре части: левую (`left`), правую (`right`), верхнюю (`top`) и нижнюю (`bottom`). Периметр прямоугольника называется *краем*, так что любой элемент может иметь до четырех краев:

- край содержимого — ограничивает отображаемую область элемента, в которой выводится его содержимое;
- край заполнителя — ограничивает свободное пространство между рамкой и содержимым. Если элемент имеет непрозрачный фон, внешней границей

его нанесения будет внешний край заполнителя. Если ширина заполнителя равна нулю, его край совпадает с краем содержимого;

- край рамки — ограничивает видимую рамку элемента. Если ее ширина равна нулю, то ее край совпадает с краем заполнителя;
- край границы — определяет прозрачные поля объемлющего прямоугольника, расположенные вне рамки. Если ширина границы равна нулю, то ее край совпадает с краем рамки.

Каждый край состоит из четырех сторон-частей.

Свойства границы и заполнителя приведены в табл. 14.7. Все они могут быть заданы в любых размерных единицах или в процентах. Указание в процентах всегда понимается относительно ширины вмещающего блока. Перечисленные свойства не наследуются и применимы ко всем элементам. Размеры границ *могут* быть отрицательными, а размеры заполнителей — нет.

Таблица 14.7. *Свойства границы и заполнителя*

Свойства	Описание

`margin-top`
`margin-right`
`margin-bottom`
`margin-left`
	Размеры верхней, правой, нижней и левой границ объемлющего прямоугольника соответственно

`margin`	Размер всех границ объемлющего прямоугольника одновременно, является сокращением для свойств `margin-top`, `margin-right`, `margin-bottom` и `margin-left`

`padding-top`
`padding-right`
`padding-bottom`
`padding-left`
	Размеры верхнего, правого, нижнего и левого заполнителя объемлющего прямоугольника соответственно

`padding`	Размер всех заполнителей объемлющего прямоугольника одновременно, является сокращением для свойств `padding-top`, `padding-right`, `padding-bottom` и `padding-left`

Приведем примеры работы с этими свойствами.

Для того чтобы браузер при размещении абзацев текста оставлял свободными по 5% ширины окна слева и справа, достаточно написать в стилевом файле определение вида

```
P { margin-left: 5%; margin-right: 5% }
```

По умолчанию в браузерах интервалы между абзацами установлены равными двух-трехкратному интервалу между строками, что порой делает текст неудобочитаемым. Исправим вид текста с помощью стиля:

```
P { margin-top: 4px; margin-bottom: 4px }.
```

Использовать "сокращенные" свойства `margin` и `padding` можно следующим образом:

```
body { margin: 10 px }      /* все границы равны 10 px */
body { margin: 10 px 8 px } /* top, bottom = 10 px, right, left = 8 px */
body {padding: 0,5 em 0.75 em 1 em }
            /* top=0,5 em, right=0,75 em, bottom=1 em, left=0,75 em */
body { margin: 1 em 2 em 3 em 4 em }
            /* top=1 em, right=2 em, bottom=3 em, left=4 em */
```

Основные свойства рамок приведены в табл. 14.8. Они применимы ко всем элементам и никогда не наследуются. Значения свойств рамок нельзя указывать в процентах.

Таблица 14.8. *Свойства рамок*

Свойства	**Описание**

`border-top-width`
`border-right-width`
`border-bottom-width`
`border-left-width`
	Задают размеры верхней, правой, нижней и левой рамок объемлющего прямоугольника соответственно. Указываются в виде фиксированного положительного размера или одним из ключевых слов `thin` (тонкая), `medium` (средняя, принято по умолчанию) или `thick` (толстая)

`border-width`	Сокращение для свойств `border-top-width`, `border-bottom-width`, `border-left-width` и `border-right-width`, позволяет задавать размер всех рамок одновременно

`border-top-color`
`border-right-color`
`border-bottom-color`
`border-left-color`
	Определяют цвета рамки объемлющего прямоугольника, цвет можно указать наименованием или 16-ричным кодом (см. *разд. 9.1*). Значение по умолчанию — цвет генерирующего элемента

`border-color`	Сокращение для свойств `border-top-color`, `border-right-color`, `border-bottom-color` и `border-left-color`, задает цвет всех рамок объемлющего прямоугольника одновременно. Значение `transparent` означает, что рамка должна быть прозрачной, прозрачная рамка при этом может иметь ненулевую ширину

Таблица 14.8 *(окончание)*

Свойства	Описание

`border-top-style`
`border-right-style`
`border-bottom-style`
`border-left-style`
	Задают стиль верхней, правой, нижней и левой рамок объемлющего прямоугольника соответственно. Могут принимать одно из *значений стиля* `none` (нет рамки), `hidden` (рамка скрыта), `dotted` (пунктирная рамка), `dashed` (короткий штрих), `solid` (сплошная линия), `double` (двойная линия), `groove` (трехмерная выемка), `ridge` (трехмерный выступ), `inset` (трехмерная врезка), `outset` (трехмерная вырезка)

`border-style`	Сокращение для свойств `border-top-style`, `border-right-style`, `border-bottom-style` и `border-left-style`, задает стиль всех рамок одновременно. Значением свойства может быть от одного до четырех стилей, стили те же, что для исходных свойств. Один заданный стиль применяется ко всем рамкам, четыре — последовательно к верхней, правой, нижней и левой сторонам, если заданы два или три значения, недостающий стиль принимается равным стилю противоположной рамки

`border-top`
`border-right`
`border-bottom`
`border-left`
	Сокращенные свойства рамки, позволяющие определить размер, стиль и цвет соответствующей рамки одновременно. Если какое-то из свойств не задано, используется его начальное значение

`border`	Устанавливает размер, стиль и цвет всех рамок одновременно. Если какое-то из свойств не задано, используется его начальное значение

Приведем примеры описанных свойств.

```
P {
  border-width: thin thick medium;
  border-color: red green blue;
  border-style: dashed dotted;
 }
```

Здесь верхняя граница абзаца будет тонкой, правая и левая — толстыми, а нижняя — средней по толщине. Верхняя граница выведется красным цветом, правая и левая — зеленым, а нижняя — синим. При этом верхняя и нижняя границы будут выведены штриховыми линиями, а левая и правая — пунктирными.

Посмотреть этот пример в своем браузере и поэкспериментировать с типами линий вы можете, открыв документ Glava_14\Borders1.html с компакт-диска.

Необязательно придавать элементам столь экзотичный вид. Вполне симпатичный блок текста можно получить, определив, например, следующее стилевое оформление абзацев:

```
P {
  border: thin dotted #333333;
  background-color: #CCCCCC; color: black;
  margin: 0.5em; padding: 0.1em
 }
```

Здесь абзац обрамляется тонкой темной пунктирной рамкой, черный текст выводится на светло-сером фоне, дополнительно настроены размеры областей заполнителя и прозрачной границы. Этот пример записан на компакт-диске под именем Glava_14\Borders2.html.

14.6. Позиционирование элементов

CSS позволяет не только задать свойства объемлющих прямоугольников, но и управлять их *взаимным расположением*. Рассмотрим основные средства позиционирования объектов.

Свойство `display` задает *тип объемлющего прямоугольника* для элемента и непосредственно влияет на его отображение. Это свойство применимо ко всем элементам и имеет фиксированный набор значений (табл. 14.9).

Таблица 14.9. *Основные значения свойства display*

Значение	Описание
`display="block"`	Блочный элемент, отображаемый как отдельный абзац. При его отображении генерируется главный прямоугольник блока, в котором располагаются объемлющие прямоугольники потомков элемента
`display="inline"`	Текстовый элемент, отображаемый как строки внутри текущего абзаца
`display="run-in"`	Присоединяемый элемент. Если следующий за ним элемент блочный, то форматируется как его первый текстовый элемент, иначе как блочный
`display="compact"`	Компактный элемент. Если следующий за ним элемент блочный, то форматируется как однострочный текстовый элемент, по возможности отображаемый на левой или правой границе последующего блока. В противном случае отображается как обычный блочный элемент
`display="none"`	Элемент и все его потомки игнорируются при отображении

Существуют также значения свойства `display` для маркировки элементов таблиц и списков.

Например, задав в документе Glava_14\Borders2.html свойство абзаца `display: inline`, вы получите отображение абзацев в виде обычных текстовых элементов, без разрыва строки.

Свойство `position` задает *схему позиционирования* элемента. Его определенные в CSS значения описаны в табл. 14.10.

Таблица 14.10. *Схемы позиционирования в CSS*

Значение	Описание
`position="static"`	*Статический* элемент, не имеющий специального позиционирования. Значение по умолчанию
`position="relative"`	*Относительно позиционируемый* элемент, для которого объемлющий прямоугольник сначала генерируется в обычном порядке, а затем смещается на величины, заданные свойствами `left` и `top`
`position="absolute"`	*Абсолютно позиционируемый* элемент, положение которого вычисляется относительно позиции ближайшего позиционированного предка на основании свойств `left` и `top`. Кроме позиции элемента, можно задать его размер свойствами `right` и `bottom`. Если все предки не позиционированы, положение вычисляется относительно элемента `body`
`position="fixed"`	*Фиксированный элемент*, который позиционирован абсолютно и дополнительно привязан к некоторой базовой точке. Его позиция зафиксирована относительно окна браузера, т. е. он остается неподвижным при прокрутке окна

Ко всем позиционированным элементам применимы свойства `left`, `top`, `right` и `bottom`, которые задают *относительное смещение* элемента. Смещение может быть задано как фиксированный размер либо указываться в процентах, что приводит к его вычислению относительно высоты или ширины вмещающего блока.

Позиционирование элементов может привести к тому, что на экране они накладываются друг на друга. При этом считается, что элементы могут располагаться в несколько *слоев* относительно оси Z, направленной от монитора на пользователя. Слои образуют *стек*, расположенный по оси Z. Управлять этим стеком легко: чем больше *индекс элемента* в стеке, тем выше он расположен, т. е. лучше видим пользователем. Элементы с одинаковым индексом

располагаются по правилу наполнения стека в программировании: чем раньше элемент расположен в тексте документа, тем глубже он находится.

Свойство `z-index` позволяет явно задавать индекс элемента в стеке, т. е. управлять наложением элементов друг на друга. Оно может принимать значения целого положительного числа или `auto` (индекс элемента автоматически принимается равным индексу его родителя).

Пример позиционирования объектов демонстрирует рис. 14.5. Код соответствующего документа содержится в листинге 14.5. На компакт-диске этот пример записан под именем Glava_14\Position1.html.

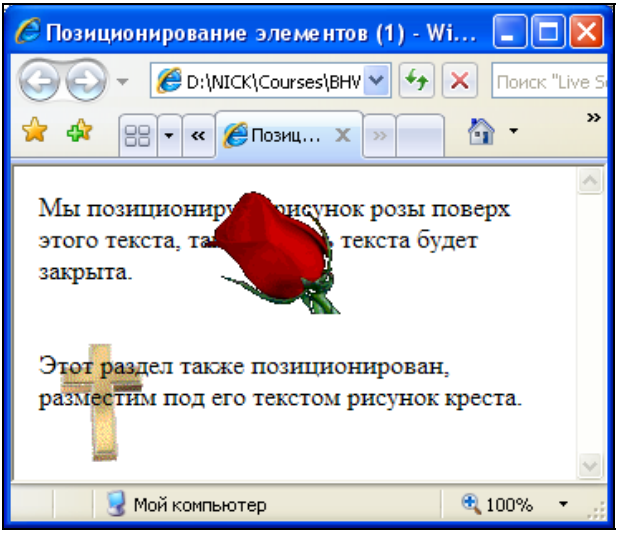

Рис. 14.5. Позиционирование элементов

Листинг 14.5. Позиционирование элементов

```
<html>
<head>
 <title>Позиционирование элементов (1)</title>
<style type="text/css">
 div.before {
  position: relative; left:5px; top: 0;
 }
 .abs1 {
```

```
  position: absolute; left: 100px; top: 0
 }
 .abs2 {
  position: absolute; z-index: -1; left: 10px; top: -5px
 }
</style>
</head><body>
<DIV class="before">Мы позиционируем рисунок розы поверх этого текста,
так что часть текста будет закрыта.
<img src="img/rose.gif" width="82" height="74" class="abs1">
</DIV>
<br><br>
<DIV class="before">Этот раздел также позиционирован, разместим под его
текстом рисунок креста.
<img src="img/cross.gif" width="55" height="75" class="abs2">
</div></body></html>
```

Свойство `float` со значением `left` или `right` указывает, что элемент *плавающий*, и задает его выравнивание влево или вправо. По умолчанию свойство имеет значение `none` (элемент не является плавающим). Оно применимо ко всем элементам, кроме позиционированных.

Плавающий элемент всегда считается блочным, т. е. его свойство `display` игнорируется. Такой элемент смещается влево или вправо до тех пор, пока не встретит рамку, заполнитель или границу другого блочного элемента. Элементы, следующие за плавающим элементом, сдвигаются относительно него и обтекают его справа или слева. Например, использование "плавающих" абзацев — это еще один способ вывести их рядом:

```
<p style="float: left; width: 50%">Абзац 1 - прижат к левому краю.</p>
<p style="float: right; width:50%">Абзац 2 - прижат к правому краю.</p>
```

Пример с плавающими абзацами записан на компакт-диске под именем Glava_14\Float1.html.

Свойство `clear` указывает, как текущий элемент *не может* обтекать предыдущий плавающий объект. Принятое по умолчанию значение `none` разрешает обтекание плавающего элемента с обеих сторон, `left` *запрещает* обтекание слева, `right` — справа, а `both` — с обеих сторон. Запрет обтекания означает, что данный элемент должен отображаться ниже предыдущего плавающего элемента. Корректная поддержка свойства пока реализована не во всех обо-

зревателях. Проверить, как обстоит дело с вашим браузером, вы можете, открыв пример Glava_14\Float2.html с компакт-диска.

В завершение раздела кратко опишем свойства, позволяющие управлять размерами и видимостью элементов на экране.

- `width` — задает ширину содержимого для блочных элементов и элементов, *внешних* по отношению к CSS (например, рисунков). Ширина не может быть отрицательной и задается в виде размера или в процентах от ширины вмещающего блока.
- `height` — аналогичным образом задает высоту содержимого. Напомним, что высота текстовых элементов устанавливается свойством `line-height`.

Свойства ширины и высоты применимы ко всем элементам, кроме текстовых, строк таблиц и групп строк. Эти свойства не наследуются.

- `clip` — задает область *обрезки* содержимого при переполнении. Оно может принимать значение `auto` (область обрезки совпадает с границами элемента) или `rect (<top> <right> <bottom> <left>)`, где `<top>`, `<bottom>`, `<right>` и `<left>` показывают величину отступов от соответствующих сторон блока.

Отрицательные смещения допускаются, например `P { clip: rect(-2px, -2px, 4px, 4px); }`.

- `overflow` — определяет *правила обрезки* содержимого элемента при переполнении. Принимает одно из значений `visible` (содержимое не обрезается), `hidden` (содержимое обрезается, размер и форма области обрезки задается свойством `clip`), `scroll` (содержимое обрезается, браузер должен обеспечить механизм прокрутки), `auto` (правила обрезки определяются браузером и при необходимости обеспечивается прокрутка).

Последние два свойства применимы к блочным, но не текстовым, элементам, не используют размеров в процентах и не наследуются.

- `visibility` — устанавливает *видимость* элемента при отображении. Принимает одно из значений `visible` (элемент видим), `hidden` (невидим).

Применяя и комбинируя изученные в этой главе свойства CSS, можно добиваться самых разнообразных эффектов. Например, несложный код из документа Glava_14\Shadow.html демонстрирует создание заголовка с тенью, полученного наложением друг на друга двух элементов `<h1>`, относящихся к классам с различными свойствами.

Еще одно популярное применение стилей — улучшение внешнего вида форм, не слишком удачно выглядящих в большинстве браузеров.

14.7. DHTML и создание визуальных эффектов

DHTML (Dynamic HTML) — не отдельный язык или формат. По сути дела, это термин для обозначения Web-страниц с динамически изменяющимся содержимым. DHTML построен на трех основных технологиях:

- HTML — средство разметки содержимого;
- CSS — средство стилевого оформления документа;
- язык программирования сценариев JavaScript — средство динамического управления содержимым и обработки событий.

Объектная модель документа (Document Object Model, DOM) связывает эти три технологии воедино, представляя собой *интерфейс прикладного программирования* Web-документов.

Суть модели DOM в том, что она делает все элементы страницы *программируемыми объектами*. С помощью языка сценариев разработчик может получить доступ к любому элементу страницы и поменять любые из его разрешенных к изменению свойств. Таким образом, значение любого атрибута любого тега можно изменять программно, что делает документ действительно динамическим. Кроме того, любое действие пользователя, связанное с навигацией по документу и выполняемое с помощью клавиатуры или мыши, трактуется как *событие*, которое может быть перехвачено и обработано сценарием страницы.

Один из основных принципов DOM — представление документа в виде логической древовидной структуры, о которой мы говорили в *разд. 14.1*, другой важный принцип — представление любых элементов документа в виде объектов, а не просто наборов данных. Соответственно, эти объекты могут обладать изменяемыми *свойствами* и программируемыми *методами* обработки связанных с ними событий.

К сожалению, объем и задачи книги не позволяют нам детально познакомиться с интереснейшими технологиями DHTML, но о самом главном следует сказать несколько слов.

Все объекты страницы образуют иерархию, на вершине которой находится объект `document`. Этот объект, как и все остальные, использует для доступа к нижележащим объектам два набора:

- `all` — содержит ссылки на все объекты, расположенные в иерархии ниже данного;
- `children` — содержит ссылки на все объекты, непосредственно порождаемые данным.

Каждый набор имеет ряд доступных *свойств*, например, свойство `length` определяет общее число элементов в наборе.

В записи имени каждого конкретного элемента имена объектов, наборов и свойств разделяются символом точки.

Если некоторый абзац текста является *вторым* на странице и имеет атрибут `id="t2"`, обратиться к нему можно следующими способами:

```
document.all.t2
document.all['t2']
document.all[1]
```

В последнем случае мы учли, что номер абзаца в тексте нам известен точно (*второй*), а вся нумерация в языке сценариев начинается *с нуля*.

Задать имя элемента HTML можно с помощью не только атрибута `id`, но и `name` там, где он допустим.

В программируемом сценарии можно обращаться к *любым свойствам CSS*, следуя простому правилу: из названий свойств CSS удаляются дефисы, а части имен после дефисов присоединяются к предшествующим частям с прописной буквы. Так, свойство CSS `background-color` доступно из сценария JavaScript как `backgroundColor`, а `border-top-width` как `borderTopWidth`.

Кроме того, язык сценариев позволяет писать подпрограммы-функции, которые, становясь обработчиками событий от элементов форм или тега `<body>`, выполняют любые требуемые расчеты и представляют пользователю их результаты.

Несложный пример, записанный на компакт-диске под именем Glava_14\Position2.html, демонстрирует динамическое позиционирование рисунка, начинающего двигаться сразу же после загрузки страницы в окно браузера.

В версии DHTML, разработанной Microsoft для браузера Internet Explorer, есть элементы, использование которых не требует изучения программирования, но позволяет легко добиться впечатляющих эффектов. Речь идет о *фильтрах*, представляющих собой готовые алгоритмы преобразования содержимого в окне браузера. Фильтры применимы к блочным элементам HTML, не являющимся отдельными окнами, например, `<body>`, `<img>`, `<button>`, `<input>`, `<table>` и `<textarea>`. Обращение к фильтру во всем подобно записи декларации в CSS и имеет общий вид

```
filter: название (параметры).
```

Отдельные параметры, разделенные запятыми, определены в виде

```
название=значение.
```

Если фильтр не имеет параметров, после названия все равно указываются пустые круглые скобки. Допускается перечислять сразу несколько фильтров, в этом случае они задаются в виде списка с пробелом в качестве разделителя. В табл. 14.11 приведены названия и краткие описания основных фильтров.

Таблица 14.11. *Основные фильтры для создания динамических эффектов*

Наименование	**Описание**
alpha	Управляет уровнем непрозрачности объекта
blendTrans	Изменяет контрастность изображения объекта
blur	Создает эффект размытия изображения
chroma	Делает прозрачными пикселы заданного цвета
dropShadow, shadow	Создают эффект наложения тени
flipH	Отражает объект зеркально по горизонтали
flipV	Отражает объект зеркально по вертикали
glow	Добавляет свечение внешних границ объекта
gray	Отображает объект в оттенках серого цвета
invert	Меняет оттенок, яркость и насыщенность цветов объекта на противоположные
light	Подсвечивает объект заданным цветом
mask	Накладывает на объект маску из пискелов заданного цвета
revealTrans	Показывает или скрывает объект, используя набор эффектов перехода
wave	Создает волнообразный эффект, наложенный на объект
xray	Имитирует рентгеновский снимок объекта

Пример использования фильтра приведен в файле Glava_14\Filter.html на компакт-диске.

Из популярных браузеров лишь последние версии Internet Explorer обеспечивают поддержку фильтров, поэтому применять их следует с осторожностью.

Глава 15

Мультимедиа и внешнее содержимое

Мультимедийные функции языка HTML позволяют Web-мастерам включать в свои страницы не только изображения, но и звуковые файлы, видеоклипы, внешние программы и другие Web-страницы.

В этой главе мы изучим имеющиеся в HTML средства для добавления в документ разнообразных видов файлов.

15.1. Тег <*object*>

Указанный в заголовке раздела тег — основа подключения любого внешнего содержимого. Вместо набора разрозненных и зачастую нестандартных тегов внедрения звуков, апплетов, внешних документов и рисунков, тег `<object>` предлагает включение в документ *любых* объектов, типы которых известны браузеру. В частности, `<object>` способен заменить теги `<img>`, `<applet>`, `<iframe>` и `<embed>`. По сути дела, элемент `<object>` позволяет разработчикам указывать браузеру тип и URL-адрес включаемого объекта для того, чтобы браузер мог определить, будет ли он отображать объект непосредственно или передаст исполнение внешней программе.

Содержимым тега могут быть:

- теги `<param>`, задающие начальные значения параметров объекта;
- следующие за ними текстовые или блочные элементы, задающие альтернативное содержание, отображаемое, если браузер не поддерживает данный тип объектов.

Атрибуты тега приведены в табл. 15.1.

Таблица 15.1. *Атрибуты тега <object>*

Атрибут	Описание
declare	Делает текущий тег <object> объявлением. В этом случае объект должен быть внедрен последующим определением <object>, ссылающимся на атрибут id этого объявления
classid="URL"	Указывает URL объекта (например, исполняемого файла апплета) или его уникальный 64-разрядный шестнадцатеричный номер GUID (для элементов ActiveX). Может использоваться вместе с атрибутом data или как альтернатива ему, в зависимости от типа объекта
data="URL"	Задает URL файла данных, необходимых для включения объекта
archive="URL"	Определяет разделенный пробелами список адресов URL, по которым находятся относящиеся к объекту ресурсы, задаваемые атрибутами classid и data. Предварительная загрузка архивов приведет к уменьшению времени загрузки объекта
codebase="URL"	Задает базовый URL для адресов, установленных атрибутами classid, data и archive. Если эти адреса указаны относительными URL, то их полные URL составляются с помощью значения codebase. Также служит для задания местоположения динамических библиотек, используемых управляющими элементами ActiveX
type=" MIME-тип файла"	Определяет MIME-тип файла для данных, задаваемых атрибутом data. Рекомендуется при наличии атрибута data, т. к. позволяет браузеру избежать загрузки содержимого, которое он не поддерживает. MIME-типы файлов приведены в табл. 15.2
codetype=" MIME-тип файла"	Указывает MIME-тип объекта, заданного атрибутом classid, рекомендуется при его применении. По умолчанию принимается равным значению type
standby="строка"	Задает текст, который может отображаться браузером во время загрузки объекта

Помимо перечисленных в таблице, допустимо указание общих атрибутов width="размер" и height="размер" для определения ширины и высоты окна объекта в пикселах или процентах, а также атрибутов align, border, hspace

и `vspace` для управления внешним представлением окна объекта. Действие этих атрибутов такое же, как для изображений (см. *разд. 10.3*). Атрибут `usemap="URL"` может использоваться для указания на клиентскую карту ссылок, определенную тегом `<map>` (см. *главу 11*), а атрибут `name="строка"` — для задания имени объекта, находящегося внутри формы (см. *главу 13*).

Необходимый набор атрибутов определяется типом внедряемого объекта. Так, для включения с помощью тега рисунка типа JPEG, хранящегося во вложенной папке images, достаточно написать фрагмент кода

```
<object type="image/jpeg" data="images/photo1.jpg" width="230"
height="240" >Мое фото</object>.
```

Проверить этот пример в своем браузере вы можете, открыв документ Glava_15\JpegObject.html с компакт-диска.

В качестве ширины и высоты окна объекта здесь указаны реальные размеры рисунка. В зависимости от браузера, это может привести к выводу изображения целиком или к появлению встроенного окна с полосами прокрутки, если браузер по умолчанию добавляет ненулевые отступы от объекта. Вставленный рисунок при этом не масштабируется.

Приведем также пример вызова компонента из динамической библиотеки ActiveX, созданной компанией Microsoft в качестве стандарта разработки платформенно-независимых элементов интерфейса и обработки данных. Все элементы ActiveX при установке в Windows регистрируются в ее *реестре* — едином для всех программ хранилище данных. Уникальным идентификатором элемента при этом служит его 64-разрядный шестнадцатеричный номер, называемый *GUID*. Для доступа к элементу из документа HTML достаточно, кроме требуемых общих атрибутов, указать его GUID в атрибуте `classid`, например:

```
<object classid="clsid:978C9E23-D4B0-11CE-BF2D-00AA003F40D0"
id="label1" width="500" height="100" align="middle">
К сожалению, ваш браузер не поддерживает ActiveX</object>
```

Здесь вызывается несложный компонент ActiveX, создающий настраиваемую текстовую метку. Посмотреть полный код примера, в том числе значения параметров, переданных компоненту ActiveX, вы можете, открыв пример Glava_15\ActivexObject.html с компакт-диска. Аналогичным образом вызываются функции надстроек многих альтернативных браузеров, не поддерживающих ActiveX.

Прежде чем говорить об использовании тега `<object>` для решения других задач, кратко обсудим то, что нужно знать о *MIME-типах* файлов.

Технология MIME (Multipurpose Internet Mail Extension) изначально предназначалась для присоединения к электронным письмам файлов различных типов.

Позднее типы MIME стали применяться для унификации действий программ по обработке файлов. Браузер по указанному MIME-типу принятого файла определяет дальнейшие действия с ним: например, изображение в формате JPEG будет показано им непосредственно, а для отображения видеофильма в формате MOV может понадобиться вызов внешнего плагина, предназначенного для этой цели. MIME-тип обозначается строкой, состоящей из двух частей, разделенных символом `/`. До разделителя указывается *базовый тип*, а после — *подтип* файла, причем базовые типы определены стандартом, а подтипы создаются и регистрируются по мере появления новых форматов файлов. В табл. 15.2 приведены базовые MIME-типы и примеры типов, соответствующих хорошо нам знакомым видам файлов, встречающихся в Интернете.

Таблица 15.2. *Основные MIME-типы файлов*

Базовый MIME-тип	Описание и примеры
`application`	Двоичные данные. Основной подтип — octet-stream (неструктурированные потоки байтов). Остальные подтипы связаны с определенной программой или форматом, например, application/zip — ZIP-архивы
`audio`	Аудиофайлы. Основной подтип — `basic` (одноканальные аудиоданные с максимальной частотой 8000 Гц), популярным форматам звука соответствуют типы `audio/wav`, `audio/vnd.digital-winds` (MIDI, поддерживается также написание `audio/midi`)
`image`	Графические образы. Основные подтипы — `image/jpeg`, `image/gif`, `image/png`
`message`	Сообщения электронной почты, имеющие строго определенную структуру, например, `message/http` — данные протокола HTTP, `message/news` — почтовый формат новостей
`model`	Модели, состоящие из объектов, например, `model/vnd.dwf` — CAD-модели формата DWF (AutoCAD), `model/vrml` — документы VRML
`multipart`	Составные файлы, состоящие из наборов данных разных типов, например, `multipart/form-data` — результаты заполнения HTML-формы пользователем
`text`	Текстовые файлы. Основной подтип — `text/plain` (неформатированный текст), остальные подтипы рассматриваются как текст, содержащий дополнительное форматирование, например, `text/html` — документ HTML, `text/css` — каскадные таблицы стилей, `text/rtf` — файлы формата RTF
`video`	Видеофайлы. Основной подтип `video/mpeg`, также `video/msvideo` (тип AVI), `video/quicktime` (формат QuickTime)

Теги `<param>`, применяемые только в начале тега `<object>`, позволяют передать объекту начальные данные (*параметры*) в том случае, когда он является программным приложением, компонентом ActiveX или иной надстройкой, позволяющей указывать режимы своей работы из вызывающей программы. Тег `<param>` не имеет содержимого и не закрывается, в подавляющем большинстве случаев он записывается как

```
<param name="параметр" value="значение">
```

и передает объекту строку данных вида "`параметр=значение`". Конкретные наименования параметров и набор их допустимых значений зависят от используемой программы или компонента. Пример применения тега для передачи данных апплету приведен в *разд. 15.6*.

15.2. Подключение звуковых файлов

Можно выделить два типа звуковых файлов: *музыкальные* (инструментальные) и *цифровые*. В первом случае мелодия хранится в виде совокупности информации о нотах и музыкальных инструментах. Это позволяет компактно описывать музыкальные данные, но ограничивает возможности воспроизведения мелодиями "без голоса". Во втором случае звук *оцифрован* с помощью аналого-цифрового преобразователя (АЦП). При записи файла непрерывные звуковые колебания, передаваемые по проводам как изменение напряжения, дискретизируются с некоторой частотой, преобразуются в цифровой формат и сохраняются в файле. При воспроизведении этого файла звуковая карта выполняет обратный процесс, преобразуя цифровую информацию в аналоговую.

На качество звукопередачи и размер звукового файла влияют следующие факторы:

- качество записи (bit rate) — число битов информации, передаваемой в секунду; параметр варьируется в широких пределах, высококачественным считается звук с качеством записи от 128 Кбит/с;
- частота дискретизации (sampling rate) — количество преобразований аналогового звука в цифровой, выполняемое в секунду. Распространенные значения этого параметра — от 8–12 до 44,1 кГц/с и выше;
- размер аудио-образца (sample rate) — в целях экономии памяти аудиозапись строится из *образцов* (samples), представляющих собой сохраненные компьютером последовательности цифровых отсчетов. Как правило, параметр имеет значение 8 или 16 бит;
- число аудиоканалов, в которых воспроизводится звук. Значение 1 соответствует моно, а 2 — стереозаписи.

Существует множество звуковых форматов, на Web-страницах чаще всего встречаются следующие типы файлов:

- MIDI (Musical Instrument Digital Sequence) — стандартный музыкальный формат для воспроизведения мелодий;
- WAV — цифровой формат звуковых файлов Windows;
- MP3 — цифровой формат хранения *сжатого* аудиосигнала. Сжатие производится за счет необратимого удаления тонких деталей звучания, неразличимых человеческим ухом;
- RA (Real Audio) — сжатый цифровой формат для передачи звука в реальном времени.

Распространены также звуковые форматы WMA (сжатый цифровой звук от Microsoft), AU ("базовый" формат Sun/NeXT, бывший фактическим стандартом на первых стадиях развития Интернета), AIFF (формат компьютеров Macintosh) и ряд других.

Можно выделить два основных способа связывания звукового файла с документом HTML:

- создание гиперссылки на звуковой файл;
- применение специализированного тега `<emded>` или `<object>`.

Оба способа имеют свои преимущества и недостатки. Первый хорош тем, что предельно прост и не требует изучения новых тегов:

```
<a href="sound/scream.wav">scream.wav</a> (2 сек., 25 Кб)<br>
<a href="sound/baby.mid">baby.mid</a> (1 мин. 3 сек., 5 Кб).
```

Здесь якорями ссылок сделаны имена музыкальных файлов, а после выбора такой ссылки, в зависимости от браузера, может произойти следующее:

- появится стандартное окно запроса на открытие или сохранение файла (Internet Explorer);
- звук будет проигран в фоновом режиме (Opera);
- в зависимости от атрибута `target` ссылки, в новом или текущем окне браузера появится панель инструментов аудиоплеера (Mozilla Firefox, Netscape).

Проверить, как поведет себя ваш браузер, вы можете, открыв файл Glava_15\SoundLinks.html с компакт-диска.

Обратите внимание на разницу во времени звучания и объеме файлов. Инструментальный формат MIDI здесь в явном выигрыше. Однако записать звук из файла scream.wav музыкальными инструментами MIDI было бы затруднительно.

При подключении ссылок на звуковые файлы (как и на любое другое содержимое большого объема) желательно информировать пользователя о размере загружаемого файла, как и сделано в рассмотренном примере.

В источниках по HTML приходится также встречать тег вида

`<bgsound src="URL" loop="число">`,

предназначенный для задания фонового звука страницы. Здесь атрибут `src` определяет URL-адрес звукового файла, а `loop` показывает число повторений мелодии или устанавливает ее бесконечное повторение специальным указанием `loop="infinite"`.

Следует учесть, что этот тег нестандартен и многие браузеры его не поддерживают, хотя Internet Explorer 7 и Opera 9.02 тег восприняли. Если в документе имеется несколько тегов `<bgsound>`, поведение браузеров также может отличаться. Например, Opera выбирает в качестве фонового первый из встретившихся фоновых звуков независимо от его формата, а Internet Explorer может выполнить динамическое наложение MIDI- и WAV-звука. Пример использования тега дан в документе Glava_15\BgSound.html с компакт-диска.

Теперь рассмотрим специализированные теги.

Нестандартный тег `<embed>` разрабатывался специально для подключения звуковых файлов. Позднее его возможности были расширены на файлы различных типов. Тег блочный, при этом он не имеет закрывающей части и содержимого. Основные атрибуты тега `<embed>` применительно к звуковым файлам приведены в табл. 15.3.

Таблица 15.3. *Основные атрибуты тега <embed>*

Атрибут	**Описание**
`src="URL"`	URL-адрес загружаемого файла
`width="размер"`	Ширина окна объекта
`height="размер"`	Высота окна объекта
`alt="строка"`	Альтернативный текст для браузеров, не поддерживающих тег
`align`	Относительное выравнивание объекта в его окне; допустимы значения `top`, `middle`, `bottom`, `left`, `right`, интерпретируемые в соответствии с табл. 10.2
`hspace="пикселы"`	Горизонтальный отступ от окна объекта до содержимого
`vspace="пикселы"`	Вертикальный отступ от окна объекта до содержимого
`pluginspage="URL"`	URL-адрес загружаемого плагина

Таблица 15.3 *(окончание)*

Атрибут	Описание
`autostart`	Воспроизведение начинается автоматически (значение `true`) или нет (значение `false`, принято по умолчанию)
`autoload`	Загрузка файла производится автоматически (значение `true`, по умолчанию) или нет (значение `false`)
`starttime="mm:ss"`	Воспроизведение начинается с указанного в минутах и секундах времени от начала файла, значение по умолчанию `00:00`
`endtime="mm:ss"`	Воспроизведение заканчивается с указанного в минутах и секундах времени от начала файла, значение по умолчанию — время окончания воспроизведения файла
`volume="проценты"`	Громкость воспроизведения в процентах от максимальной, значение по умолчанию — установленное в системе
`controls`	Определяет вид панели управления. Допустимы значения `console` (все органы управления, принято по умолчанию), `smallconsole` (компактная панель), `playbutton` (только кнопка воспроизведения) и ряд других

Поскольку тег нестандартен, нет гарантий ни одинаковой интерпретации его атрибутов различными браузерами, ни поддержки тега в будущем. Так, даже достаточно "безопасный" набор атрибутов

```
<embed src="sound/rooster.wav" width="144" height="60" autostart="true"
align="left" hspace="5" vspace="5">
```

был интерпретирован популярными браузерами по-разному. Тем не менее, в настоящее время тег поддерживается всеми браузерами, приведенный здесь пример его использования вы можете увидеть, открыв документ Glava_15\Embed.html с компакт-диска.

Internet Explorer при работе с тегом задействует возможности приложения Windows Media, а многие другие популярные браузеры — внешние модули-плагины, такие как LiveAudio от компании Netscape или QuickTime от Apple.

В рассмотренном примере ширина и высота выбраны исходя из стандартного размера панели органов управления модуля LiveAudio. Впрочем, современные версии плагинов позволяют обращаться с размерами своего окна достаточно гибко.

При подключении звукового файла тегом `<object>` следует указать соответствующий тип файла (см. табл. 15.2), а также обычные общие атрибуты (листинг 15.1).

Листинг 15.1. Подключение звуковых файлов тегом <object>

```
<p>Файл в формате WAV:
<object type="audio/wav" data="sound/rooster.wav" align="middle"
width="144" height="96" hspace="8">Тип объекта Wav не поддерживается</
object>
<p>Файл в формате MIDI:
<object type="audio/midi" data="sound/saintsgo.mid" align="middle"
width="144" height="96" hspace="8">Тип объекта Midi не поддерживается</
object>
```

К сожалению, аудиофайл формата MP3, подключенный данным способом с указанием MIME-типа `audio/mpeg`, отобразят не все современные браузеры. Некоторые из них могут запросить установку соответствующего плагина.

Посмотреть приведенный в листинге 15.1 пример вы можете, открыв с компакт-диска документ Glava_15\SoundObject.html.

15.3. Подключение видео

Существуют два основных типа видеоданных: *аналоговый* и *цифровой*. Аналоговое видео — самый ранний метод передачи видеосигнала. Его первая разновидность — *композитное* видео — комбинирует все видеокомпоненты (сигналы яркости, цвета и синхронизации) в один сигнал, что существенно ограничивает качество. В *компонентном* видео различные видеокомпоненты представлены независимыми сигналами. Этот тип видео используется, прежде всего, в телевидении.

Цифровое видео, как следует из названия, представляет собой серию изображений, информация о которых хранится в оцифрованном виде. Принципы *оцифровки* видео и звука (см. *разд. 15.2*) во многом совпадают.

Кратко охарактеризуем распространенные на сегодняшний день форматы видеофайлов.

- Audio Video Interleaved (тип файла AVI) — файловый формат, введенный компанией Microsoft для работы с видеоизображениями в среде Windows. В этом формате секторы видео- и звуковых данных чередуются таким образом, чтобы видеоплеер мог поддерживать минимальную буферизацию данных. В зависимости от используемого в данный момент *кодека* (устройства компрессии/декомпрессии видеоданных и звука), формат поддерживает как сжатие без потерь качества, так и с потерями. В области записи и редактирования видео AVI, фактически, является стандартом.

- Quick Time Movie (типы файлов QT, MOV) — формат, разработанный в рамках технологии Quick Time. Поддерживает несколько различных алгоритмов сжатия, в том числе свой собственный метод компрессии. Особенность формата — возможность записи информации на независимые "дорожки" (видео и аудио). На разных дорожках видеоданные могут иметь различную частоту и разрешение, аудиоданные — различный формат и т. д.
- Motion JPEG (типы файлов MPG, MPEG, MP4, 3GP, DAT и др.) — формат для записи и воспроизведения видео, разработанный группой экспертов по движущимся изображениям (Moving Picture Expert Group). Имеет собственный алгоритм компрессии.
- RealVideo (тип файла RM) — формат предназначен для низкоскоростной передачи видео в Интернете в реальном времени. Поддерживает небольшое разрешение и обладает невысоким качеством. Воспроизводится только на компьютере. Требует специального программного декодера.

Для подключения видеофайла к Web-странице можно воспользоваться возможностями тега `<embed>`:

```
<embed src="images/Clock.avi" width="400" height="400" align="middle"
alt="Видео не поддерживается" autoload="false" controls="smallconsole">.
```

Увидеть данный пример в своем браузере вы можете, открыв документ Glava_15\Video1.html с компакт-диска. Вероятнее всего, качество отображения этого классического ролика современным браузером будет невысоким.

Более гибкое поведение тега `<embed>` могут обеспечить его дополнительные нестандартные атрибуты, такие как `cache` (управление кэшированием), `volume` (уровень звука), `scale` (масштаб), `hidden` (отключение звука), `loop` (управление циклом показа ролика) и ряд других. Некоторые из этих атрибутов, например, `volume`, требуют задания числовых величин, другие имеют одно из двух логических значений "истина"(`true`) или "ложь" (`false`).
К сожалению, в различных браузерах одни и те же значения могут приводить к несовпадающим результатам.

Второй распространенный способ работы с видеоданными состоит в том, что видеофайл подключается тегом `<object>`, указывающим через свои параметры информацию об используемом компоненте ActiveX или внешнем плагине, а обращение к видеоклипу происходит через встроенный код на языке сценариев JavaScript (листинг 15.2).

Листинг 15.2. Подключение видеофайла тегом `<object>`

```
<object classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"
 codebase="http://www.apple.com/qtactivex/qtplugin.cab"
 width="400" height="300" id="QTPlayer">
```

```
 <param name="src" value="images/Sample.mov">
 <embed width="400" height="300" src="" type="video/quicktime"
  pluginspace="http://www.apple.com/quicktime/download"
  enablejavascript="true" autostart="false" name="QTPlayer">
</object>
<script type="text/javascript">
function playUrl(url) {
 var player = document.QTPlayer;
 player.SetURL(url);
 player.play();
}
playURL("images/Sample.mov");
</script>
```

Здесь атрибут `classid` тега `<object>` задает GUID требуемого компонента ActiveX, а атрибут `codebase` указывает на страницу Интернета, предназначенную для установки плагина QuickTime, отображающего видео. Сам ролик, сохраненный в формате MOV, подключается из кода на языке сценариев, описанного внутри тега `<script>`. При отсутствии в системе плагина QuickTime браузер предложит установить его. Посмотреть этот пример вы можете, открыв на прилагаемом компакт-диске документ Glava_15\Video2.html. Аналогичные технологии существуют на основе Java (см. *разд. 15.6*) и Macromedia Flash (см. *разд. 15.7*).

Разумеется, не запрещено и прямое указание ссылки на видеофайл, который пользователь может скачать к себе на компьютер:

```
<p><a href="images/Clock.avi">Clock.avi, 81 Кб</a>
<p><a href="images/Sample.mov">Sample.mov, 922 Кб</a>
<p><a href="images/demo.mpg">demo.mpg, 55 Кб</a>
```

Как и в случае со звуком, событие, произошедшее после выбора ссылки, зависит от браузера (см. *разд. 15.2*). Этот пример реализован в документе Glava_15\Video3.html на компакт-диске.

При работе с видеофайлами следует учесть их объем, значительный даже по сравнению с качественными аудиозаписями. В любом случае, следует предупредить пользователя о размере загружаемого или просматриваемого файла.

Своеобразное и очень популярное сегодня применение видео в Интернете — *Web-трансляции* — отображение сменяющих друг друга через некоторый интервал времени "живых" картинок, снятых специализированной *Web-камерой*. В простейшем случае технология Web-трансляции вовсе не пред-

полагает передачи по каналам связи видеофайлов. Установленная вместе с Web-камерой программа передает на Web-сервер изображения в формате JPEG с одним и тем же именем, а Web-страница или ее часть, содержащая изображение, обновляется с определенной частотой. Это обновление может быть выполнено с помощью метатега или сценария на JavaScript. Необходимое программное обеспечение и примеры кода, как правило, поставляются вместе с Web-камерой.

При подготовке Web-трансляции придется вспомнить о традиционно невысокой пропускной способности отечественных сетей. Большинство интернет-продюсеров рекомендует для трансляции разрешение картинки не более чем 176×144 пиксела.

15.4. Использование скриптов

Мы уже обсуждали теоретические аспекты использования клиентских программ-скриптов в *разд. 2.1*. Нужно лишь добавить, что фактическим стандартом языка клиентского программирования в современных браузерах остается JavaScript. Он, как и другие языки сценариев, является *интерпретируемым*, т. е. не требует преобразования в исполняемый модуль операционной системы, но предполагает наличие программы-*интерпретатора*, читающей и выполняющей исходный текст скрипта (ее роль исполняет браузер). Основные области применения JavaScript следующие:

- динамическое формирование и изменение содержимого Web-страниц "на лету";
- обработка событий, таких, как загрузка или выгрузка документа, движение курсора мыши, нажатие клавиши и т. п.;
- проверка правильности ввода данных в формы;
- выполнение действий, зависящих от выбора пользователя;
- выполнение действий, зависящих от генератора случайных чисел, даты (или времени);
- подключение внешних источников данных.

Можно выделить следующие ограничения языка сценариев:

- ограничение доступа к физическим ресурсам компьютера и файловой системе;
- невозможность сохранения информации на стороне клиента (кроме специальных *cookie-файлов*, объем которых не превышает 4 Кбайт).

Программа на JavaScript может быть вставлена в документ тегом, общий вид которого показан в листинге 15.3.

Листинг 15.3. Общий вид сценария на JavaScript

```
<script language="JavaScript">
<!--
 код на JavaScript
// -->
</script>
<noscript>
 Замещающий текст для браузеров, не поддерживающих JavaScript
</noscript>
```

Тег `<noscript>` необязателен, но рекомендуется. Он может включать любые блочные элементы и задает альтернативное содержимое для браузеров, не поддерживающих JavaScript. Комментарии `<!--` и `//-->` прячут код JavaScript от старых браузеров, которые не могли его отобразить. Количество тегов `<script>` не ограничено. Вкладывать один тег `<script>` в другой нельзя. Функции на JavaScript принято заключать в тег `<head>` документа.

Согласно спецификации, атрибуту `language` следует предпочитать стандартный атрибут `type`, позволяющий указать MIME-тип языка сценариев (табл. 15.4).

Таблица 15.4. *MIME-типы языков сценариев*

Язык сценариев	Значение атрибута `type`
JavaScript (JScript)	`type="text/javascript"`
PerlScript	`type="text/perlscript"`
TCL	`type="text/tcl"`
VBScript	`type="text/vbscript"`

Атрибут `type` может быть опущен, если документ содержит метатег, задающий язык программирования сценариев (см. *разд. 4.2*).

Различие между языками JavaScript и JScript состоит в том, что второй из них — это разработанная компанией Microsoft версия JavaScript для Internet Explorer. На практике этим различием обычно можно пренебречь.

В теге `<script>` также может быть указан атрибут `charset="кодировка"`, например, в виде `charset="Windows-1251"` или `charset="KOI8-R"`.

Необязательный атрибут `defer` тега `<script>` указывает, что браузер может отложить обработку сценария до момента полной загрузки документа. Такой сценарий не должен содержать кода, который выполняется в процессе загрузки документа.

Наконец, атрибут `src="URL"` определяет абсолютный или относительный URL *внешнего сценария*, хранящегося на сервере в отдельном файле. Файл внешнего сценария на JavaScript имеет расширение js и представляет собой обычный текст программы на JavaScript.

Внешний сценарий, хранящийся во вложенной папке files под именем hello.js, можно вызвать следующим образом:

```
<script type="text/javascript" src="files/hello.js"></script>
```

Объем и цели книги не позволяют нам детально ознакомиться с операторами языка JavaScript, но заинтересованные читатели могут это сделать, используя документ Glava_15\Javascript.rtf с компакт-диска. Заметим лишь, что JavaScript *различает* строчные и прописные буквы, и это следует учитывать при копировании и изменении примеров.

В основном тексте книги мы ограничимся несколькими примерами, иллюстрирующими рассмотренные аспекты применения JavaScript.

Ничто так не оживляет страницу, как случайно выбранная "фраза дня", обновляющаяся при каждом входе пользователя на страницу. Описав произвольный набор фраз и единственный оператор JavaScript так, как показано в листинге 15.4, мы получаем простейший скрипт для случайного выбора одной из этих фраз. Такой фрагмент кода может быть помещен в любое место документа.

Листинг 15.4. Сценарий выбора "фразы дня"

```
<script language="JavaScript">
var a=new Array (
  'Текст 1', 'Текст 2', … 'Текст N'
);
document.write (a[Math.round(Math.random()*(a.length-1))]);
</script>
```

Чтобы добавить произвольный текст вместо `'Текст 1',…,' Текст N'`, достаточно не забывать каждую строку заключать в апострофы и ставить после нее запятую. Листингу 15.4 соответствует пример, записанный на компакт-диске под именем Glava_15\RandomText.html.

Для определения имени, версии и других характеристик браузера, можно обратиться к свойствам объекта `Navigator`, например

```
document.writeln ('<b>Браузер:</b> '+navigator.userAgent);.
```

В дальнейшем сценарий может предпринять действия, зависящие от браузера. Этот пример сохранен на компакт-диске под именем Glava_15\UserAgent.html.

JavaScript может оказаться удобным и для автоматической подстановки в шаблон страницы некоторых свойств документа, таких как его текущий URL-адрес и дата последнего обновления. Видоизменив код нижнего информационного блока в шаблоне Glava_8\Template1.html так, чтобы указанные данные подставлялись автоматически, вы заметно сократите объем изменений, вносимых в каждый файл, созданный по этому образцу. Соответствующий пример записан на компакт-диске под именем Glava_15\StatusString.html.

Пример обработки события для проверки правильности заполнения формы приведен в документе Glava_13\MailForm.html.

JavaScript позволяет красиво решить и важнейшую задачу *навигации по сайту*, о которой мы подробно говорили в *разд. 6.5*. Действительно, вставив в каждую страницу своего сайта тег вызова внешнего скрипта вида

```
<script type="text/javascript" src="menu.js"></script>,
```

мы сможем все ссылки на новые пункты меню вносить *только в один* файл с именем `menu.js`, устроенный по следующему образцу:

```
document.writeln (' \
<nobr>[<a href="1.html">page 1</a>]</nobr> \
<nobr>[<a href="2.html">page 2</a>]</nobr> \
');
```

Здесь неразрывные пробелы в якорях ссылок и теги `<nobr>` запрещают браузеру разрывать строку пункта меню. Символы `\` в конце строк обязательны, т. к. позволяют перенести аргумент функции `writeln`. Обратите также внимание на вложение двойных кавычек в апострофы, позволяющее браузеру не запутаться в строках. Простейший пример организации такой навигации приведен на компакт-диске в папке Glava_15\menu.

Наша навигация обладает существенным недостатком — она окажется недоступной тем пользователям, у которых в настройках браузера отключен JavaScript. И пусть по статистике таких пользователей всего около одного процента, но это могут оказаться именно люди, способные оценить ваш сайт и создать ему ту или иную репутацию. То же самое касается корпоративных клиентов, которым "злой администратор" отключил все, что можно, в том числе, язык сценариев. Есть и второе соображение. Вероятно, на свете существуют "умные" поисковики, проиндексирующие сайт с навигацией на JavaScript,

но совершенно точно есть и такие, которым структура вашего сайта останется неизвестной, а этого допустить нельзя.

Несложный выход из ситуации есть. Достаточно к тегу скрипта в шаблоне страницы добавить тег альтернативного описания `<noscript>`, содержащий ссылки на *карту сайта* и *страницу, описывающую, как можно включить JavaScript* (или просто главную страницу):

```
<center><script type="text/javascript" src="./menu.js"></script>
<noscript>
<p>Меню недоступно. Вы можете <a href="javascript.html">включить
JavaScript</a> или перейти к <a href=map.html>карте сайта</a></p>
</noscript></center>.
```

Теперь общая картина сайта будет доступна и пользователю с отключенным JavaScript, и поисковой машине. Недостаток такой навигации — вносить ссылки на новые файлы придется не в одно, а в *два* места — документы `menu.js` и `map.html`. Но это намного лучше, чем в 100 или 200 мест. Посмотреть меню на JavaScript в работе вы можете, открыв файл Glava_15\menu2\index.html с компакт-диска.

Создание коллекций картинок, рассмотренное в *разд. 10.6*, также может стать значительно менее трудоемким, если использовать сценарий. Метод `document.write`, способный формировать текст страницы динамически, а также возможность перегружать из сценария изображения, позволяют сделать процесс почти автоматическим. Пример несложного скрипта фотогалереи с подробными комментариями приведен в документе Glava_15\PhotoGallery.html.

Документ Glava_15\Test_YN.html иллюстрирует еще один интересный вариант применения языка сценариев — создание на странице *интерактивного теста*. Внимательно изучив и модифицировав его исходный текст, вы сможете создать любой тест, ответы на вопросы которого предполагают выбор одного из двух вариантов.

15.5. Внешние баннеры и информеры

Сегодня трудно встретить сайт, на котором не было бы ни одного счетчика посетителей или рекламной картинки (*баннера*). По мере развития вашего интернет-проекта они, вероятнее всего, появятся и у вас. Чаще всего поддержка счетчиков и баннеров организуется с помощью языка сценариев, фрагмент кода на котором вы получаете, заполнив регистрационную форму на сайте рекламодателя.

В простейшем случае для установки счетчика или рекламы сценарий не нужен, достаточно поместить в тег ссылки, ведущей на требуемую страницу,

рисунок, который в свою очередь может быть расположен на внешнем сервере:

```
<a href="http://www.domain.ru" target="_blank"><img src="
http://www.domain.ru/img/banner.gif" width="88" height="31" border="0"
alt="Описание ссылки"></a>.
```

Здесь выбран стандартный формат кнопки сайта размером 88×31 пиксел. При щелчке по рисунку адресуемая ссылка открывается в новом окне.

Использование JavaScript в баннерах и счетчиках дает дополнительные возможности по сбору статистики о посетителях страницы и учету переходов с сайта, совершаемых посетителями, а также позволяет осуществлять динамическую смену баннеров (листинг 15.5).

Листинг 15.5. Пример кода для динамической смены баннеров

```
<center><script language="JavaScript"> var loc = ''; </script>
<script language="JavaScript1.4">try{ var loc = es-
cape(top.location.href); }catch(e){;}</script>
<script language="JavaScript">document.write("<sc"+"ript lan-
guage='JavaScript'
src='http://ad.ltxt.ru/bb.cgi?cmd=ad&hreftarget=_blank&pubid=ID_USER&pg=1
&r=js&ssi=nofillers&vbn=366&num=3&w=0&h=0&&ref="+escape(document.referrer
)+"&loc="+loc+"&nocache="+Math.round(Math.random()*999111)+"'>\n</sc"+"ri
pt>");</script></center>
```

Приведенный пример — реальный код баннера одной из текстовых рекламных сетей, в котором регистрационный номер пользователя заменен на строку ID_USER. Вставка такого кода в блок вашего сайта не представляет никаких трудностей.

Тем не менее, для уменьшения объема страницы, содержащей баннеры или счетчики, можно воспользоваться несложным приемом: если в коде скрипта, предоставленного вам в качестве образца, есть часть, заключенная в тег `<noscript>…</noscript>`, имеет смысл включить в код страницы только ее:

```
<a target=_top href="http://top.mail.ru/jump?from=ID_USER"><img
src="http://top.list.ru/counter?js=na;id=ID_USER;t=210" border=0
height=31 width=88 alt="Рейтинг@Mail.ru"/></a>.
```

Со всеми включенными скриптами, код этого популярного счетчика Mail.Ru увеличится не менее чем в 10 раз.

В последнее время распространилась еще одна форма применения скриптов — так называемые *информеры*, представляющие собой текстовый или графический блок информации, импортируемый со специализированного сайта, такого как **www.informer.ru**. Информер может содержать ленту новостей,

информацию о погоде или курсе валют, форму для проверки файла на вирус или мини-каталог коммерческих предложений. Чтобы задействовать информеры, достаточно зарегистрироваться на указанном или другом специализированном сайте.

Добавление в код страницы блоков с информерами позволяет оживить даже статический сайт, не следует только включать "для красоты" блоки информации, никак не соответствующей вашей тематике.

Скрипт, отвечающий за вывод содержания информера, может включать только тег `<script>`:

```
<script src="http://www.lenty.ru/export/mainnews.js"></script>
```

или два тега, первый из которых ссылается на тело программы-сценария, находящейся на сервере, а второй вызывает некоторую ее функцию, параметры которой обеспечивают соблюдение пользовательских настроек:

```
<script language="JavaScript"
src="http://pics.rbc.ru/js/informer_drw.js"></script>
<script language="JavaScript"><!--
  av_form_print(53051);
//-->
```

Следует учесть, что скорость загрузки страницы, переполненной баннерами и информерами, существенно снижается за счет многократных обращений к различным серверам.

15.6. Использование Java-апплетов

В *разд. 2.1* уже давалась краткая характеристика языка Java, на котором можно писать приложения-*апплеты*, встраиваемые в код Web-страницы.

В отличие от скрипта, апплет не связан непосредственно с HTML-кодом документа и в файле с апплетом не хранится его исходный текст. Java относится к *компилируемым* языкам, т. е. исходный текст программы переводится в один или несколько бинарных файлов с расширением class. Если приложение содержит множество классов, они могут быть упакованы в архив формата ZIP и подключены указанием атрибута `archive` в теге подключения апплета.

В файлах class хранится не код, пригодный для исполнения в какой-либо операционной системе, но так называемый *код виртуального процессора*, который каждая операционная система переводит в совместимые с ней инструкции. Разумеется, этот перевод требует дополнительных затрат времени и вычислительных ресурсов. Компенсацией является то, что однажды написанное приложение применимо во всех операционных системах, поддерживающих язык Java.

При использовании готового апплета мы не имеем доступа к его исходному тексту и можем лишь указать размеры области экрана, в которую браузер поместит окно апплета, а также передать ему в виде параметров информацию об исходных данных или режимах работы.

Все популярные браузеры поддерживают Java, однако, как и для языков сценариев, эта поддержка эта может быть отключена в настройках.

Исторически первым тегом подключения апплетов был блочный элемент `<applet>...</applet>`. Его содержимое — один или несколько элементов `<param>`, за которыми следуют произвольные блочные или текстовые теги, отображаемые только в том случае, если апплет не удается запустить.

Основной атрибут тега имеет вид `code="строка"`, где строковое значение содержит относительный или абсолютный URL, по которому можно получить файл класса. Альтернативный атрибут `object="строка"` также позволяет определить имя ресурса, содержащего специальное *сериализованное* представление состояния апплета.

Атрибуты `codebase`, `width`, `height`, `archive`, `name`, `alt`, `align`, `hspace` и `vspace` могут быть указаны в теге `<applet>` с теми же значениями, что и в `<object>` (см. табл. 15.1).

Пример вызова класса `Clock3D`, реализующего вращающиеся трехмерные цифровые часы, приведен в листинге 15.6.

Листинг 15.6. Вызов класса тегом `<applet>`

```
<applet code="Clock3D.class" width="120" height="80">
 <param name=fps value=18>              <param name=a1 value=12500>
 <param name=pixd value=25>             <param name=pixangle value=5>
 <param name=radius value=20>           <param name=roty value=-4>
 <param name=rotx value=0>              <param name=rotz value=0>
 <param name=style value=1>             <param name=color value=#00FF66>
 <param name=bgcolor value=#000000>
Апплет недоступен… </applet>
```

Выполнить этот пример в браузере можно, открыв документ Glava_15\Clock1.html.

Не все параметры, описанные в документации к конкретному апплету, должны получать начальные значения через тег `<param>`, обычно существуют значения по умолчанию. Если это не так, в описании апплета должно быть сказано о его обязательных параметрах. В качестве примера вызовем тот же

самый класс тегом `<object>`, изменив и упразднив значения некоторых параметров (листинг 15.7).

Листинг 15.7. Вызов класса тегом `<object>`

```
<object classid="java:Clock3D.class" codetype="application/java"
 width="100" height="100">
<param name=a1 value=16000>          <param name=rotx value=0>
<param name=rotz value=1>            <param name=color value=#000000>
<param name=bgcolor value=#FFFFFF>
Объект недоступен...</object>
```

Изменив значение параметра `rotz`, мы получили трехмерное вращение часов, кроме того, в листинге 15.7 заданы другие цвета фона и текста, а также в открывающей части тега `<applet>` указаны новая ширина и высота окна апплета. На компакт-диске этот пример доступен под именем Glava_15\Clock2.html. Для браузера Internet Explorer способ, показанный в листинге 15.7, менее надежен, чем использование тега `<applet>`.

Рассмотрим случай, когда апплет содержит несколько классов, упакованных в архив (как правило, типа ZIP или JAR). Для вызова такого апплета достаточно указать атрибут `archive`, указывающий браузеру, где искать остальные классы:

```
<applet code="ImageFilterDemo.class" arc-
hive="archives/ImageFilterDemo.zip" width="600" height="400">
 <param name="image" value="images/city1.jpg">
 <param name=filters value="GrayScale+Invert+Contrast+Blur+Sharpen">
</applet>.
```

Здесь параметры апплета задают URL рисунка и набор фильтров, которые будут применены к изображению. Посмотреть этот пример в работе можно, открыв документ на компакт-диске по адресу Glava_15\JavaApplet.html.

В нашем примере собственные средства управления апплета сводятся к набору кнопок, на самом деле набор интерфейсных компонентов, которые могут содержать Java-приложения, гораздо более широкий.

Несмотря на стандартность тега `<object>`, браузеры не всегда корректно загружают апплеты с его помощью. Так, указанный далее способ загрузки предыдущего апплета

```
<object archive="archives/ImageFilterDemo.zip" name="ImageFilterDemo"
classid="java:ImageFilterDemo.class" codetype="application/java"
width="600" height="400">…</object>
```

не был "правильно понят" браузерами Internet Explorer 7 и Safari 3.0.1 (см. пример Glava_15\JavaObject.html на компакт-диске).

Существует множество свободно распространяемых апплетов, решающих широкий круг задач. Отметим, что "перегрузка" сайта апплетами, в частности, использование Java для навигации или в качестве элементов дизайна, приводит к значительному замедлению загрузки страниц. Универсальный и платформенно-независимый код апплетов вообще интерпретируется медленно, что является неизбежной платой за эту независимость.

15.7. Использование технологии Macromedia Flash

Технология Macromedia Flash появилась почти одновременно с DHTML. "Ролик" Flash, хранящийся на диске в файле типа SWF, представляет собой объект, встраиваемый в Web-страницу, и является отдельной технологией Интернета, требующей установки собственного плагина-проигрывателя. Приложения Flash — своеобразные "интерактивные мультфильмы". Они не зависят от платформы, а созданные корпорациями Macromedia и Adobe программные средства Flash-разработки предоставляют в распоряжение Web-мастера множество инструментов. При правильном их использовании размеры файла SWF получаются не столь уж большими.

Наиболее популярные сферы применения Flash — интерактивные игры, мультипликация, сложные графические баннеры.

Подключить ролик Flash к своей Web-странице можно, пользуясь знакомыми нам возможностями тегов `<embed>` и `<object>`. В качестве примера выберем скромный по возможностям ролик, представляющий собой движущийся логотип Windows XP (листинг 15.8).

Листинг 15.8. Пример подключения Flash

```
<object codeBase="http://download.macromedia.com/pub/shockwave/cabs/
flash/swflash.cab#version=7,0,19,0"
 classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
 width="600" height="400">
 <param name="movie" value="images/intro.swf">
 <param name="quality" value="high">
 <embed src="images/intro.swf" quality="high"
   pluginspage="http://www.macromedia.com/go/getflashplayer"
```

```
   type="application/x-shockwave-flash" width="600" height="400">
<noembed>
  Ваш браузер не поддерживает технологию Macromedia Flash.
</noembed></object>
```

Атрибут `codebase` тега `<object>` содержит базовый для всех Flash-роликов URL, ведущий на сайт компании Macromedia. В конце URL содержится информация о требуемой версии проигрывателя, она может быть использована при обновлении. Атрибут `classid` указывает GUID требуемого для отображения в Internet Explorer компонента ActiveX. Стандартные для роликов параметры `movie` и `quality` позволяют передать в тег `<object>` информацию о URL воспроизводимого файла и необходимом качестве воспроизведения (другие допустимые значения параметра `quality` — `medium` и `low`). Из значения параметра `movie` видно, что подключаемый файл `intro.swf` находится во вложенной папке `images`. Обратите внимание на характерное для подключения роликов Flash вложение тега `<embed>` в тег `<object>`. Отчасти заданная тегом `<embed>` информация дублирует уже имеющуюся, дополнительно в теге указаны атрибуты `pluginspage`, адресующий страницу Macromedia, содержащую плагины, и `type`, задающий MIME-тип ролика Flash `"application/x-shockwave-flash"`. С одной стороны, такой подход позволяет браузеру ускорить загрузку приложений Flash, а с другой, — гибко прореагировать на ситуацию, когда требуемый плагин еще не установлен.

Выполнить этот пример в браузере можно, запустив файл Glava_15\Flash1.html с компакт-диска. Пример Glava_15\Flash2.html подключает более сложный ролик, имеющий интерактивные средства управления.

Internet Explorer последних версий обычно имеет уже установленный проигрыватель для отображения роликов Flash и покажет анимацию сразу же, другие популярные браузеры могут предложить загрузить соответствующий плагин.

Как правило, при вызове контекстного меню на активизированном Flash-ролике становятся доступными дополнительные команды, позволяющие изменять масштаб просмотра, управлять качеством отображения и последовательностью сцен.

Следует отметить существенные недостатки технологий Flash на современном этапе развития:

- сложность в изучении;
- несовместимость различных версий проигрывателей;
- ограниченная индексация текста в роликах Flash поисковыми системами. Следовательно, содержание такого сайта может быть недоступно для поиска,

изготовление же альтернативной текстовой версии сайта удвоит время его разработки;

- неприемлемо большое время загрузки Flash-ресурса при низкоскоростном соединении с Интернетом.

Подытоживая сказанное, заметим, что использование Flash оправдано в тех же случаях, что и встраивание любой "тяжелой" анимированной графики, — если поставленная задача не может быть решена другими средствами.

15.8. Программное обеспечение для работы с мультимедиа

Из популярных мультимедийных *плееров* следует отметить, прежде всего, стандартное приложение Windows Media, входящее в состав последних версий этой операционной системы, а также компактный и удобный плеер Winamp от компании Nullsoft, информацию о котором можно получить на сайте **www.winamp.info**. Последние версии графического браузера ACDSee (**www.acdsee.com**) также работают со всеми популярными форматами видео и звука.

Программы для профессиональной обработки звука позволяют, среди прочего, изменять форматы музыкальных файлов, а также управлять параметрами их сжатия. Лидерами в области профессиональной обработки и создания звуковых файлов заслуженно считаются программы Sony Sound Forge (**www.sonycreativesoftware.com**), Adobe Audition (**www.adobe.com/products/audition**), FL Studio (**www.flstudio.com**), а также Goldwave (**www.goldwave.com**) — простое и удобное приложение для записи с внешних источников и оцифровки звука.

Для обработки и конвертирования видео можно порекомендовать программы MPEG4 Direct Maker (**www.artech365.com**) и VirtualDub (**www.virtualdub.org**).

Кроме перечисленных плееров, для работы с видео популярны программы DivX Player и DivX Converter, поставляемые в комплекте с кодеками DivX (**www.divx.com**), QuickTime от Apple (**www.apple.com/quicktime**).

Мастер-модуль RealProducer (**www.real.com**) позволит легко подготовить видео для публикации на Web-сайте. Отличные возможности для редактирования потокового видео, предназначенного к размещению в Интернете, дает и программа VideoVista (**www.vista.it/videovista**).

Приложения WebcamXP (**www.Webcamxp.com**), WebCam Plus (**www.akcentplus.ru/Webcam**) предоставляют удобную систему организации и управления трансляциями с Web-камеры.

Среди популярных приложений для работы с языком сценариев JavaScript можно выделить JavaScript Editor, JavaScript Menu Builder и JavaScript Maker. Последние версии HTML-редактора CoffeeCup HTML Editor также содержат мощные средства поддержки скриптов и апплетов.

Работа с технологиями Java предполагает, прежде всего, установленный пакет Java 2 SDK, разработанный корпорацией Sun (**java.sun.com**). Java 2 SDK включает в себя компилятор, виртуальную Java-машину, библиотеку стандартных классов и несколько полезных дополнительных приложений. Стандартный дистрибутив Java SDK не содержит удобной среды разработки и отладки программ, поэтому для создания или редактирования апплетов можно порекомендовать программные продукты Sun ONE Studio и Borland JBuilder.

Изучить и применить технологии Flash можно с помощью широко распространенных программных продуктов Macromedia Flash и Adobe LiveMotion.

ЧАСТЬ IV

От HTML к САЙТУ

Знание HTML — не единственное, что требуется для создания страничек. Интернет растет все быстрее, и сайтов, предлагающих своим посетителям самые разнообразные материалы, становится все больше. Как сделать так, чтобы ваш сайт не утонул в этом море информации? Как написать текст, который понравится не только читателю-человеку, но и роботу поисковой системы? На какие ресурсы лучше ссылаться и с кем обмениваться ссылками? В последнее время эти вопросы особенно занимают как начинающих, так и опытных Web-мастеров, а советы по "раскрутке" интернет-проектов можно найти во множестве популярных изданий.

В последних главах книги мы постараемся найти ответы на эти вопросы. Изложенный материал представляет собой не "руководство по раскрутке", а скорее, подборку советов о том, как создать достойный сайт без больших затрат времени и изучения сложных специализированных технологий.

Глава 16

Делаем сайт правильно

Ни одна Web-страница не существует изолированно. Хотя на свете есть сайты, состоящие из одного-единственного документа, число их посетителей обычно также равно одному — это сам автор страницы. Мы научились создавать HTML-документы и связывать их ссылками, рассмотрели различные варианты структуры сайта и навигации по нему, настала пора поговорить о том, как поступать при создании реального, а не учебного интернет-проекта.

В этой главе мы обсудим "правила хорошего тона", которые следует соблюдать в отношении своих страниц, поговорим о том, где и как их можно размещать на всеобщее обозрение, а также изучим основы технологии, в обиходе называемой "раскруткой" сайта.

16.1. Прежде чем делать сайт

Начнем с самого главного вопроса — зачем вам нужен сайт и что конкретно он сможет предложить своим посетителям? Желание "просто сделать страничку" было похвально, пока вы изучали язык HTML. Теперь вы собираетесь создать что-то, что увидят люди. Сегодняшний Интернет переполнен информацией и действительно ценным в нем остается немногое — прежде всего, это уникальное авторское наполнение и свежие идеи. Поверьте, даже если ваш проект будет посвящен одному узкоспециальному вопросу, в котором вы хорошо разбираетесь (ремонт велосипедов, разведение кактусов, содержание сиамских кошек) — это лучше, чем безликий сайт-клон, забитый чужими статьями. Разумеется, никто не помешает вам завести на сайте страничку "о себе, любимом" и дополнить его любыми смежными материалами, но ценность в глазах посетителей ему будет придавать лишь та информация, первоисточником которой он является. Да и современные поисковые машины значительно "поумнели", так что выдавать копию за оригинал становится все труднее.

Тематика сайта и краткие, но четкие сведения о том, что именно он предлагает, должны быть представлены на главной странице, желательно на самом видном месте. Ведь пользователь, имеющий конкретную информационную потребность, обычно уделят лишь несколько секунд каждому просмотренному документу. В идеале каждая Web-страница должна быть четко ориентирована тематически и включать в начале своего HTML-кода краткое "резюме", раскрывающее ее тему и уникальность содержимого. Подробнее об этом мы поговорим в *разд. 16.2*.

Второй важнейший момент — продумайте структуру сайта сразу же. Когда он разрастется, "ломать" и переделывать его будет гораздо сложнее. В *разд. 6.5* мы обсудили основные схемы навигации, а в *разд. 8.4* — способ создания табличного шаблона для будущих страниц. Если никаких специальных ограничений нет, в общем случае следует предпочесть иерархическую навигацию и табличный шаблон, состоящий, как минимум, из "верха", "содержимого" и "низа". Предусмотрите в шаблоне такие важные "мелочи", как "резиновую" верстку страницы, позволяющую ей подстроиться под любое разрешение экрана (см. *разд. 8.4*), автоматическое указание на странице ее URL-адреса и даты обновления (*разд. 15.4*), подключение стилевого файла, позволяющее при необходимости гибко и быстро менять оформление (*разд. 14.2*), место для счетчика посещений и, возможно, баннеров (*разд. 15.5*). Разумеется, при разработке сайта вы будете использовать только относительные внутренние ссылки, что позволит легко переносить его на другой сервер.

Как бы хорошо ни была продумана структура сайта, со временем в нем обязательно появятся новые разделы. Это необходимо учесть уже на ранних стадиях разработки. Иерархическая навигация не создаст серьезных проблем с добавлением новых разделов, при других видах навигации упростить "переделку" меню может применение JavaScript, при условии, что оно сделано так, как описано в *разд. 15.4*.

Наконец, третьим ключевым вопросом является соответствие дизайна и содержимого. На этот счет автору приходилось слышать два диаметрально противоположных заблуждения:

- Дизайн не важен. Главное — это содержимое.
- Любой сайт должен иметь очень "крутой" и "навороченный" дизайн.

Повторюсь, что истина обычно находится посередине между крайностями. Дизайн зависит от того, какие цели вы ставите перед собой при создании страницы.

Так, если основное назначение сайта — информация, дизайн должен быть подчинен цели удобного и качественного ее предоставления, включая про-

думанную систему навигации по отдельным статьям или документам. Для сайта художественного плана, напротив, изысканное оформление может сыграть ведущую роль в оценке его посетителями. Для сайта развлечений (заведомо неконкурентоспособного, т. к. развлечениями Сеть переполнена) важны постоянное обновление и различные информационные "сюрпризы", позволяющие удержать на нем капризного искателя новинок.

Вероятнее всего, в ваших первых проектах дизайнерские изыски сведутся к созданию табличного шаблона, выбору цветов и фоновых изображений. Невозможно дать совет на все случаи жизни, но есть одно правило, которое я рекомендую вам соблюдать. Дизайн сайта должен быть *функциональным*, т. е. соответствовать его содержимому.

Если самое интересное у вас — это рисунки или фотографии, то дополненные множеством украшений страницы-галереи будет смотреться празднично и оцениваться по достоинству, не забывайте только учесть советы из *разд. 9.5*, сформулированные при обсуждении цветовых моделей сайта. Если вы начинающий программист, поэт или прозаик, жаждущий поделиться с миром своими творениями, то уместнее будет простой строгий дизайн, не отвлекающий от чтения текстов. Скромные "виньетки" по углам или графические разделители, разнообразящие большие текстовые блоки, лишними не будут никогда, а вот почти невидимый шрифт на серо-буро-малиновом фоне точно не вызовет у посетителя желания познакомиться даже с интересными текстами.

Стоит подумать и о *целевой аудитории*. Каким вы видите типичного посетителя сайта? Это студент или специалист, озабоченный поиском конкретных информационных материалов, ваш единомышленник по сетевой компьютерной игре или другому хобби, просто бездельник, желающий поразвлечься? Едва ли первый оценит пестрый "кричащий" дизайн, а последний — строгость и простоту.

С вопросом об основной аудитории связана и тема *развития* проекта. Как часто вы планируете его обновлять? Как смогут посетители узнать об этих обновлениях? Не стоит публиковать сайт, на которым вместо половины страниц выводится пустая заготовка с надоевшей всем картинкой "Under construction". Помните, что привести пользователя на страницу и удержать его — две совершенно разные задачи. Даже заплатив компании с сомнительной репутацией за кратковременную "раскрутку" такого полупустого сайта, вы потеряете однажды завоеванного посетителя навсегда. А удержать его поможет, в том числе, регулярное обновление содержимого. Не следует оптимистично заявлять о "ежедневных обновлениях" мало кому известного проекта. Для частного сайта небольшого или среднего объема куда более реальными будут 2–3 обновления в месяц. Важно, чтобы они не только фикси-

ровались на отдельной странице новостей, но и оставались доступными в "архиве обновлений" после того, как потеряют актуальность. Простейшая схема здесь такова — запись с прямой ссылкой на обновленное содержимое каждый раз добавляется в начало файла "ленты", при некоторых обновлениях часть новостей из конца файла "ленты" переносится в начало файла "архива". Таким образом, больше двух файлов повторно закачивать не придется.

Кстати, указанная частота обновления сайта позволит также "не забывать" о нем поисковым машинам — без всяких дополнительных усилий с вашей стороны.

Итак, главное в интернет-проекте — оригинальное и качественное содержимое, дополненное функциональным дизайном. Посмотрите со стороны на свое творение. Подумайте, будет ли оно кому-нибудь полезно. Если вы ответили на этот вопрос "да" — имеет смысл читать дальше. Тем более что кроме перечисленных есть и другие важные требования к сайту, соблюдение которых позволит ему жить долго и счастливо.

16.2. Управление структурой сайта и оптимизация HTML-кода

При написании HTML-кода страничек Web-мастеру приходится учитывать интересы как будущих посетителей-людей, так и поисковых машин, от которых в первую очередь зависит приток этих посетителей. Несмотря на постоянное развитие, поисковые средства Интернета еще далеки от совершенства, так что специально написанный для "обмана" поисковика спамерский ресурс имеет все шансы быть популярнее вашего честно сделанного. К счастью, популярность эта недолговечна, а крупные поисковики активно заносят такие ресурсы в "черные списки". Мы еще поговорим о поисковой специфике в *разд. 16.4*, а пока хотелось бы уяснить, в чем главное различие между интересами человека в отношении сайта и его "привлекательностью" для поискового робота.

Человеку нужна точная и исчерпывающая информация по теме, которая его интересует. Кроме того, он не хотел бы тратить время и средства на поиск или чтение онлайн. В идеале мгновенно найденный по запросу и точно соответствующий ему документ должен быть в архивном виде скачан из Интернета на компьютер пользователя.

Поисковая машина не "понимает" текста, который индексирует и заносит в свою базу данных. Она судит о нем по разработанным математиками и программистами алгоритмам, учитывающим, в основном, различные характери-

стики слов и словосочетаний из имеющегося в системе словаря. Это приводит к двум негативным последствиям:

- из-за несовершенства алгоритмов поиска или намеренного их "обмана" недобросовестными разработчиками может быть найдена не та информация, что нужна человеку. Разумеется, сами запросы тоже могут быть (и чаще всего бывают) несовершенны;
- огромная часть информации, содержащаяся в архивах или "приватных" защищенных паролями зонах Интернета, остается недоступной поисковым машинам, а следовательно, и рядовым пользователям. Кроме того, алгоритмы поиска любой нетекстовой информации находятся в зачаточном состоянии.

В нашем примере с размещенным в Интернете архивом, независимо от того, насколько интересны и важны упакованные в нем файлы, их найдут лишь те, кому вы дали прямую ссылку. Поэтому важнейший фактор будущей популярности сайта —*текст*, содержащийся в его HTML-коде.

При написании текста нужно учитывать следующее:

- пишите грамотно. Многочисленные орфографические и грамматические ошибки не только отпугнут перспективного посетителя, но и затруднят доступность информации через поисковые сервисы — большинство все-таки будет искать "рефераты", а не "рефираты". Для проверки текста во многих случаях достаточно стандартных возможностей Microsoft Word или других офисных программ;
- пишите красиво. Конечно, многие из нас не имеют больших способностей к написанию привлекательных и "читабельных" текстов. Тогда лучше обратиться за помощью к профессионалам или просто людям, хорошо владеющим слогом. Это более дорогой вариант, но в перспективе он себя оправдает;
- пишите конкретно. Например, в рекламном тексте пустые фразы об "уникальных предложениях на уровне мировых стандартов", "соответствии лучшим зарубежным аналогам" или "технологически продуманном решении программного модуля" сродни ни к чему не обязывающим речам политиков и не вызывают ничего, кроме зевоты. Если вы что-то предлагаете людям, просто объясните им, в чем суть и новизна вашего решения.

В коде страницы существуют особенно важные теги, на которые надо постоянно обращать внимание. Во-первых, это тег `<title>` (см. *разд. 4.2*), заголовки `<h1>`…`<h6>` (*разд. 5.6*) и метатеги `description` и `keywords` (*разд. 4.4*), активно используемые поисковыми машинами при оценке сайта, во-вторых, это атрибуты `alt` (*разд. 10.3*) и `title` (*разд. 4.2*) — едва ли не единственные в HTML средства альтернативного описания нетекстовой информации.

Работать с указанными тегами и атрибутами легче, если составить то, что профессионалы называют *семантическим ядром* сайта. Этот термин означает слова, словочетания и предложения, кратко и точно описывающие вашу предметную область.

На первом этапе вам следует подобрать ключевые слова. Как это сделать лучше всего? Для начала поставьте себя на место человека, который пытается найти в Сети вашу страничку. Что бы вы сами ввели в строке запроса? Какие слова точнее всего отражают содержание вашего сайта? Именно их следует выбрать в качестве ключевых. Набор из 20–40 слов можно считать достаточным. Учтите, что для каждой конкретной Web-страницы, в зависимости от ее тематики, набор ключевых слов может отличаться и пополняться. При этом все документы могут и должны быть пронизаны несколькими главными ключевыми словами, но на каждой отдельно взятой странице необходимо добавить словосочетания, характеризующие именно ее.

Допустим, что ваше хобби — путешествия, и вы можете создать интересный сайт о своих поездках. Слова "путешествия", "приключения", "туризм", "поездки", "курорт" напрашиваются сами собой. Логично, если проект будет содержать подразделы, посвященные конкретным местам, где вы были, а значит, "Алтай", "Испания", "Кипр" или более общие "Южная Сибирь", "Европа", "Средиземноморье" также не помешают. Внутри подраздела по Алтаю могут пригодиться "Белокуриха", "Чемал", "Шерегеш" и т. д. Разумеется, ваш сайт будет содержать не только впечатления от поездок, но и освещать практически важные для других туристов вопросы, такие как "цены на авиабилеты", "гостиничный сервис", "турфирмы Москвы" (которым стоит доверять) и т. п. Как правило, при всестороннем "мозговом штурме" на тему ключевых слов вы не только уясните себе структуру информационных разделов будущего сайта, но и увидите словосочетание-*слоган*, которое можно использовать в качестве названия проекта и его главной ключевой фразы. Лучше всего, если слоган, с одной стороны, не содержит малоинформативных слов (так называемых стоп-слов, например, "мои" в заголовке "Мои путешествия"), но, с другой стороны, слишком необычный и вычурный слоган едва ли соответствует поисковым ожиданиям большинства людей. Остановимся на варианте "Путешествия, приключения, туризм".

Кстати, если ваша страница не личная, а корпоративная, то это совсем не значит, что следует делать слоганом название компании или организации (если только вы не владеете известным и "раскрученным" брендом). Если ваша фирма называется "Мострансагент", об этом известно только очень небольшому кругу лиц и искать вас по названию никто не будет. А вот то, что вы занимаетесь "автомобильными перевозками в Москве", обязательно следует

отразить в слогане. Ну а название родной организации можно показать на графическом логотипе вверху страницы, не забыв снабдить его атрибутом `alt`.

При планировании структуры сайта и подборе ключевых слов предусматривайте отдельный документ для каждой самостоятельный темы, пусть даже объем страницы будет невелик. Единственная поездка в Испанию может со временем превратиться в несколько, а вот набор ключевых слов страницы и ее описание останутся. В идеале каждая страница должна иметь четкую тематическую направленность, однозначно описанную набором характерных ключевых слов. Возможно, вы уже догадались, куда потом девать эти слова — в метатег `keywords`.

Теперь об описаниях. Когда структура сайта ясна и продумана, а ключевые слова подобраны, можно заняться составлением коротких "резюме" каждой будущей (или уже существующей) страницы. Именно эти описания мы поместим в метатеги `description` и именно их большинство поисковых машин будут отображать в качестве основной информации о вашем документе.

Описания должны быть лаконичными и емкими (3–4 предложения). Увы, при отображении описаний поисковые машины будут обрезать их каждая по-своему, так что следует соблюдать краткость и выносить главное содержание в начало текста.

С другой стороны, описание должно завлечь и заинтересовать потенциального посетителя или покупателя. Возможно, не грех даже немного приукрасить содержимое, не нужно только писать бессмысленных фраз вроде "Здесь есть все" или "Зайдите на самый крутой сайт". Как и наборы ключевых слов, описания должны быть разными для разных страниц. Для главной страницы нашего сайта о путешествиях вполне подойдет следующее:

```
<meta name="description" content="Уникальные материалы опытного путешественника. Поездки на Алтай, в Европу и Азию. Интересные истории и приключения, фотографии курортов, лучшие турагентства Москвы, советы туристам, сравнение цен и услуг">.
```

Обратите внимание, что здесь использованы ключевые слова сайта. Для страницы об Алтае описание будет другим:

```
<meta name="description" content="Путешествия на Алтай - агентства, курорты, цены. Все, что нужно знать об Алтае, чтобы поездка была приятной. Белокуриха, Чемал, Шерегеш и другие туристические зоны в интересных историях и фотографиях">.
```

Словом, описание должно быть таким, чтобы заинтересовать будущего посетителя. Обязательно проследите за тем, чтобы в заголовке страницы, метатегах и основном тексте было как можно больше совпадающих ключевых слов. Не лишними они будут и в атрибутах `alt` тегов рисунков.

Теперь поговорим о каждом из "ключевых" тегов и атрибутов подробнее.

- Тег `<title>`. Ни в коем случае нельзя игнорировать этот тег. Во-первых, именно правильное наполнение элементов `<title>` может сильно повысить позицию сайта в выдаче поисковой системы, во-вторых, содержимое этого тега становится в большинстве браузеров названием закладки при сохранении ссылки на страницу. В заголовке страницы должен быть указан текст, отражающий ее тематику возможно более широким набором терминов. Не расходуйте место на бесполезную с точки зрения здравого смысла информацию. Оптимальная длина текста заголовка — 60–120 символов. Как показывает практика, главное здесь — емко и кратко передать *смысл* содержания страницы и ее *место* на вашем сайте. Для иерархической структуры сайта вполне подойдет аналогичная структура заголовков, например, на главной странице тег будет иметь вид `<title>Путешествия, приключения, туризм. Все нужное о поездках из Москвы</title>`. На странице, посвященной Алтаю, заголовок изменится на `<title>Путешествия, приключения, туризм. Алтай. Курорты, турагенства Москвы, цены, истории и фото</title>`, а в документе, рассказывающем о вашей поездке в Белокуриху, тег примет вид `<title>Путешествия, приключения, туризм. Алтай. Белокуриха, обо всем, что нужно для отдыха</title>`. Никогда не используйте один и тот же заголовок на нескольких страницах, но видоизменяйте его в зависимости от тематики. С другой стороны, основная тема сайта должна "красной нитью" присутствовать во всех заголовках. В теге `<title>` следует избегать "стоп-символов", таких как скобки, символы `*`, `=`, `!` и т. п.

- Заголовочные теги `<h1>…<h6>`. В тегах заголовков в четырех случаях из пяти располагают все, что угодно: от бесполезных высказываний вроде "Добро пожаловать на мой сайт!" до адресов и телефонов. А ведь по этим словам никто вашу информацию искать не будет. Именно содержимое заголовков имеет повышенный вес в глазах поисковой системы, а придать им удобный для вас вид легко с помощью стилей. Основной совет в отношении заголовков — используйте их по назначению. Титул, раздел или подраздел статьи, название важного блока содержимого, краткое информационное "резюме" страницы, — вот части сайта, естественным образом предназначенные к разметке тегами `<h1>…<h6>`. Не стоит ни совсем обходиться без заголовков, ни "загонять" в них весь документ, как советуют иногда доморощенные "специалисты по раскрутке".

- Метатеги `<description>` и `<keywords>`. Все основное о них уже было сказано ранее. Добавим, что в `description` должен содержаться не набор ключевых слов, а грамотный красивый текст, привлекающий внимание посетителя. Для ключевых слов существует тег `<keywords>`, пример кото-

рого мы еще не приводили. Ключевые слова главной страницы нашего воображаемого проекта могли бы иметь вид `<meta name="keywords" content="отдых, туризм, поездка, путешествия, развлечения, приключения, турне, тур, курорт, путевка, турагентство, виза, снаряжение, Москва, Алтай, Европа, Испания, зарубежье">`. Для вложенных страниц этот список может быть видоизменен и дополнен. Современные поисковые машины способны учесть различные формы одного и того же слова, так что писать отдельными словами "курорт", "курорты" и "курортам" не нужно.

- Атрибуты `alt` и `title`. Атрибут `alt` обязателен в тегах `<img>` для всех изображений кроме тех, что служат лишь элементами дизайна. Его содержимое индексируется современными поисковыми машинами и будет полезным для пользователей с отключенной графикой. Правила составления альтернативных описаний те же — краткость, ясность, передача основного содержания. Указывайте атрибут также в тегах `<applet>` и `<embed>`. При подключении внешних объектов тегом `<object>…</object>` аналогом `alt` служит содержимое элемента после вложенных тегов `<param>`. Атрибут `title` может сопровождать любой блочный или текстовый тег и содержит дополнительную информацию о теге, для которого написан. Поскольку содержимое `title` отображается браузером только при наведении курсора мыши на элемент, в теге `<a href=…>` этот атрибут может содержать полезную дополнительную информацию о ссылке, а в теге `<img>` — более подробное, чем в `alt`, описание рисунка или фотографии. Применять ли атрибут `title` к другим тегам — вопрос спорный, обычно большого смысла в этом нет.

Независимо от выбранной схемы навигации и способа создания страниц, при проектировании структуры сайта нужно учесть общие правила, делающие его "дружественным" как для посетителей, так и для поисковых роботов.

- Правило "трех кликов" — любые страницы сайта должны быть доступными по ссылкам с главной страницы не более, чем за три щелчка мышью. Если структура сайта этого не допускает, выполнить данное правило поможет "карта сайта", содержащая ссылки на все основные подразделы и страницы.
- Правило статического URL — любые динамические элементы адреса, не входящие в описанную в *разд. 1.4* спецификацию, затрудняют доступность страницы как для человека, так и для поисковой машины. Они целесообразны только там, где без этого нельзя обойтись. То же относится к навигации через скрипты.
- Правило объема — базы данных поисковых машин велики, но не беспредельны; их разработчиками всегда предусмотрены ограничения на объем

информации, индексируемой по одному URL. Верхняя оценка допустимого объема текста на одной странице — 10–20 Кбайт. Иначе говоря, при индексации сайта поисковая машина часто "захватывает" только верх страницы. Поэтому старайтесь размещать все важное как можно ближе к началу документа, не загромождая его графическими элементами, баннерами и бесполезными украшениями. Желательно, чтобы полный размер одной страницы, включая графику и звук, не превышал 100–120 Кбайт. Достичь этого можно, прежде всего, масштабированием и изменением качества JPEG-картинок (см. *разд. 10.8*), а также вынесением в отдельные файлы всех стилей и сценариев JavaScript. Последнее не только поднимет важный текст выше в документе, но и упростит работу по изменению стилей или сценариев.

- Правило стандартных технологий — не следует гнаться за модными новинками, пока они не станут стандартом (т. е. не будут поддерживаться большинством браузеров). Лишь малая часть посетителей сможет оценить "навороты", для большинства они только создадут лишние проблемы. В Сети нетрудно найти массу "крутых примочек" к сайту с подробными инструкциями по их подключению. Но порадует ли падающий на страничке "снег" человека, пришедшего за информацией? Если очень хочется, сделайте для эффектов отдельные странички с описаниями, как их применять.

Последний по порядку, но не по важности совет этого раздела — работайте с кодом страницы аккуратно, как можно чаще проверяя вносимые изменения в браузере. Только отточив базовый шаблон проекта и видоизменив его для конкретных разделов, можно приступать к наполнению заготовки интересными материалами. Не забывайте и то, что говорилось в *главе 5* о логическом и физическом форматировании текста.

16.3. Выбор хостинга

Еще одно важное решение, которое придется принять перед публикацией сайта — выбор *хостинга*, т. е. серверной и доменной площадки, заслуживающей чести разместить у себя ваше творение.

Идеальным вариантом был бы собственный Web-сервер с постоянным и высокоскоростным выходом в Интернет. У вас его нет и не предвидится? Тогда рассмотрим более реальные возможности.

Самое простое решение — воспользоваться услугами бесплатного хостинга. Наиболее популярные в России сервисы такого рода располагаются по адресам **www.narod.ru**, **www.newmail.ru**, **www.boom.ru**, **www.hotbox.ru**, **www.by.ru**, есть и еще целый ряд служб.

Преимущество подобного хостинга заключается только в его бесплатности. Недостатков намного больше. Как правило, платой за "бесплатный сыр" является автоматическое размещение на всех страницах баннеров компании-хостера, ограничения на общий объем сайта и размер одного файла, а также слабая поддержка дополнительных сервисов (или полное отсутствие таковых).

С другой стороны, для размещения странички на подобном сервере достаточно зайти по ссылке "Регистрация", выбрать себе свободное регистрационное имя (*логин*), не слишком короткий пароль и заполнить небольшую анкету. Скорее всего, регистрацию потребуется подтвердить, щелкнув по ссылке в письме, присланном на указанный в анкете адрес E-mail. Как правило, бесплатные хостинги предоставляют домен третьего уровня вида **логин.хостинг.ru**, так что для пользователя misha344 на хостинге **www.narod.ru** будет создан сайт с URL-адресом **http://misha344.narod.ru**.

Закачать файлы на созданный пустой сайт можно либо через предлагаемый хостером Web-интерфейс, либо через FTP-соединение. Когда файлов немного и структура сайта проста, вы вполне справитесь средствами Web-интерфейса, только зайдите по ссылке "Мастерская" или "Управление сайтом". В случае, когда важнее удобство работы, имеет смысл изучить работу с *FTP-клиентом*, которых существует множество — от специализированных мощных приложений до простых в обращении модулей, встроенных в популярные файловые менеджеры, такие как Far Manager. Кстати, можно обойтись и средствами самой Windows — для этого достаточно, имея подключение к Интернету, набрать в адресной строке любой папки адрес FTP-сервера, например, для **http://www.narod.ru** это будет **ftp://ftp.narod.ru**. После этого на экране появится диалоговое окно, приглашающее ввести ваш логин и пароль, а затем вы увидите файлы и папки своей странички точно так же, как на собственном компьютере, и точно так же сможете копировать их на сайт или обратно обычным перетаскиванием с помощью мыши.

Кроме доступа по FTP, есть и другие требования к бесплатному хостингу. Как минимум, он должен существовать более трех лет, обладать приемлемыми скоростью соединения и стабильностью работы, предоставлять не слишком мало дискового пространства (хотя бы 10–20 Мбайт). Неплохо, если хостинг обеспечивает готовые сервисы гостевой книги, форума, счетчика посещений и т. п.

Посетите несколько уже существующих проектов на различных бесплатных хостингах, посоветуйтесь с их владельцами, обратите внимание на скорость загрузки страниц и объем рекламы хостера — и сможете выбрать место себе по вкусу.

Перечисленные массовые хостинги обычно совмещены с почтовыми службами, так что регистрация на них дополнительно дает почтовый адрес вида **логин@хостинг.ru**. Несмотря на значительные ограничения, в пользу упомянутых сервисов говорит хорошая репутация компаний-хостеров и отсутствие с их стороны каких-либо "сюрпризов" в отношении пользователей.

Ко второй группе бесплатных хостингов относятся более серьезные площадки, предоставляющие возможность размещения CGI-приложений, прокрутки ваших баннеров, сбора статистики о сайте и т. п.

Из этой группы "условно-бесплатных" серверов популярны **www.fatal.ru**, **IpHosting.ru**, **www.best-host.ru**, **www.bestof.ru**, **www.jino-net.ru**, **hosting.promostudio.ru** и ряд других. Хостеры второй группы предъявляют более серьезные требования к вам и вашему проекту, например, наличие купленного платного домена, ограниченный срок бесплатного использования сервиса, тематические ограничения информации. Поэтому перед регистрацией на них имеет смысл внимательно изучить условия размещения сайтов.

Если вас смущают ограниченные возможности общедоступных сервисов и вы готовы потратить немного времени на поиск приемлемых для себя условий — этот тип бесплатного хостинга вам подойдет.

В лучшей ситуации находятся пользователи, решившие заплатить за размещение сайта. Основные преимущества платного хостинга следующие:

- качественный сервис, быстрая загрузка страниц, поддержка всех актуальных технологий Web-программирования, отсутствие навязанной хостером рекламы;
- покупка хостинга и оплата его на год-два вперед обычно дают возможность бесплатно или с существенной скидкой зарегистрировать доменное имя (обратите только внимание, что собственником домена должны становиться вы, а не хостинг-компания);
- при смене хостинга потребуется лишь изменить данные об этом на сервере, предоставившим вам домен; неприятная альтернатива этому — многочисленные и трудоемкие перерегистрации бесплатного домена в поисковых машинах и каталогах ресурсов;
- наконец, посетители склонны больше доверять сайтам, расположенным на платных хостингах. Так же поступают многие каталоги и поисковые машины.

Выбор конкретного платного хостинга определяется вполне практическими соображениями — кроме приемлемой для вас цены и скорости соединения с сервером, на котором располагаются сайты, следует обратить внимание на предлагаемые доменные имена. Так, например, URL-адрес **ВашеИмя.runet.ru**

явно лучше запомнится посетителям, чем **ВашеИмя.superprovider.ru**. А **ВашеИмя.ru** выглядит еще лучше, поэтому при выборе компании-хостера имеет смысл дополнительно заплатить за домен верхнего уровня. "ВашеИмя" также должно быть по возможности коротким, запоминающимся и однозначно произносимым.

Резюмируя, можно сказать, что для "пробы пера" вполне подойдет надежный и долго существующий бесплатный хостинг, для серьезных же, особенно коммерческих проектов, платный хостинг просто необходим.

Итак, технические вопросы решены, проект размещен и доступен, теперь самое время задуматься, как бы сделать так, чтобы потраченные силы не пропали впустую. Ведь даже неплохо сделанный и интересный сайт не принесет вам никакой пользы, если на него никто не будет заходить. Об этом мы и поговорим в следующем разделе.

16.4. Поисковые системы и ранжирование результатов поиска

Что делает рядовой пользователь, когда хочет получить информацию, но не знает, по какому адресу она расположена? Чаще всего он заходит на поисковый сервис и набирает запрос. Казалось бы, ничего сложного. На самом деле, качество такого поиска оставляет желать много лучшего. Разобраться в ситуации нам поможет знакомство с работой поисковой машины.

Во-первых, "поиск в Интернете" происходит вовсе не по ресурсам Сети, что технически невозможно за разумное время, а по базе данных поискового сервера. Специальная программа, называемая spider ("паук", "червь") или crawler ("краулер", "поисковый робот", "бот"), просматривая доступное по ссылкам содержимое Web-страниц, выделяет из текста каждого документа отдельные словоформы, удаляет неинформативные слова из стоп-словаря, а все остальные записывает во внутренний словарь с указанием для каждого слова адресов страниц, на которых оно встречается. Этот процесс и называется *индексированием* сайта. Человек, который хочет что-то найти в Интернете, делает запрос на странице поисковой машины. И по этому запросу ему выдаются ссылки на все документы из словаря, где это слово встречалось.

Вплоть до конца XX века, когда Интернет только начинал развиваться, представление о поиске было именно таким: у нас есть N документов, в которых встречается M слов. Достаточно проиндексировать эти документы и выдать список. *Ранжирование* найденных документов, т. е. имеющий принципиальное значение *порядок* их выдачи, определялось простейшими статистическими алгоритмами, на уровне "слово A встречается в документе B чаще, поэтому

оно считается с бóльшим весом и документ *B* следует показать на первом месте". Эти механизмы перестали работать, когда документов стало очень много. Кроме того, различные люди, понимающие, как устроен механизм ранжирования, начали создавать документы, специально предназначенные для того, чтобы попасть на первое место выдачи ("*поисковый спам*"). За несколько лет проблема отбора действительно важных и отвечающих запросу (*релевантных*) документов стала основной для всех поисковых машин.

Дело в том, что когда на каждое слово находятся миллионы или хотя бы десятки тысяч ссылок, важно оказаться в первых рядах выдачи пользователю. Ведь большинство людей, задав запрос, не смотрят более одной-двух страниц выдачи результатов. А Интернет сегодня — это не только информационное пространство, но и бизнес, в который вкладываются деньги. Поэтому в Сети есть множество сайтов, основное назначение которых — любой ценой завлечь посетителя, применяя для этого даже нечестные и нечистоплотные методы.

В настоящее время борьба разработчиков поисковых средств за качество поиска только набирает обороты. К сожалению, алгоритмы ранжирования все еще далеки от совершенства и улучшаются, в основном, введением новых слагаемых и весов в формулу, учитывающую релевантность. Так, крупнейшая в русскоязычном Интернете поисковая машина Яndex (**www.yandex.ru**) использует для оценки релевантности документа запросу *аддитивную модель*, согласно которой показатель релевантности складывается из взвешенных слагаемых, учитывающих встречаемость слов из запроса в документе, пар слов и текста запроса целиком, наличие всех слов запроса в документе и присутствие многих слов запроса в одном предложении. После этого, на втором этапе поиска производится оценка релевантности документов, отобранных на первом этапе. Похожие модели реализованы и во многих других поисковых системах.

Таким образом, определяющим фактором успешности поиска оказывается именно ранжирование найденных Web-страниц системой выдачи результатов поисковика. Не владея глубокими знаниями об этих механизмах, которые, к тому же, всегда являются корпоративными секретами, рядовые Web-мастера, чтоб их сайт не "утонул" в информационном море, способны сделать только одно — учесть при разработке своих сайтов факторы, которые *могут влиять* на положение документа в выдаче поисковой машины.

Эти факторы можно разбить на *внешние* и *внутренние*. К внутренним относят все то, что находится под контролем владельца сайта (текст страницы, ее оформление и т. д.). Перечислим и кратко охарактеризуем внутренние факторы ранжирования с точки зрения повышения "находимости" сайта. Отчасти

перечисленные далее рецепты будут дублировать положения *разд. 16.2*, что неудивительно, ведь мы и стремились сделать "правильный" HTML.

- *Объем текста на странице и текстовое оформление* — в общем случае следует стремиться к увеличению текстового наполнения сайта. Оптимальными считают страницы, содержащие 500–3000 слов или 2–20 Кбайт текста (от 2 до 20 тыс. символов). С другой стороны, большее количество текста на странице увеличивает ее видимость в поисковых системах за счет редких или случайных поисковых фраз.
- *Число ключевых слов на странице* — ключевые слова (фразы) должны встречаться в тексте как минимум 3–4 раза. Верхняя граница зависит от общего объема страницы. Наилучшие результаты наблюдаются, если ключевая фраза встречается в тексте несколько раз, а кроме того, слова из фразы встречаются несколько раз поодиночке.
- *Плотность ключевых слов* — показывает относительную частоту содержания ключевых слов в тексте, измеряется в процентах. Слишком низкая плотность приведет к тому, что поисковая система не придаст должного значения этому слову. Слишком высокая плотность способна включить *спам-фильтр* поисковой системы (т. е. релевантность страницы будет искусственно понижена из-за чрезмерно частого употребления ключевой фразы). Оптимальной считается плотность ключевого текста 3–7%. Для фраз, состоящих из нескольких слов, следует посчитать суммарную плотность всех ключевых слов фразы и убедиться, что она укладывается в указанные пределы.
- *Расположение ключевых слов на странице* — обычно чем ближе ключевое слово или фраза находятся к началу документа, тем больший вес они получают при оценке страницы поисковой системой.
- *Стилистическое оформление текста* — рекомендуется использовать заголовочные теги `<h1>`…`<h6>`, физическое (`<b>`) и логическое (`<strong>`) выделение ключевых слов, ключевые слова в теге `<title>`. Тег `<title>` должен не только содержать ключевые слова, но быть информативным и привлекательным. Как правило, в выдачу поисковой системы попадает 50–80 символов из тега `<title>`, поэтому размер заголовка желательно ограничить этой длиной.
- *Ключевые слова в тексте ссылок* — указание ключевых слов в качестве якорей исходящих ссылок обеспечивает преимущество при ранжировании.
- *Атрибуты alt для изображений* — корректное задание атрибутов "альтернативного текста" позволяет поисковой системе запомнить значение тега при индексации.

- *Метатеги <desciption> и <keywords>* — особенно полезен грамотно составленный `<description>`, многие поисковики отображают его информацию в результатах поиска и повышают релевантность документа, если содержимое тега соответствует содержимому страницы. Использование тега `keywords` современными поисковыми машинами ограничено, однако пренебрегать им не следует.

Независимо от конкретного способа организации Web-сайта, можно дать следующие рекомендации относительно его наполнения "легко находящейся" информацией:

- *Увеличение числа страниц сайта* — как правило, при этом улучшается его видимость в поисковых системах. Кроме того, постепенное добавление новых материалов на сайт может дать дополнительные преимущества при ранжировании.
- *Информативные меню* — ключевые слова в ссылках навигации позволяют придать дополнительный вес тем страницам, на которые ведет ссылка.
- *"Неглубокая" каталожная структура* — следует, по возможности, избегать в структуре сайта вложений каталогов глубиной более двух-трех; все важные страницы должны находиться в корневой папке сайта.
- *Тематическая и поисковая оптимизация страниц* — в идеале каждая страница должна быть четко определена тематически и оптимизирована под одну-две собственных ключевых фразы.
- *Главная страница сайта* — более трети посетителей, как правило, приходят с поисковой машины на главную страницу сайта, отсюда следует необходимость особенно тщательной и продуманной ее оптимизации под наиболее важные ключевые фразы предметной области.

Само по себе добавление сайта в индекс поисковой машины не представляет никаких проблем, более того, с течением времени оно произойдет автоматически, если на ваш проект есть хотя бы одна внешняя ссылка, и он отвечает требованиям поисковика (например, Яndex старается индексировать только те сайты, на которых есть текст, написанный по-русски, предпочитая страницы в доменной зоне ru всем остальным). Не дожидаясь этого неопределенного по времени события, следует сразу по размещении зарегистрировать сайт в четырех ведущих российских поисковых машинах: Яndex (**www.yandex.ru**), Рамблер (**www.rambler.ru**), Апорт (**www.aport.ru**) и русскоязычной версии Google (**www.google.ru**). Все они содержат ссылку вроде "Добавить сайт" и инструкции по добавлению. После того как это будет проделано, есть смысл перейти к ведущим зарубежным поисковикам, таким как **www.google.com**, **www.yahoo.com**, **www.altavista.com**, **www.infoseek.com**, **www.lycos.com**, **www.hotbot.com**.

Регистрация сайта в других "местах" Интернета будет рассмотрена в следующем разделе.

16.5. Внешние ссылки и увеличение ссылочной популярности

Из сказанного ранее следует, что основные факторы, влияющие на ранжирование, находятся под контролем разработчика страницы. В числе прочего это означает, что для поисковой системы становится невозможным отличить действительно релевантный документ от страницы, созданной специально под заданную поисковую фразу или сгенерированной автоматически. Единственный *внешний* фактор ранжирования, непосредственно не контролируемый разработчиком сайта, — количество и качество внешних ссылок на оцениваемую страницу. Этот фактор также называют *ссылочной цитируемостью*, *ссылочной популярностью* или *индексом цитирования*.

Простой подсчет количества внешних ссылок не дает достаточной информации для оценки цитируемости сайта (к тому же, эту величину легко "накрутить", указывая URL сайта везде, где только можно). Очевидно, что ссылка на ваш проект с сайта крупной компании должна иметь больший вес, чем ссылка на те же материалы с чьей-то домашней страницы, расположенной на бесплатном хостинге. Для "взвешенной" оценки цитируемости с учетом *качества* внешних ссылок на сайт поисковые системы вводят понятие *индекса цитирования* (ИЦ).

ИЦ представляет собой числовой показатель, оценивающий популярность ресурса, т. е. некоторое абсолютное значение "важности" страницы. Каждая поисковая система имеет собственные алгоритмы для подсчета ИЦ, как правило, детали их являются "секретом фирмы" и не публикуются.

Помимо обыкновенного ИЦ, представляющего собой абсолютный показатель, вводится понятие *взвешенного индекса цитирования* (ВИЦ), который является относительным значением и показывает рейтинг данной страницы относительно остальных документов, доступных поисковой машине. Термин ВИЦ в России обычно употребляют в отношении поисковой системы Яndex. Значения ВИЦ нигде не публикуются, и детали его подсчета известны только разработчикам алгоритма.

Как минимум, следует учесть влияние на индекс цитирования следующих факторов:

- *якорь ссылки на вас* — если текст, стоящий непосредственно в теге `<a>…</a>`, содержит релевантные ключевые слова, поисковая система воспринимает всю страницу как более релевантную;

- *релевантность ссылающихся страниц* — кроме ссылочного текста учитывается общее информационное содержимое ссылающейся страницы. Так, ссылки с тематически близких и более цитируемых ресурсов оцениваются поисковой системой как более весомые.

Существует также алгоритм тематического индекса цитирования (ТИЦ). Значения ТИЦ показываются в плагине Яndex.Bar, он рассчитывается для сайта в целом. В силу наибольшего охвата российских ресурсов Интернета поисковой машиной Яndex, значение ТИЦ лучше всего подходит для оценки русскоязычных проектов. Зайдя по адресу **yaca.yandex.ru**, вы можете получить специальный счетчик для оценки цитируемости своего сайта. Увы, первое время он неизбежно будет показывать ноль, пока информация о ссылающихся на вас ресурсах не станет известной Яndex.

Из "общемировых" показателей индекса цитирования сайта классическим считается алгоритм учета внешних ссылок PageRank (PR), запатентованный компанией Google. PageRank рассчитывается для каждой Web-страницы отдельно и определяется цитируемостью ссылающихся на нее страниц. Критерий, выражающий значимость страницы в PageRank, — ее *теоретическая* посещаемость. Предполагается, что пользователь начинает просмотр сайтов с некоторой случайно выбранной страницы. По ссылкам он переходит на другие страницы. При этом существует вероятность того, что посетитель покинет сайт и вновь начнет просмотр документов со случайного адреса (в алгоритме PageRank она принята равной 0,15 на каждом шаге). Соответственно, с вероятностью 0,85 пользователь переходит по одной из доступных на текущей странице ссылок, все они предполагаются изначально равноправными. При большом числе переходов пользователь побывает на популярных страницах много раз, а на малоизвестных — меньше. Отсюда можно заключить, что PageRank Web-страницы определяется как *вероятность нахождения* пользователя на ней; при этом сумма вероятностей по всем Web-страницам Интернета равна единице.

После масштабирования с PageRank можно работать как с конкретным числом. Так, в настраиваемой панели инструментов Google ToolBar каждая страница имеет целочисленное значение PR (PageRank) от 0 до 10, при этом PR определяется по логарифмической шкале. Познакомиться с инструментом Google ToolBar можно по ссылке **toolbar.google.com**, а узнать текущее значение PR или ТИЦ для любого сайта без установки плагинов — по ссылке **besa.ru**.

Можно сделать вывод, что алгоритм ранжирования построен на следующих принципах:

- каждая Web-страница, даже если на нее нет внешних ссылок, изначально имеет малое ненулевое значение PageRank;

- каждая страница, имеющая исходящие ссылки, передает часть своего PageRank страницам, на которые ссылается. При этом переданный PageRank обратно пропорционален числу ссылок на странице: чем ссылок больше, тем меньший PageRank передается по каждой из них;
- PageRank передается не полностью, на каждом шаге происходит "затухание", принятое как вероятность того, что пользователь начнет просмотр с нового случайно выбранного URL-адреса.

После того как поисковая система, используя текстовые критерии, отобрала релевантные документы, они могут быть отсортированы согласно PageRank. Учитывая, что документ, имеющий большее число качественных внешних ссылок, должен содержать более ценную информацию, PageRank выделяет в поиске те страницы, которые наиболее популярны.

PageRank не используется непосредственно в алгоритме Google или других поисковых машин. Основная причина этого в том, что данный показатель характеризует лишь количество и качество внешних ссылок на сайт, не учитывая ссылочный текст и информационное содержимое ссылающихся страниц, между тем именно эти факторы должны быть наиболее весомыми при ранжировании. Предполагается, что ранжирование Google осуществляется на основе модифицированного "тематического PageRank", учитывающего только ссылки с тематически связанных страниц, однако детали этого алгоритма в настоящее время известны лишь разработчикам Google.

На практике значение PR удобно в двух случаях:

- при быстрой оценке уровня популярности сайта. Для сайтов среднего объема и популярности наиболее типично значение PR от 2 до 5. Значения PR от 8 и выше встречаются только у порталов крупных компаний;
- при косвенной оценке уровня конкуренции по поисковому запросу. Так, если в первой десятке выдачи поисковой системы находятся сайты со значением PR 6–7, то сайт с PR=4 по данному запросу практически неконкурентоспособен.

Распространенный способ увеличения ссылочной популярности сайта — добавление его в *каталоги ресурсов*. Среди огромного множества существующих каталогов ценность представляют лишь те, которые размещают прямую ссылку на сайт с помощью стандартного тега `<a>`. Как правило, поисковые системы учитывают лишь прямые ссылки из каталогов. Также имеет значение способ регистрации — только регистрация, проходящая под контролем человека, дает удовлетворительные результаты. Каталоги "Free For All" (FFA), автоматически размещающие ссылки любой тематики, всегда нерелевантны и чаще всего находятся в "черном списке" поисковых систем.

Действительно полезных "*белых*" каталогов, размещающих прямую ссылку на ваш ресурс и не требующих размещения обратной, насчитывается всего несколько сотен. Из зарубежных каталогов, имеющих большое количество серверов-"зеркал", прежде всего следует назвать проекты DMOZ или Open Directory Project (**www.dmoz.org**) и Yahoo (**www.yahoo.com**). Список наиболее известных "белых" каталогов России можно получить, например, по адресу **wlist.com.ru**.

Всегда полезна регистрация проекта в каталогах, близких вам тематически или регионально.

Эффективным для увеличения ссылочной популярности может оказаться и *обмен прямыми ссылками* с другими сайтами. Однако неквалифицированный подход при этом может дать результат обратный ожидаемому. Следует также помнить, что поисковые системы в общем "не приветствуют" обмен ссылками, поскольку в большинстве случаев он имеет цель изменить выдачу поисковой системы и не несет ничего полезного пользователям. Здесь можно сформулировать следующие правила:

- есть смысл меняться ссылками только с тематически связанными ресурсами, нетематический обмен почти всегда малоэффективен;
- страница, где размещается ваша ссылка, должна быть качественной (т. е. иметь PageRank не менее 2) и доступной для индексации поисковыми системами, размещаемая ссылка должна быть только прямой, общее число ссылок на странице не должно превышать 40–60 и т. д.;
- без особой необходимости не следует создавать собственных каталогов ссылок. Во-первых, это препятствует "удерживанию веса PageRank" внутри сайта, во-вторых, создание действительно полноценного каталога для средних и даже крупных сайтов в настоящее время нереально по трудоемкости, в третьих, многие поисковые системы сегодня искусственно занижают веса ссылок из каталогов. Имеет смысл сосредоточить все внешние ссылки на одной или нескольких страницах, имеющих ненулевой PageRank, ограничить число внешних ссылок, запретить индексацию ссылочных страниц с помощью специальных директив файла robots.txt, либо применяя JavaScript-ссылки, не индексируемые поисковыми машинами. Ряд поисковых машин учитывают действие нестандартного тега `<noindex>…</noindex>`, который предписывает роботам не следовать по URL-адресам, включенным в его содержимое. Как правило, поисковые системы отслеживают взаимные ссылки, и прямой обмен по принципу "сайт на сайт" неэффективен, размещать ответные ссылки выгоднее на другом домене/сайте, отличном от продвигаемого. Например, схема "сайт *A* ссылается на сайт *B*, сайт *B* на *C*, а *C* на *A*" повышает PR более эффективно;

- ссылочный текст должен быть разнообразным, одинаковые внешние ссылки на проект могут интерпретироваться поисковыми системами как попытка спама. То же относится к описаниям сайта, даваемым в любые каталоги.

Уже зарегистрированный в поисковиках и каталогах сайт может получать заметную долю посетителей с различных *рейтингов ресурсов*. Следует регистрироваться лишь в нескольких популярных рейтингах. Как правило, присутствие в рейтинге требует размещения обратной ссылки на него, что замедляет загрузку ваших страниц и "украшает" ее множеством лишних кнопок. При регистрации в рейтинге выбирайте категорию, наиболее точно отражающую содержимое вашего ресурса. Если таких категорий несколько, выбирайте ту, в которой наименьшее число сайтов-конкурентов.

Продуктивный, но кратковременный рост посещаемости могут дать и популярные сетевые *доски объявлений*. Самое лучшее здесь — воспользоваться специализированной программой автоматического добавления объявлений, с помощью которой без особых усилий можно подать одно объявление на сотни досок — достаточно только заполнить форму.

В заключение заметим, что количество и качество информации на сайте должны всегда идти на шаг впереди рекламы, ведь "быстрая и дешевая раскрутка" не бывает долговременной и эффективной. Никогда не забывайте, что Web-страницы создаются для людей, а не для поисковых систем. Привести посетителя на сайт — только один шаг, сделать который помогут технологии или специалисты. Удержать посетителя на сайте или превратить его в покупателя можете только вы сами, и путь к этому — грамотное наполнение сайта содержимым, интересным для человека.

Глава 17

Как делать не нужно?

Пусть вы — еще не опытный Web-мастер, но народная пословица не зря гласит, что умный учится на чужих ошибках. В этой главе мы рассмотрим самые распространенные ошибки начинающих Web-мастеров, что поможет вам избежать их при разработке собственных проектов. Для удобства в *разд. 17.4* приведены краткие ссылки на программы, способные помочь при проверке сайта и его отдельных элементов.

17.1. Ошибки структуры сайта

Начнем с главного. Любой ваш посетитель попадает на чужой для него сайт и не может ориентироваться на нем так же хорошо, как вы. Обеспечьте ему простую и надежную информационную поддержку. Это не значит, что от "украшений" в виде графической карты или кнопок обязательно следует отказываться. Это означает лишь, что обычные текстовые ссылки с информативными якорями должны присутствовать на ваших страницах хотя бы как альтернатива, а все графические элементы навигации необходимо снабдить атрибутами `alt`. Если навигация использует JavaScript, Java или другие "внешние" по отношению к HTML технологии, то поисковые системы могут не получить статические адреса страниц, соответственно, документы не будет проиндексированы. Решение данной проблемы уже обсуждалось — "карта сайта", доступная с каждой его страницы и полностью отражающая его структуру (*разд. 15.4*).

Независимо от того, работаете вы "вручную" в Блокноте, в простейшем HTML-редакторе или WYSIWYG-приложении, следует раз и навсегда выбрать для себя *единый стиль* написания имен тегов и атрибутов (или настроить на этот стиль свой редактор). Можно долго спорить о том, сколько пробелов отступать вправо при открытии нового блочного тега, и о том, прописные

или строчные буквы следует предпочесть в названиях тегов и атрибутов, бесспорно лишь то, что строгое соблюдение собственного стиля избавит вас от многих часов поиска ошибок в коде.

Кстати говоря, все имена и расширения файлов или папок нужно давать только *строчными буквами*, чего Windows не делает или делает крайне неаккуратно. Назвав документ File.html и сославшись на него ссылкой вида `<a href="file.html">якорь</a>`, вы получите от сервера сообщение "Страница не найдена", хотя на локальной Windows-машине все работало.

После этих обязательных технических рекомендаций, перечислим другие распространенные дефекты организации сайтов.

- Недостаточные или явно избыточные, многократно дублируемые средства управления и навигации по сайту; отсутствие его четкой структуры, сложные перекрестные ссылки, отсутствие ссылочной карты сайта.
- "Фиксированная" верстка страницы с указанием всех важных размеров в пикселах. Поверьте, что таблица шириной 700 пикселов выглядит при разрешении монитора 1600×1200 просто ужасно. Общую ширину таблиц шаблона, по возможности, указывайте в процентах. Столбец таблицы с основным содержимым всегда следует делать "резиновым", чтобы шаблон мог подстроиться под разрешение монитора пользователя. Естественно, такими же "резиновыми" должны быть другие строки этого столбца, их наполнением может быть фоновый цвет или рисунок, произвольно масштабируемый по горизонтали.
- Верстка всей страницы в одной таблице. Большинство браузеров по-прежнему показывают таблицу, только полностью загрузив ее содержимое. Всегда лучше разбить основную таблицу страницы на несколько отдельных таблиц. Этот несложный прием даст очевидный выигрыш в скорости загрузки. Расставляйте все размеры таблиц и их отдельных ячеек, не злоупотребляете атрибутом `nowrap`.
- Разработка громоздких "начальных страниц", с применением фреймов, Flash-технологий, "тяжелой" графики и т. д. Средний пользователь заходит на сайт на 2–3 щелчка мышью. При разработке бесполезной начальной страницы с картинками и надписью "Enter" один переход вы теряете сразу.
- Цвета ссылок, близкие к цвету основного текста, одинаковые цвета для посещенных и непосещенных ссылок, удаление подчеркивания у ссылок. Увы, 99% пользователей привыкли видеть ссылку подчеркнутой, как бы ваш дизайнерский вкус этим не тяготился. Они привыкли также легко отличать по цвету ссылку от текста, а ссылку, по которой они уже переходили, от непосещенной.

- Ссылка на страницу на этой же самой странице. Обычно проблема вызвана тем, что на сайте используется одно и то же навигационное меню. Это неплохо. Но когда меню копируется в код страницы, которая в нем упоминается, этот пункт-ссылка должен быть отключен или выделен таким образом, чтобы можно было понять, где мы сейчас находимся. При создании навигации на JavaScript, меню, описанное в *разд. 15.4*, также можно видоизменить для соблюдения этого принципа.
- Наличие ссылок на несуществующие ресурсы, как в рамках сайта, так и за его пределами. Не ссылайтесь на страницу, которой еще нет. В крайнем случае, перед этим создайте для нее пустую "заготовку" (которая тоже будет раздражать пользователя). Что касается внешних ссылок, то их нужно периодически проверять средствами, описанными в *разд. 17.4*.
- Отсутствие обновлений на протяжении длительного времени, игнорирование информации об актуальности представленного содержимого и сроках его обновления, несоблюдение заявленных сроков (лучше не указывать их совсем, если не уверены, что сумеете соблюдать).
- Затруднение доступа к старым материалам, которые недавно были на главной странице или странице новостей. Удобный в обращении "архив" — необходимая часть любого развивающегося сайта. Как минимум, реализуйте прием с "лентой новостей", описанный в *разд. 16.1*.
- Отсутствие услуг "downloads" (т. е. загрузки демо-версий рекламируемых программ или произведений), невозможность получения прайс-листов для перечисленных на сайте товаров, описание услуг и продукции без указания конкретных технических характеристик или диапазона цен.
- Недостаточное администрирование и фильтрация информации в средствах интерактивного общения (гостевые книги, форумы, доски объявлений). Никем не управляемый форум или гостевая книга быстро превратятся в "свалку" бессмысленных спамерских сообщений.
- Бесконтрольное появление новых окон браузера. Проблему следует считать серьезной, т. к. большинство пользователей не хотят "засорять" свой экран лишними окнами. Если пользователю нужно будет открыть новое окно, он сделает это сам. Программируйте открытие новых окон лишь в особо важных случаях, предупреждая об этом пользователя.
- Невозможность поиска по сайту. У любого современного сайта должна быть поисковая машина, в простейшем случае это легко организовать с помощью бесплатного сервиса компании Яndex (**company.yandex.ru/forms**).
- Изобилие неудачно спроектированных форм. О "правильном" дизайне форм мы говорили в *разд. 13.4*.

- Отсутствие контактной информации. Любой сайт призван сообщить миру нечто о вас или вашей компании. Поэтому предусмотрите обратную связь мира с вами, лучше всего вынести важнейшие координаты на главную страницу и более подробно продублировать их на специальной странице контактов.

17.2. "Неправильное" содержимое

При создании текстового содержимого избегайте "недозволенных" приемов, таких как наполнение страницы скрытым текстом или "накачка" ее популярными ключевыми словами. Возможно, это даст кратковременный эффект. Может быть, вы даже не сразу попадете в "черный список" поисковых систем. Но репутация сайта будет подорвана необратимо, и обманутый пользователь не вернется к вам никогда.

Обратите внимание и на описанные далее типичные ошибки работы с содержимым.

- Тезисный и рекламный характер представляемой информации без каких-либо конкретных данных. "Общие слова", ни к чему не обязывающие и ничего не предлагающие, на сайте плохи всегда.
- Отсутствие на главной странице сайта краткого описания того ценного, что он предлагает для посетителей и чем отличается от своих конкурентов.
- Нечитаемые шрифты — очень распространенная проблема. Большинство пользователей недовольны "экзотическими шрифтами", их фиксированным размером (результат работы в WYSIWYG-редакторе), низким контрастом шрифта и фона. Обязательно проверьте, как выглядит страница при различных размерах шрифта в браузере и разных разрешениях экрана. По возможности, ограничьтесь только стандартными для всех браузеров шрифтами (см. *разд. 5.2* и *14.4*).
- Применение "неродных" для Интернета стандартов представления информации, таких как документы Word или Adobe. Возможно, современный браузер с помощью плагина отобразит такой документ, но весьма вероятно и то, что плагин у пользователя не установлен. Для размещения в Интернете существуют форматы, описанные в *главе 3*.
- Отсутствие или применение не по назначению метатегов, заголовков, шрифтовых выделений и атрибутов альтернативного описания. Проблема подробно изучалась в *разд. 16.2*.
- Избыточные и бессмысленные элементы страницы. Учтите, что "снегопад" или "листопад" поверх текста, строка статуса, в которой "дергается" посторонняя надпись, скрипт, указывающий сегодняшнюю дату, только

раздражают большинство пользователей и замедляют загрузку страницы. Это же относится к многочисленным Java- и Flash-элементам. "Используйте внешние технологии только там, где без них нельзя обойтись" — еще раз напомним уже указанное в книге правило.

- Избыток на странице внешних блоков, таких как баннеры, счетчики посещений и информеры. Серверы, с которых они загружаются, очень часто медленны или постоянно перегружены. К сожалению, совсем без указанных элементов обойтись нельзя. Поэтому ограничивайтесь их небольшим количеством на одной странице. Не следуйте рекомендациям различных рейтингов и баннерных сетей по установке посторонних элементов вверху страницы. Счетчик, даже однопиксельный, всегда ставьте внизу.

Основные ошибки работы с графическим содержимым мы обсудим подробнее в следующем разделе.

17.3. Ошибки при работе с графикой

Сначала вернемся к технологиям работы с графикой. Браузеры хранят загруженные из Интернета рисунки в особой папке — *кэше*, и при повторной загрузке изображения стараются брать его оттуда. Поэтому на страницах сайта лучше не размещать рисунков с одинаковыми именами, находящихся в разных папках, т. к. это может "запутать" систему кэширования и привести к неправильному отображению графики. Отличный вариант — поместить все изображения в одной специально выделенной папке, для сокращения объема кода ей достаточно дать имя img или даже просто i. Кстати, подумайте о коротких именах и для других папок, на которые много ссылок из кода. Не забывайте о правильных значениях атрибутов `width` и `height` для всех тегов `<img>`, иначе при загрузке каждого нового изображения все уже открытое содержимое будет сдвигаться.

Современные браузеры умеют закачивать картинки параллельно. Обычно одна большая картинка размером 80 Кбайт грузится медленнее, чем четыре картинки по 20 Кбайт. Часто целесообразно разрезать большие картинки с помощью графического редактора. Слишком большое количество маленьких кусочков картинки может привести к обратному эффекту — ведь браузер параллельно закачивает ограниченное число файлов, да и трудоемкость "нарезки" графики всегда высока. Поэтому "режьте" разумно, зная, что уменьшения размеров файла картинки можно достичь и иными способами — сокращением количества цветов для GIF или снижением качества JPEG. Последовательность технологических этапов обработки изображений была описана в *разд. 10.8*.

Не следует также помещать в ячейку таблицы или другой контейнер более одного изображения. Это чревато непредсказуемым поведением всего табличного шаблона.

При использовании любых графических ссылок, нельзя забывать об альтернативной текстовой навигации для пользователей, отключивших загрузку рисунков. Не нужно злоупотреблять медленно загружающимися анимированными рисунками.

Избыток на сайте сложных графических элементов влечет за собой увеличение времени загрузки и мешает концентрации внимания на размещенной информации. С другой стороны, доминирование больших текстовых блоков над элементами оформления мешает чтению за счет снижения концентрации внимания и усложняет навигацию.

Как уже обсуждалось в *разд. 9.5*, одна из типичных ошибок начинающего разработчика — "дикая" цветовая палитра. Цвета должны гармонировать, а не подавлять друг друга. Не рекомендуются чрезмерно яркие, ядовитые цвета, поскольку посетителя они раздражают. При выборе цветов необходимо также учитывать их контрастность. При наличии графических обоев постарайтесь сделать так, чтобы фон не отвлекал от основного содержимого страницы и не мешал ее прочтению.

Любые графические элементы на странице должны быть *тематически обоснованы*. Например, размещайте на странице фотографии людей, имеющих отношение к материалам сайта, а не взятые с компакт-диска фотоснимки моделей. Неуместное изображение только отвлечет внимание посетителя от важного материала. Вряд ли кто-то поверит в выдающиеся качества вашего товара только потому, что на посвященной ему странице изображена неизвестная юная девушка. Лучше покажите сам товар.

Этот же принцип относится к графическим украшениям. Летающий по странице череп или бегающий по графическому разделителю котенок нисколько не улучшат восприятие страницы, если на ней рассказывается о каком-нибудь приеме программирования.

Очень распространенный элемент дизайна, к которому, тем не менее, нужно относиться с осторожностью — графический заголовок ("шапка"), занимающий всю ширину верхней части страницы и содержащий логотип компании, название и тому подобную информацию. Помните, что верх страницы — очень ценное место, где можно разместить самые важные ключевые слова и описания. В случае графического изображения, да еще без атрибутов `alt` и `title`, это пространство пропадет впустую. Бывают и совсем нелепые ситуации: заголовок страницы содержит текст, но выполненный в виде картинки и не учитываемый поисковиками. Несложный выход — комбиниро-

ванный вариант, при котором графический логотип занимает только часть ширины страницы, а оставшееся место занято текстовым описанием с ключевыми словами.

Не нужно злоупотреблять и различной скрытой графикой, такой как графические счетчики размером 1×1 пиксел. Обилие скрытых графических ссылок может быть воспринято поисковыми системами как попытка обмана и привести к внесению сайта в "черный список".

17.4. Средства для проверки сайта

Сначала поговорим об избыточности HTML-кода, которой особенно грешат WYSIWYG-редакторы. Например, для подобного редактора типичен код

```
<font face="Arial"><font size="+1">Текст</font></font>
```

вместо

```
<font face="Arial" size="+1">Текст</font>.
```

Еще хуже выглядит

```
<font face="Arial">текст, </font><font face="Arial">еще текст</font>
```

вместо

```
<font face="Arial"> текст, еще текст</font>.
```

Увы, существующие WYSIWYG-редакторы пока нерациональны. Как получить разумный и компактный код? Средство одно — знать HTML и применять только редактор тегов.

Кстати, по поводу приведенного примера с тегом `<font>`. Прежде чем использовать любые средства физического форматирования, описанные в *разд. 5.2*, задумайтесь — а не проще ли применить стили? Обычно это не только удобнее, но и серьезно сокращает размер кода.

На протяжении всей книги мы ставили значения всех атрибутов в кавычки. Несмотря на пожелания спецификаций, с точки зрения браузеров кавычки нужны только в атрибутах `alt` и `href`, а также в метатегах и в случаях, когда значение атрибута состоит более чем из одного слова. Во всех остальных ситуациях кавычки можно не ставить и код

```
<font face=Arial size=+1>Текст</font>
```

будет прекрасно понят всеми браузерами.

Проверить код на элементарные ошибки вам помогут такие приложения, как CSE HTMLValidator Professional (**www.htmlvalidator.com**) и AceHTML (**software.visicommedia.com/en/products/acehtml**). Протестировать свою страничку на соответствие спецификациям HTML можно и в режиме онлайн, например, по адресу **uitest.com/en/check/**.

Приходится встречать еще один оригинальный путь разработки сайтов: страница делается в визуальном редакторе, а затем полученный код "чистится" одной из программ *сжатия HTML-файлов*, такой, например, как Advanced HTML Optimizer (**www.aevita.com/Web/htmlopt**), HTML-Optimizer (**www.tonbrand.nl**) или WordHTML CV. Что ж, использовать их не возбраняется, однако учтите, что качество "чистки" далеко не всегда высокое, а модифицировать "подчищенный" код будет трудно.

Классический образец дурного тона при Web-разработке — ориентация страницы под конкретный браузер (браузеры). Вместо этого следует придерживаться стандартного "ядра" HTML, которое я старался описать в этой книге, и проверять получаемую страницу хотя бы в браузерах Internet Explorer, Opera и Mozilla Firefox.

Необходимо тестировать страницу при различных разрешениях экрана. Облегчить этот процесс могут известные программы BrowserSizer и BrowserMaster, имитирующие смену разрешения в широких пределах. Важно, чтобы у пользователя не возникало потребности в горизонтальной прокрутке. С осторожностью применяйте любые указания размеров в абсолютных единицах (см. *разд. 14.3*) — ведь настройки экрана и браузера пользователя нам неизвестны. Онлайн-сервисы проверки внешнего вида страницы в различных браузерах доступны по адресам **www.danvine.com** и **www.browsercam.com**.

Наконец, при наличии на сайте любых внешних ссылок следует периодически проверять их "жизнеспособность". Помочь в этом могут программы Xenu's Link Sleuth (**home.snafu.de/tilman/xenulink.html**), Sitemapper (**www.trellian.com/mapper**) и HTML Link Validator (**lithopssoft.com**).

И помните главное — страница создается для пользователя, она должна удовлетворять его информационную потребность. Только этот принцип поможет превратить сайт из бесполезного развлечения в мощное современное средство поддержки вашего бизнеса, общения и отдыха.

Как заметил "гуру" в области Web-дизайна Якоб Нильсен, "Пользователям нет никакого дела до технологий. Им нужно только, чтобы текст читался, контент отвечал на их вопросы, а навигация и поиск позволяли найти требуемую информацию".

Ну а мне остается пожелать вам успехов в нелегком, но очень интересном деле Web-строительства.

ПРИЛОЖЕНИЯ

Приложение 1

Краткий справочник по тегам HTML

Таблица П1.1. *Обозначения типов атрибутов*

Обозначение в табл. П1.2	**Возможные значения**
MIME-тип	MIME-тип файла для связанных или вложенных ресурсов (см. также табл. 15.2)
URL	Абсолютный или относительный адрес URL (см. также *разд. 1.4*)
дата-время	Дата и время в формате `ГГГГ-ММ-ДДТчч:мм:ссУВЗ`, где `ГГГГ` — 4-значный год, `ММ` — 2-значный номер месяца (`01`–`12`), `ДД` — двузначный номер дня (`01`–`31`), "`Т`" латинская прописная — разделитель, `чч` — двузначный номер часа (`00`–`23`), `мм` — двузначный номер минуты (`00`–`59`), `сс` — двузначный номер секунды (`00`–`59`), `УВЗ` — указатель временной зоны. Неизвестные компоненты даты и времени заменяются на `00`. Указатель временной зоны `Z` обозначает время по Гринвичу, `+чч:мм` — положительную разницу с Гринвичем в часах и минутах, `-чч:мм` — аналогичную отрицательную разницу
пикселы	Целые положительные числа, задающие количество пикселов (например, `800`)
размер	Целые положительные числа, задающие количество пикселов (например, `800`), либо доли в процентах от размера по горизонтали или вертикали (например, `75%`)
текст	Строка символов, возможно, включающая специальные символы. При отображении переносы строк игнорируются, символы перевода строки и табуляции заменяются на пробелы, начальные и конечные пробелы в значении атрибута игнорируются. Обычно значения зависят от регистра
типы ссылок	Список типов ссылок, разделенных пробелами. Тип ссылки не зависит от регистра и не может содержать пробелов (см. также *разд. 4.5*)

Таблица П1.1 *(окончание)*

Обозначение в табл. П1.2	**Возможные значения**
цвет	Шестнадцатеричное число с префиксом # вида #RRGGBB, задающее RGB-код цвета, либо одно из символических имен цветов. Не зависит от регистра символов (см. также *разд. 9.1*)
число	Десятичное число, состоящее хотя бы из одной арабской цифры
|	Выбор одного из нескольких возможных значений атрибута

Таблица П1.2. *Описание тегов*

Тег	**Описание и атрибуты**
<a>	Текстовый элемент. Его содержимое определяет положение якоря ссылки или закладки в документе. Применяется для создания закладок и гиперссылок. Может включать любые текстовые элементы, кроме <a>. Атрибуты: name="текст" (название закладки); href="URL" (URL адресуемой ссылки); rel="типы ссылок" (прямая ссылка); rev="типы ссылок" (обратная ссылка); target="фрейм" (фрейм для отображения ссылки или одно из значений "_blank", "_top", "_parent", "_self"). Примеры: внешняя ссылка, текущее окно внешняя ссылка, новое окно браузера внутренняя ссылка на файл из вложенной папки folder определение закладки с именем start ссылка на закладку с именем start
<abbr>	Текстовый элемент логического форматирования. Содержит маркировку аббревиатуры. Расшифровка задается атрибутом title="текст". Пример: Дистанция 100 <abbr title="метров">м</abbr>

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
<acronym>	Текстовый элемент логического форматирования. Содержит маркировку акронима. Расшифровка задается атрибутом title="текст". Пример: <acronym title="Организация Объединенных Наций">ООН</acronym >
<address>	Содержит контактную информацию о документе (имена автора, адрес E-mail и т. д.). Может содержать другие текстовые элементы. Пример: <address>Автор: Иванов В.А. E-mail: ivanov@mail.com </address>
<applet>	Блочный элемент. Описывает апплет на языке Java. Содержимым являются один или несколько элементов <param>, за которыми следуют блочные и/или текстовые элементы, отображаемые, если апплет не запускается. Атрибуты: code="текст" (имя файла класса); codebase="URL" (базовый URL для файлов класса); width="размер" (ширина окна апплета); height="размер" (высота окна апплета); archive="текст" (список библиотек классов); object="текст" (сериализованный апплет); name="текст" (имя апплета для взаимодействия с другими объектами); alt="текст" (альтернативный текст); align = "top" | "middle" | "bottom" | "left" | "right" (выравнивание окна апплета); hspace="пикселы" (горизонтальный отступ); vspace="пикселы" (вертикальный отступ). Пример: <applet code="slideways.class" width="700" height="100"> <param name="Notice" value="Applet by www.BrainCode.com"> <param name="Direction" value="1">\ <param name="LinkMode" value="0">\ <param name="Image1" value="01.jpg">\ <param name="Image2" value="02.jpg">\ </applet>

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
<area>	Не имеет содержимого и закрывающей части. Определяет активную область изображения, чувствительную к нажатию кнопки мыши. Атрибуты: shape="rect" | "circle" | "poly" | "default" (область в карте ссылок); coords="список координат в карте ссылок" (числа через запятую); href="URL" (URL ссылки); nohref (используется, если область неактивна); alt="текст" (альтернативный текст); target="фрейм" (фрейм для отображения ссылки). Пример: см. <map>
	Текстовый элемент физического форматирования. Отображает содержимое полужирным шрифтом. Пример: <p>Выделение полужирным шрифтом
<base>	Не имеет содержимого и закрывающей части. Помещается в элементе <head>. Задает в обязательном атрибуте href базовый URL для данного документа, используемый браузером для приведения относительных URL к полным. Атрибуты: href="URL" (базовый URL документа); target="фрейм" (имя фрейма для отображения ссылок). Пример: <base href="http://www.domain.com/folder/myurl.html"> Эта ссылка указывает на URL http://www.domain.com/images/picture.gif
<basefont>	Не имеет содержимого и закрывающей части. Изменяет параметры базового шрифта, влияя на внешний вид всех последующих за ним элементов, кроме заголовков. Обязательный атрибут size имеет по умолчанию значение 3. Атрибуты: size="число" (размер шрифта от 1 до 7); color="цвет" (цвет шрифта); face="текст" (наименование гарнитуры шрифта). Пример: <basefont size="3" color="#000033">

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
<bdo>	Текстовый элемент. Обязательным атрибутом dir определяет направление вывода текста. Может содержать любые текстовые элементы и содержаться в любых блочных и текстовых элементах. Атрибуты: dir="ltr" | "rtl" (направление вывода текста — слева направо (по умолчанию) или справа налево). Пример: <bdo dir="ltr">текст</bdo>
<bgsound>	Фоновый звук (только IE, Opera). Нестандартен, не рекомендуется. Пример: <bgsound src="sound/saintsgo.mid" loop="infinite">
<big>	Текстовый элемент физического форматирования. Отображает содержимое крупным шрифтом. Пример: Выделение <big>крупным</big> шрифтом
<blink>	Создает мерцающий текст. Нестандартен. Не поддерживается Internet Explorer. Не рекомендуется к использованию Пример: <blink>этот текст мерцает</blink>
<blockquote>	Блочный элемент. Применяется для маркировки блочных цитат, состоящих из нескольких строк. Отображается как блок текста, сдвинутый вправо. Может содержать блочные элементы, но не может быть частью абзаца или текстового элемента. Пример: Как заметил Станислав Лем, <blockquote>любая возможная информация содержится в компьютерных инвентарях благ, а ее недоступность обусловлена лишь ужасающей избыточностью накопленных сведений</blockquote>
<body>	Содержит HTML-элементы, предназначенные для отображения браузером. Задается в элементе <html> после элемента <head>. Атрибуты тега считаются устаревшими и должны заменяться на стили. Атрибуты: background="URL" (ссылка на файл фонового рисунка); bgcolor="цвет" (цвет фона страницы); text="цвет" (цвет основного текста на странице); link="цвет" (цвет непосещенных ссылок на странице); alink="цвет" (цвет активных ссылок на странице); vlink="цвет" (цвет посещенных ссылок на странице);

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
<body>	Атрибуты: topmargin="пикселы" (верхний отступ в окне браузера); leftmargin="пикселы" (левый отступ в окне браузера); marginheight="пикселы" (верхний и нижний отступы в окне браузера); marginwidth="пикселы" (левый и правый отступ в окне браузера). Пример: см. <html>
 	Не имеет содержимого и закрывающей части. Определяет разрыв строки. Может игнорироваться сразу за открывающей или непосредственно перед закрывающей частью тега. Пример: Строка этого абзаца будет разорвана
<button>	Текстовый элемент, создает кнопку формы. Может содержать любые текстовые элементы, кроме <a>, <button>, <iframe>, <input>, <label>, <select>, <textarea>, и любые блочные элементы, кроме <form> и <fieldset>. Может содержаться в любых текстовых и блочных элементах, кроме <button>. Атрибуты: type="submit" | "reset" | "button" (тип кнопки); name="текст" (имя кнопки); value="текст" (начальное значение кнопки); disabled (используется, если элемент запрещен). Пример: <button type="submit" name="submit" value="Отправить">Кнопка</button>
<caption>	Текстовый элемент. Определяет заголовок таблицы. Употребляется только как первый элемент внутри тега <table>. Может содержать только текстовые элементы. Расположение заголовка зависит от браузера. Атрибуты: align = "top" | "bottom" | "left" | "right" (горизонтальное выравнивание относительно таблицы). Пример: см. <table>
	Блочный элемент. Определяет блок, содержимое которого центрируется браузером по горизонтали. Пример: Это блок текста центрирован

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
`<cite>`	Текстовый элемент логического форматирования. Служит для выделения ссылок на названия книг, журналов, газет и других источников. По умолчанию отображается курсивом. Пример: `Газета <cite>Аргументы и факты</cite>`
`<code>`	Текстовый элемент логического форматирования. Указывает на программный код. По умолчанию отображается телетайпным (моноширинным) шрифтом. Пример: `Простейшая программа на Паскале имеет вид<code> begin end.</code>`
`<col>`	Не имеет содержимого и закрывающей части. Позволяет группировать атрибуты для столбцов таблицы, не создавая их групп. Должен располагаться после элемента `<caption>` и перед `<thead>`. Элементы `<col>` могут указываться как внутри `<colgroup>`, так и самостоятельно. Атрибуты: `span="число"` (число столбцов в группе); `width="размеры"` (список значений через запятую, задающих ширину каждого столбца); `align="left" | "center" | "right" | "justify"` (горизонтальное выравнивание содержимого); `valign="top" | "middle" | "bottom" | "baseline"` (вертикальное выравнивание содержимого). Пример: см. `<colgroup>`
`<colgroup>`	Определяет группу столбцов таблицы. Закрывающая часть не обязательна. Содержимым тега являются нуль или более тегов `<col>`. Должен располагаться после элемента `<caption>` и перед `<thead>`. Атрибуты: `span="число"` (число столбцов в группе); `width="размеры"` (список значений через запятую, задающих ширину каждого столбца); `align="left" | "center" | "right" | "justify"` (горизонтальное выравнивание содержимого); `valign="top" | "middle" | "bottom" | "baseline"` (вертикальное выравнивание содержимого).

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
<colgroup>	Пример: <table width="90%" align="center" border="1" cellspacing="0"> <colgroup span="6" width="30" align="center" valign="middle">> </colgroup> <col width="1*" align="center" valign="middle">> <col width="3*" align="right" valign="bottom">> <tr> <th>Пн</th><th>Вт</th><th>Ср</th><th>Чт</th> <th>Пт</th><th>Сб</th> <th>Вс</th> <td>Неделя</td> </tr> </table>
<dd>	Блочный элемент. Создает элемент списка определений <dl> (описание термина). Может содержать как текстовые, так и блочные элементы. Обычно определяемый тегом блок описания сдвигается браузером вправо. Пример: см. <dl>
<del>	Блочный или текстовый элемент. Означает удаление текста из документа. Способ отображения не определен. Атрибуты: cite="URL" (URL, где хранится объяснение внесения изменений); datetime="дата-время" (дата и время внесения изменений). Пример: <del cite="www.url.com" datetime="2007-12-31T15:30:00+03:00">Этот текст удален</del>
<dfn>	Текстовый элемент логического форматирования. Указывает на определение термина. По умолчанию отображается курсивом. Пример: Элемент разметки или оформления языка HTML называют <dfn>тегом</dfn>
<dir>	Устаревший элемент. Использовался вместо

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты

`<div>`	Блочный элемент. Предназначен для группировки элементов HTML-документа. Служит для задания стиля группы элементов.
	Атрибуты:
	`align = "left" | "center" | "right" | "justify"` (выравнивание текста раздела).
	Пример:
	`<div align="center">`
	`<p>абзац 1<p>абзац 2`
	`</div>`

`<dl>`	Блочный элемент. Содержит список определений. Содержимым является последовательность пар элементов `<dt>` и `<dd>`. Допускается давать несколько определений одному термину или сопоставлять одно определение нескольким терминам.
	Пример:
	`<dl>`
	`<dt><strong>Браузер</strong>`
	`<dt><strong>Web-обозреватель</strong>`
	`<dd>Программа для просмотра web-страниц`
	`<dt><strong>Internet Explorer</strong>`
	`<dd>Сокращенно IE, MSIE - бесплатный графический браузер компании Microsoft, входящий в состав всех последних версий Windows`
	`</dl>`

`<dt>`	Блочный элемент. Создает элемент списка определений `<dl>` (определение термина). Может содержать любые текстовые, но не блочные элементы. Обычно задаваемое тегом определение термина никак не выделяется визуально. Пример: см. `<dl>`
`<em>`	Текстовый элемент логического форматирования. Служит для смыслового выделения важных фрагментов текста. По умолчанию отображается курсивом. Пример: `Тег служит для <em>смыслового выделения</em> информации.`
`<embed>`	Блочный элемент, не имеет закрывающей части и содержимого. Описывает связанный с документом объект, является устаревающей альтернативой тегу `<object>`. Нестандартен, но широко поддерживается

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
<embed>	Атрибуты: code="текст" (имя файла, содержащего Java-апплет); src="URL" (URL загружаемого файла); width="размер" (ширина окна объекта); height="размер" (высота окна объекта); units="px" | "em" (единицы измерения высоты и ширины); name="текст" (имя для использования в сценариях); alt="текст" (альтернативный текст); align="top" | "middle" | "bottom" | "left" | "right" (способ выравнивания объекта); hspace="пикселы" (горизонтальный отступ от объекта до содержимого); vspace="пикселы" (вертикальный отступ от объекта до содержимого); pluginspage="URL" (URL загружаемого модуля для отображения объекта). Пример: <embed src="sound/rooster.wav" width="144" height="60" autostart="true" align="left" hspace="5" vspace="5">
<fieldset>	Блочный элемент. Определяет группу управляющих элементов формы. Содержимое — элемент <legend>, задающий заголовок группы элементов, за которым могут располагаться любые блочные и/или текстовые элементы, включая другой тег <fieldset>. Тег может содержаться в элементах <blockquote>, <body>, <dd>, <del>, , <fieldset>, <form>, <iframe>, <ins>, , <map>, <noframes>, <noscript>, <object>, <td>, <th>. Пример: см. <form>
	Блочный элемент. Задает параметры шрифта для отображения своего содержимого. Атрибуты: size="число" (размер шрифта от 1 до 7 включительно); color="цвет" (цвет шрифта); face="текст" (наименование гарнитуры шрифта). Пример: золотой цвет размера 5

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
`<form>`	Блочный элемент. Используется для создания интерактивных форм, служит контейнером для управляющих элементов, образующих форму. Кроме управляющих элементов, может содержать сценарии и любые блочные элементы, кроме `<form>`. Может содержаться в элементах `<blockquote>`, `<body>`, `<button>`, `<dd>`, `<del>`, `<div>`, `<fieldset>`, `<form>`, `<iframe>`, `<ins>`, `<li>`, `<map>`, `<noframes>`, `<noscript>`, `<object>`, `<td>`, `<th>`. Атрибуты: `action="URL"` (URL сценария-обработчика формы); `method="get" | "post"` (метод HTTP пересылки формы); `enctype="MIME-тип"` (тип файла для пересылки формы); `name="текст"` (название формы для сценариев); `target="фрейм"` (фрейм для отображения результатов). Пример: `<form name="form1" action="mailto:email@address.com" method="post" enctype="text/plain">` `<fieldset>` `<legend>Личная информация</legend>` `<label>Введите Ваше имя:` `<input type="text" name="name" maxlength="20" size="10" value="">` `</label>` `</fieldset>` `<fieldset>` `<legend>Пожалуйста, оцените мой сайт</legend>` `<label>Общая оценка:` `<select name="ball" size="1">` `<option value="5" selected>отлично!</option>` `<option value="4">хорошо для начала</option>` `<option value="3">нужно еще поработать</option>` `</select>` `</label>` `</fieldset>` ` <input type="submit" value="Отправить">` `</form>`

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
`<frame>`	Не имеет содержимого и закрывающей части. Определяет фрейм, т. е. независимое прямоугольное подокно внутри окна браузера. Содержится внутри элемента `<frameset>`, прямо или косвенно определяющего размеры окна. Атрибуты: `name="текст"` (имя фрейма для ссылки атрибутом `target`); `src="URL"` (URL целевого документа); `frameborder="1" | "0"` (наличие рамки) `marginwidth="пикселы"` (ширина полей вокруг содержимого); `marginheight="пикселы"` (высота полей вокруг содержимого); `noresize` (используется для запрета изменения размера фрейма); `scrolling="yes" | "no" | "auto"` (служит для управления полосами прокрутки фрейма). Пример: см. `<frameset>`

`<frameset>`	Блочный элемент. Описывает разбиение окна обозревателя на фреймы, употребляется вместо элемента `<body>` и располагается сразу после `<head>`. Состоит из одного или нескольких элементов `<frameset>` или `<frame>`, может содержать элемент `<noframes>`.
	Атрибуты:
	`rows="размеры"` (список размеров строк через запятую — пикселы, проценты или относительные размеры вида `i*`, где `i` — число);
	`cols="размеры"` (список размеров столбцов через запятую — пикселы, проценты или относительные размеры вида `i*`, где `i` — число).
	Пример:
	`<frameset rows="40,*" frameborder="1">`
	` <frame src="reclama.html" name="top" scrolling="no" no-resize>`
	` <frameset cols="128,*" frameborder="1">`
	`  <frame src="menu.html" name="menu" scrolling="auto">`
	`  <frame src="index.html" name="main" scrolling="yes">`
	` <noframes>`
	`  Извините, ваш браузер не поддерживает фреймы!`
	` </noframes>`
	`</frameset>`

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
<h1>,…, <h6>	Блочные элементы. Создают заголовки от первого до шестого уровня включительно. Все параметры отображения заголовков можно изменить через стили. Атрибуты: align="left" | "right" | "center" | "justify" (задает выравнивание заголовка). Пример: Заголовок Подзаголовок
<head>	Содержит информацию об общих свойствах документа. Задается в элементе <html> перед <body>. Не отображается браузерами. Может включать в себя элементы <title>, <meta>, <base>, <link>, <style>, <script>. Пример: см. <html>
<hr>	Не имеет содержимого и закрывающей части. Блочный элемент. Создает горизонтальную черту, обычно используемую в качестве разделителя. Атрибуты: size="пикселы" (толщина черты); width="размер" (длина черты); align="left" | "right" | "center" (горизонтальное выравнивание черты); color="цвет" (цвет черты); noshade (указывается при создании линии без трехмерных эффектов). Пример: <hr size="3" width="90%" align="center" color="red" noshade>
<html>	Указывает, что последующий документ является HTML-документом. Включает в себя все остальное содержимое документа. Пример: <html> <head> <meta name="Description" content="Изучение основных возможностей языка гипертекстовой разметки HTML"> <meta name="Keywords" content="HTML, тег, учебник"> <title>HTML. Самое необходимое</title> </head> <body bgcolor="#FFFFFF" text="#000000" link="#0000FF" alink="#FF0000" vlink="#800080"> <p>Содержимое документа </body> </html>

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
	Текстовый элемент физического форматирования. Отображает содержимое курсивом. Пример: Выделение курсивом
<iframe>	Текстовый элемент. Определяет вложенный фрейм. Может содержаться в текстовых и блочных элементах. Может содержать текстовые и блочные элементы, отображаемые только в случае, если браузер не поддерживает фреймы. Атрибуты: name="текст" (имя фрейма для ссылки атрибутом target); src="URL" (URL целевого документа); align= "top" | "middle" | "bottom" | "left" | "right" (выравнивание фрейма); width="размер" (ширина фрейма); height="размер" (высота фрейма); frameborder="1" | "0" (наличие рамки); marginwidth="пикселы" (ширина полей вокруг содержимого); marginheight="пикселы" (высота полей вокруг содержимого); scrolling="yes" | "no" | "auto" (используется для управления полосами прокрутки фрейма). Пример: <iframe name="links" src="links.html" frameborder="1" align="left" width="170" height="100"> Извините, Ваш браузер не поддерживает плавающие фреймы. Это окно адресует документ links.html </iframe>

<img>	Не имеет содержимого и закрывающей части. Указывает точку вставки в документ рисунка. Может содержаться в любых текстовых и блочных элементах, кроме <pre>.
	Атрибуты:
	src="URL" (URL файла, содержащего рисунок, обязателен);
	alt="текст" (альтернативный текст для отображения вместо графики);
	name="текст" (название рисунка для сценариев);
	width="размер" (ширина рисунка);
	height="размер" (высота рисунка);
	align = "top" | "middle" | "bottom" | "left" | "right" (выравнивание рисунка);

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты

`<img>`	Атрибуты *(окончание)*:
	`hspace="пикселы"` (горизонтальный отступ);
	`vspace="пикселы"` (вертикальный отступ);
	`border="пикселы"` (ширина рамки);
	`usemap="URL"` (URL клиентской карты ссылок).
	Примеры:
	`<img src="http://www.domain.com/folder/picture.gif">`
	`<img src="images/smile.gif" width="19" height="19" align="left">`
	`<img src="images/picture.jpg" width="600" height="400" align="center" hspace="4" vspace="4" alt="Рисунок">`

`<input>`	Не имеет содержимого и закрывающей части. Создает управляющий элемент формы, предназначенный для ввода данных пользователем. Может содержаться в любых блочных или текстовых элементах, кроме `<button>`. Многими браузерами отображается только внутри элемента `<form>`. Атрибуты: `type="text" | "password" | "checkbox" | "radio" | "submit" | "reset" | "file" | "hidden" | "image" | "button"` (задает тип элемента); `name="текст"` (имя элемента); `value="текст"` (начальное значение элемента для текста, возвращаемое — для флажков и переключателей, отображаемый текст — для кнопок); `checked` (состояние "выбран" для флажков и переключателей); `readonly` (состояние "только чтение" для текста и паролей); `disabled` (используется, если элемент запрещен); `size="число"` (видимая ширина поля для текста); `maxlength="число"` (максимальное количество вводимых символов для текста); `src="URL"` (URL графического образа для кнопки); `alt="текст"` (альтернативный текст для графической кнопки); `usemap="URL"` (URL клиентской карты ссылок для графической кнопки). Значения атрибута `type` имеют следующее назначение: `text` — однострочное поле ввода (принято по умолчанию); `password` — поле ввода пароля; `checkbox` — флажок; `radio` — переключатель;

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
<input>	Атрибуты *(окончание)*: reset — кнопка сброса формы; submit — кнопка пересылки формы; image — графическая кнопка пересылки; button — кнопка общего назначения; file — элемент выбора файлов. Форма, содержащая селектор файлов, должна иметь атрибуты method="post" и enctype="multipart/form-data"; hidden — скрытый элемент, служащий для передачи скрипту данных вида name="value". Пример: <form method="post" action="mailto:email@domain.com> <p>Ваше имя: <input name="name" type="text" maxlength="40" size="40" value="">> <p>Ваш пол: <input type="radio" name="pol" value="male">Мужской <input type="radio" name="pol" value="female">Женский <p>Какие языки программирования Вы знаете? <input type="checkbox" name="pascal" value="yes" checked>Паскаль <input type="checkbox" name="c" value="yes">Си <p>Напишите несколько слов о себе: <textarea name="about" rows="6" cols="60"></textarea> <p><input type="submit" value="Отправить">> <input type="reset" value="Очистить">> </form>
<ins>	Блочный или текстовый элемент. Означает вставку нового текста в документ. Способ отображения не определен. Атрибуты: cite="URL" (URL, где хранится объяснение внесения изменений); datetime="дата-время" (дата и время внесения изменений). Пример: <ins cite="www.url.com" datetime="2007-12-31T15:30:00+03:00">Этот текст вставлен</ins>
<isindex>	Устаревший элемент. Использовался вместо <input type="text">.
<kbd>	Текстовый элемент логического форматирования. Указывает на текст, вводимый пользователем с клавиатуры. По умолчанию отображается телетайпным (моноширинным) шрифтом. Пример: Введите команду <kbd>dir</kbd>

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
`<label>`	Текстовый элемент. Создает текстовую метку для другого элемента формы. Содержимым могут быть любые текстовые элементы, кроме `<label>`. Может содержаться в любых блочных и текстовых элементах, кроме `<button>`. Связывается с другим элементом формы атрибутом `for="id элемента"` или размещением соответствующего элемента в качестве содержимого тега. Пример: см. `<form>`
`<legend>`	Текстовый элемент. Задает заголовок группы элементов. Может содержать любые текстовые элементы и употребляться только в начале элемента `<fieldset>`. Пример: см. `<form>`
`<li>`	Блочный элемент. Определяет один элемент списка в теге `<ul>` или `<ol>`. Закрывающая часть необязательна. Как правило, браузер автоматически переводит строку перед каждым элементом списка. Атрибуты: `type = "disc" | "square" | "circle" | "1" | "a" | "A" | "i" | "I"` (определяет тип маркера элемента); `value="число"` (изменяет нумерацию, начиная с этого элемента). Примеры: см. `<ul>`, `<ol>`

`<link>`	Не имеет содержимого и закрывающей части. Определяет взаимосвязь между документами. Помещается в элементе `<head>`.
	Атрибуты:
	`rel="типы ссылок"` (прямая ссылка);
	`rev="типы ссылок"` (обратная ссылка);
	`href="URL"` (URL ссылки);
	`target="фрейм"` (имя фрейма для отображения ссылки).
	Примеры:
	`<link rel="stylesheet" type="text/css" href= "style.css">`
	`<link href="mailto:webmaster@domain.com" rev=made">`

`<listing>`	Устаревший элемент. Использовался вместо `<pre>`

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
<map>	Создает графическую клиентскую карту ссылок. Содержимым могут быть либо элементы <area>, либо любые блочные элементы. Употребляется совместно с элементами <img>, <object> или <input>, загружающими графический образ. Обязательный атрибут name="текст" содержит название карты, используемое в атрибуте usemap тега графического образа. Пример: <map name="map1"> <area shape="circle" coords="50,50,50" href="1.html">\ <area shape="poly" coords="50,200,200,264,400,0,50,200" href="2.html">\ <area shape="rect" coords="0,270,399,299" href="3.html">\ </map> <img src="img/html_map.gif" width="400" height="300" border="1" usemap="#map1">
<marquee>	Создает бегущую строку текста. Нестандартен, поддерживается не всеми браузерами, не рекомендуется. Атрибут behavior со значением "scroll", "slide" или "alternate" управляет поведением строки. Пример: <marquee behavior="alternate" bgcolor="green" direction="right" loop="infinite" width="90%" height="50" hspace="5" vspace="5">\ Бегущая строка текста</marquee>
<menu>	Устаревший элемент. Использовался вместо .
<meta>	Не имеет содержимого и закрывающей части. Обычно размещается в элементе <head>. Содержит мета-описатели общих свойств документа в виде <meta name="свойство" content="значение">> или <meta http-equiv="Заголовок сообщения HTTP" content="значение">>. Примеры: <meta name="Author" content="А.С. Пушкин" lang="ru">\ <meta name="Description" content="Изучение основных возможностей языка гипертекстовой разметки HTML">\ <meta http-equiv="Content-Type" content="text/html; charset=windows-1251">\ <meta http-equiv="Expires" content="Sun, 6 Jan 2008 09:00:00 GMT">\ <meta http-equiv="Refresh" content="10;URL=http://www.domain.com/new.html">\

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
<nobr>	Текстовый элемент. Указывает, что содержащийся в нем текст должен отображаться без разрывов строк. Нестандартен, но широко поддерживается. Пример: <nobr>Этот текст отображается без переноса на новую строку в окне браузера</nobr>
<noframes>	Блочный элемент. Задает альтернативное содержание фреймового документа для браузеров, не поддерживающих фреймы. Располагается в конце тела документа перед закрывающей частью </frameset>. Может содержать любые элементы, допустимые внутри тега <body>. Пример: см. <frameset>
<noscript>	Блочный элемент. Обычно указывается после закрывающей части тега </script>. Определяет альтернативное содержимое для случаев, когда обозреватель не может исполнить ранее указанный сценарий. Может содержать любые элементы разметки содержимого. Пример: см. <script>

<object>	Блочный элемент. Обеспечивает включение в документ объектов различных типов. Может содержаться в заголовке документа и любых текстовых или блочных элементах, кроме <pre>. Содержимым тега являются элементы <param>, за которыми могут следовать текстовые или блочные теги. Все теги, следующие за <param>, задают альтернативное содержимое, отображаемое, если браузер не поддерживает указанный тип объектов.
	Атрибуты:
	classid="URL" (URL кода объекта);
	data="URL" (URL данных объекта);
	archive="текст" (список архивных файлов с кодом объекта);
	codebase="URL" (базовый URL для атрибутов classid, data, archive);
	width="размер" (ширина окна объекта);
	height="размер" (высота окна объекта);
	align = "top" | "middle" | "bottom" | "left" | "right" (выравнивание объекта);
	hspace="пикселы" (горизонтальный отступ);
	vspace="пикселы" (вертикальный отступ);
	border="пикселы" (ширина рамки);
	usemap="URL" (URL клиентской карты ссылок);
	name="текст" (название объекта для форм и сценариев);
	codetype="MIME-тип" (тип кода объекта);
	type="MIME-тип" (тип данных объекта).

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
<object>	Примеры: <object type="image/jpeg" data="images/photo1.jpg" width="230" height="240" >Мое фото</object> <object classid="clsid:978C9E23-D4B0-11CE-BF2D-00AA003F40D0" id="label1" width="500" height="100" align="middle">К сожалению, ваш браузер не поддерживает ActiveX</object>
	Блочный элемент. Определяет нумерованный список. Содержимое — один или более элементов . Элементы списка обычно отображаются с отступом вправо и с номером перед началом элемента. Атрибуты: type="1"|"I"|"i"|"A"|"a" (задает способ нумерации элементов списка); start="число" (определяет начало нумерации элементов списка). Пример: первый элемент; второй элемент
<optgroup>	Текстовый элемент. Задает группу элементов списка. Используется внутри элемента <select>. Может содержать только элементы <option>. Атрибуты: label="текст" (метка группы); disabled (используется, если элемент запрещен). Пример: <select name="news" size="5" multiple> <optgroup label="Технические новости"> <option value="server" selected>Новости сервера <option value="soft">Програмнное обеспечение <option value="mobile">Мобильные компьютеры </optgroup> <optgroup label="Прочее">> <option value="weather" selected>Прогноз погоды <option value="tv">Программа ТВ </optgroup> </select>

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
`<option>`	Текстовый элемент. Описывает один элемент списка. Размещается внутри элемента `<select>`. Может содержать только обычный текст, включая специальные символы. Содержимое тега отображается браузером как элемент списка. Атрибуты: `value="текст"` (имя элемента); `label="текст"` (метка элемента); `selected` (используется, если элемент выбран изначально); `disabled` (указывается, если элемент запрещен). Примеры: см. `<form>`, `<optgroup>`

`<p>`	Блочный элемент. Определяет абзац текста. Обычно абзац отделяется браузерами пустой строкой.
	Атрибуты:
	`align="left" | "right" | "center" | "justify"` (задает выравнивание абзаца)
	Пример:
	`<p align="right">11 января 2008 г.`
	`<p align="justify">Текст документа</p>`

`<param>`	Не имеет содержимого и закрывающей части. Один или несколько тегов `<param>` могут находиться только в начале тегов `<object>` и `<applet>` для задания начальных значений параметров объекта в виде `name="value"`. Атрибуты: `name="текст"` (название параметра); `value="текст"` (значение параметра). Пример: см. `<applet>`
`<plaintext>`	Устаревший элемент. Использовался вместо `<pre>`.

`<pre>`	Блочный элемент. Содержит предварительно отформатированный текст, выводимый шрифтом фиксированной ширины без сжатия пробелов и автоматического переноса слов на следующую строку. Применяется для вывода текстов, в которых размещение слов в строках имеет существенное значение.
	Пример:
	`<p>Текст простейшей программы на Си имеет следующий вид:`
	`<pre>`
	`#include <stdio.h>`
	` void main () {`
	`  printf ("\nHello, world!");`
	` }`
	`</pre>`

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
<q>	Текстовый элемент логического форматирования. Служит для выделения коротких цитат в основном тексте. Атрибуты: cite="URL" (URL, где хранится источник цитаты). Пример: Колин Уилсон писал: <q>Самообман - это делать вид, что свобода есть необходимость</q>
<s>	Текстовый элемент физического форматирования. Отображает содержимое перечеркнутым шрифтом. Пример: В тексте есть <s>перечеркнутое</s> слово
<samp>	Текстовый элемент логического форматирования. Указывает на вывод результатов работы программы. По умолчанию отображается телетайпным (моноширинным) шрифтом. Пример: При обнаружении ошибка программа выведет сообщение <samp>Bad command or file name</samp>

<script>	Определяет сценарий, выполняемый на стороне клиента. Тег может содержаться в элементе <head>, а также в любых текстовых и блочных элементах внутри тега <body>. Содержимым тега может быть сценарий на языке программирования, заданном атрибутом type (обычно type="text/javascript").
	Атрибуты:
	type="MIME-тип" (MIME-тип языка сценариев);
	language="текст" (язык сценариев, обычно JavaScript);
	src="URL" (URL для файла с внешним сценарием);
	defer (только декларация).
	Примеры:
	<script type="text/javascript" src="scripts/script1.js"></script>
	<script language="JavaScript">function showCY(domain) {
	if (domain==''') {
	 window.alert ('Вы не указали адрес сайта в поле ввода!'); }
	else {
	 if (domain.indexOf ('http://')==-1) { domain='http://'+domain; }
	window.open('http://www.yandex.ru/yandsearch?Link='+domain, '_blank'); }

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты

`<script>`	Примеры *(окончание)*:
	`}`
	`</script>`
	`<noscript>`
	`<p>Извините, Ваш браузер не поддерживает JavaScript.`
	`<p>Программа недоступна.`
	`</noscript>`

`<select>`	Текстовый элемент. Создает интерфейсный элемент формы, предназначенный для выбора значений из списка. Отображается в виде выпадающего меню (при `size="1"`) или многострочного списка. Содержимым тега могут быть только теги `<optgroup>` и `<option>`, определяющие элементы списка. Тег может содержаться в любых блочных и текстовых элементах, кроме `<button>`. Атрибуты: `name="текст"` (имя списка); `size="число"` (число видимых элементов списка); `multiple` (разрешает множественный выбор); `disabled` (указывается, если элемент запрещен). Примеры: см. `<form>`, `<optgroup>`
`<small>`	Текстовый элемент физического форматирования. Отображает содержимое мелким шрифтом. Пример: `<p>Выделение <big>мелким</big> шрифтом`
`<span>`	Текстовый элемент. Предназначен для группировки элементов HTML-документа. Используется для задания стиля группы элементов. Пример: `<span lang="en">Windows</span> означает "окна"`
`<strike>`	То же, что `<s>`
`<strong>`	Текстовый элемент логического форматирования. Служит для сильного смыслового выделения важных фрагментов текста. По умолчанию отображается полужирным шрифтом. Пример: `Тег <strong> — это <strong>более сильное</strong> выделение.`

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты

`<style>`	Включает в документ внутренние таблицы стилей. Обычно размещается в теге `<head>`. Обязательный атрибут `type` указывает на язык описания стилей (в настоящее время всегда задается `type="text/css"`).
	Атрибуты:
	`type="MIME-тип"` (тип таблицы стилей).
	Пример:
	`<style type="text/css">`
	`P {`
	`  font-family: serif, sans-serif;`
	`  color: #000000;`
	`  font-size: 12pt;`
	`  text-align: justify;`
	`  }`
	`</style>`

`<sub>`	Текстовый элемент физического форматирования. Отображает содержимое подстрочным шрифтом. Пример: `Формула воды — H<sub>2</sub>O`
`<sup>`	Текстовый элемент физического форматирования. Отображает содержимое надстрочным шрифтом. Пример: `My 1<sup>st</sup> HTML document`
`<table>`	Блочный элемент, определяющий таблицу. Может содержать теги заголовка таблицы (`<caption>`), группирования столбцов (`<colgroup>`, `<col>`), выделения строковых секций (`<thead>`, `<tfoot>`, `<tbody>`), отдельных строк (`<tr>`) и ячеек (`<th>`, `<td>`). Ячейки таблиц `<td>` и `<th>` могут содержать текст, графические образы, гиперссылки, формы, другие таблицы. Атрибуты: `width="размер"` (ширина таблицы); `border="пикселы"` (ширина рамки); `frame="void" | "above" | "below" | "hsides" | "lhs" | "rhs" | "vsides" | "box" | "border"` (вид внешних рамок); `rules="none" | "groups" | "rows" | "cols" | "all"` (вид внутренних рамок); `cellspacing="размер"` (расстояние между ячейками); `cellpadding="размер"` (отступ от краев ячеек до содержимого); `align="left" | "right" | "center"` (горизонтальное выравнивание таблицы); `bgcolor="цвет"` (цвет фона элемента).

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты

`<table>`	Пример:
	`<table align="center" border="1" cellpadding="4" cellspacing="0">`
	`<caption>Заголовок таблицы</caption>`
	`<tr bgcolor="yellow" align="center">`
	`<th>Строка заголовка, столбец 1</th>`
	`<th>Строка заголовка, столбец 2</th>`
	`</tr>`
	`<tr>`
	`<td align="center">Строка данных, столбец 1</td>`
	`<td align="center">Строка данных, столбец 2</td>`
	`</tr>`
	`</table>`

`<tbody>`	Блочный элемент. Определяет группу строк тела таблицы. Содержимое элемента — один или несколько тегов `<tr>`.
	Атрибуты:
	`align="left" | "center" | "right" | "justify"` (горизонтальное выравнивание содержимого);
	`valign="top" | "middle" | "bottom" | "baseline"` (вертикальное выравнивание содержимого).
	Пример:
	`<table border="1">`
	`<thead><tr>`
	`<th>Заголовок 1</th>`
	`<th>Заголовок 2</th>`
	`</tr></thead>`
	`<tfoot><tr>`
	`<th>Итог 1</th>`
	`<th>Итог 2</th>`
	`</tr></tfoot>`
	`<tbody><tr>`
	`<td>Блок 1, строка 1, данные 1</td>`
	`<td>Блок 1, строка 1, данные 2</td>`
	`</tr><tr>`
	`<td>Блок 1, строка 2, данные 1</td>`
	`<td>Блок 1, строка 2, данные 2</td>`
	`</tr></tbody>`
	`<tbody><tr>`
	`<td>Блок 2, данные 1</td>`
	`<td>Блок 2, данные 2</td>`
	`</tr></tbody>`
	`</table>`

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
<td>	Блочный элемент. Определяет одну ячейку данных таблицы. Содержимым тега могут быть как блочные, так и текстовые элементы. Располагается внутри тега <tr>. Атрибуты: rowspan = число (число строк таблицы, занятых ячейкой); colspan = число (число столбцов таблицы, занятых ячейкой); align="left" | "center" | "right" | "justify" (горизонтальное выравнивание содержимого); valign="top" | "middle" | "bottom" | "baseline" (вертикальное выравнивание содержимого); width="размер" (ширина элемента); height="размер" (высота элемента); bgcolor="цвет" (цвет фона элемента); nowrap (если указано — не переносить содержимое на новую строку). Пример: см. <table>
<textarea>	Текстовый элемент. Создает многострочное поле ввода данных. Содержимым может быть обычный текст, включая специальные символы. Этот текст становится начальным отображаемым в поле значением. Тег может содержаться в любых блочных и текстовых элементах, кроме <button>. Атрибуты: name="текст" (имя поля ввода); rows="число" (рекомендуемое число видимых строк); cols="число" (рекомендуемое число видимых столбцов); readonly (используется, если элемент предназначен только для чтения); disabled (указывается, если элемент запрещен). Пример: см. <input>
<tfoot>	Блочный элемент. Определяет группу строк подзаголовка таблицы. Содержимое элемента — один или несколько тегов <tr>. Поддерживается рядом браузеров некорректно. Атрибуты: align="left" | "center" | "right" | "justify" (горизонтальное выравнивание содержимого); valign="top" | "middle" | "bottom" | "baseline" (вертикальное выравнивание содержимого). Пример: см. <tbody>

Таблица П1.2 *(продолжение)*

Тег	Описание и атрибуты
`<th>`	Блочный элемент. Определяет одну заголовочную ячейку таблицы. Содержимым тега могут быть как блочные, так и текстовые элементы. Располагается внутри тега `<tr>`. Содержимое может по умолчанию выводиться полужирным шрифтом. Атрибуты: `rowspan="число"` (число строк таблицы, занятых ячейкой); `colspan="число"` (число столбцов таблицы, занятых ячейкой); `align="left" | "center" | "right" | "justify"` (горизонтальное выравнивание содержимого); `valign="top" | "middle" | "bottom" | "baseline"` (вертикальное выравнивание содержимого); `width="размер"` (ширина элемента); `height="размер"` (высота элемента); `bgcolor="цвет"` (цвет фона элемента); `nowrap` (если указано — не переносить содержимое на новую строку). Пример: см. `<table>`
`<thead>`	Блочный элемент. Определяет группу строк надзаголовка таблицы. Содержимое элемента — один или несколько тегов `<tr>`. Поддерживается рядом браузеров некорректно. Атрибуты: `align="left" | "center" | "right" | "justify"` (горизонтальное выравнивание содержимого); `valign="top" | "middle" | "bottom" | "baseline"` (вертикальное выравнивание содержимого). Пример: см. `<tbody>`
`<title>`	Содержит текст, отображаемый в строке заголовка окна браузера. Размещается в элементе `<head>`. Текст заголовка может содержать специальные символы, но не другие элементы. Пример: см. `<html>`
`<tr>`	Блочный элемент. Определяет одну строку таблицы. Содержимое — один или несколько тегов `<th>` и `<td>`. Атрибуты: `align="left" | "center" | "right" | "justify"` (горизонтальное выравнивание содержимого); `valign="top" | "middle" | "bottom" | "baseline"` (вертикальное выравнивание содержимого); `bgcolor="цвет"` (цвет фона элемента). Пример: см. `<table>`

Таблица П1.2 *(окончание)*

Тег	Описание и атрибуты
<tt>	Текстовый элемент физического форматирования. Отображает содержимое телетайпным (моноширинным) шрифтом. Пример: Выделение <tt>телетайпным шрифтом</tt>
<u>	Текстовый элемент физического форматирования. Отображает содержимое подчеркнутым шрифтом. Пример: В тексте есть <s>подчеркнутое</s> слово
	Блочный элемент. Определяет маркированный список. Содержимым являются один или более элементов . Элементы списка обычно отображаются с отступом вправо и с маркером перед началом элемента. Атрибуты: type="disc" | "circle" | "square" (задает тип маркера списка). Пример: Список браузеров Internet Explorer Opera Mozilla Firefox
<var>	Текстовый элемент логического форматирования. Применяется для выделения имен переменных и аргументов программ. По умолчанию отображается курсивом. Пример: Определим переменные <var>X</var> и <var>Y</var>
<wbr>	Не имеет конечного тега и содержимого. Употребляется только внутри элемента <nobr>, указывая, что в этом месте возможен разрыв строки. Нестандартен, но широко поддерживается. Пример: <nobr>Неразрывную строку текста можно разорвать <wbr>в этом месте</nobr>
<xmp>	Устаревший элемент. Использовался вместо <pre>

Приложение 2

Основные специальные символы HTML

Таблица П2.1. *Специальные символы общего назначения*

Описание символа	HTML-код	Вид
Двойная кавычка	"	"
Амперсанд	&	&
Левая угловая скобка	<	<
Правая угловая скобка	>	>
Короткий пробел		
Длинный пробел		
Узкий пробел		
Неразрывный пробел		
Разделитель нулевой ширины	‌	
Соединитель нулевой ширины	‍	
Указатель слева направо	‎	
Указатель справа налево	‏	
Короткое тире (дефис)	–	–
Длинное тире	—	—
Открывающая одинарная кавычка	‘	‘
Закрывающая одинарная кавычка	’	’
Нижняя одинарная кавычка	‚	‚
Открывающая двойная кавычка	“	“
Закрывающая двойная кавычка	”	”
Нижняя двойная кавычка	„	„
Открывающая угловая кавычка	‹	‹
Закрывающая угловая кавычка	›	›
Мягкий перенос		

Таблица П2.2. *Специальные символы для оформления документа*

Описание символа	Обозначение	Вид
Лигатура OE	Œ	Œ
Лигатура oe	œ	œ
Циркумфлекс	ˆ	ˆ
Малая тильда	˜	˜
Кинжал	†	†
Двойной кинжал	‡	‡
Знак промилле	‰	‰
Перевернутый восклицательный знак	¡	¡
Цент	¢	¢
Фунт стерлингов	£	£
Знак денежной единицы	¤	¤
Иена	¥	¥
Евро	€	€
Вертикальная черта	¦	¦
Параграф	§	§
Знак авторского права	©	©
Показатель женского рода	ª	ª
Открывающая двойная угловая кавычка	«	«
Знак отрицания	¬	¬
Охраняемый знак	®	®
Надчеркивание	¯	¯
Градус	°	°
Плюс-минус	±	±
Вторая степень	²	²
Третья степень	³	³
Острое ударение	´	´
Знак микро	µ	µ
Конец абзаца	¶	¶
Средняя точка	·	·
Единица в верхнем индексе	¹	¹

Таблица П2.2 *(окончание)*

Описание символа	Обозначение	Вид
Показатель мужского рода	`º`	º
Закрывающая двойная угловая кавычка	`»`	»
Одна четвертая	`¼`	¼
Одна вторая	`½`	½
Три четверти	`¾`	¾
Перевернутый вопросительный знак	`¿`	¿
Знак умножения	`×`	×
Знак деления	`÷`	÷

Таблица П2.3. *Математические символы и греческие буквы*

Описание символа	Обозначение	Вид
Знак функции	`ƒ`	ƒ
Прописная альфа	`Α`	Α
Прописная бета	`Β`	Β
Прописная гамма	`Γ`	Γ
Прописная дельта	`Δ`	Δ
Прописной эпсилон	`Ε`	Ε
Прописная дзета	`Ζ`	Ζ
Прописная эта	`Η`	Η
Прописная тета	`Θ`	Θ
Прописная иота	`Ι`	Ι
Прописная каппа	`Κ`	Κ
Прописная ламбда	`Λ`	Λ
Прописная мю	`Μ`	Μ
Прописная ню	`Ν`	Ν
Прописная кси	`Ξ`	Ξ
Прописной омикрон	`Ο`	Ο
Прописная пи	`Π`	Π
Прописная ро	`Ρ`	Ρ

Таблица П2.3 *(продолжение)*

Описание символа	Обозначение	Вид
Прописная сигма	Σ	Σ
Прописная тау	Τ	Τ
Прописная ипсилон	Υ	Υ
Прописная фи	Φ	Φ
Прописная хи	Χ	Χ
Прописная пси	Ψ	Ψ
Прописная омега	Ω	Ω
Строчная альфа	α	α
Строчная бета	β	β
Строчная гамма	γ	γ
Строчная дельта	δ	δ
Строчная эпсилон	ε	ε
Строчная дзета	ζ	ζ
Строчная эта	η	η
Строчная тета	θ	θ
Строчная иота	ι	ι
Строчная каппа	κ	κ
Строчная ламбда	λ	λ
Строчная мю	μ	μ
Строчная ню	ν	ν
Строчная кси	ξ	ξ
Строчный омикрон	ο	ο
Строчная пи	π	π
Строчная ро	ρ	ρ
Строчная сигма конечная	ς	ς
Строчная сигма	σ	σ
Строчная тау	τ	τ
Строчная ипсилон	υ	υ
Строчная фи	φ	φ
Строчная хи	χ	χ

Таблица П2.3 *(продолжение)*

Описание символа	Обозначение	Вид
Строчная пси	ψ	ψ
Строчная омега	ω	ω
Символ строчная тета	ϑ	ϑ
Ипсилон с крючком	ϒ	ϒ
Символ пи	ϖ	ϖ
Маркер списка	•	•
Многоточие	…	…
Знак прим	′	′
Знак двойной прим	″	″
Надчеркивание	‾	‾
Дробная черта	⁄	⁄
Рукописная Р	℘	℘
Мнимая часть числа	ℑ	ℑ
Действительная часть числа	ℜ	ℜ
Торговая марка	™	™
Алеф	ℵ	ℵ
Стрелка влево	←	←
Стрелка вверх	↑	↑
Стрелка вправо	→	→
Стрелка вниз	↓	↓
Стрелка влево-вправо	↔	↔
Возврат каретки	↵	↵
Двойная стрелка влево	⇐	⇐
Двойная стрелка вверх	⇑	⇑
Двойная стрелка вправо	⇒	⇒
Двойная стрелка вниз	⇓	⇓
Двойная стрелка влево-вправо	⇔	⇔
Квантор всеобщности	∀	∀
Знак дифференциала	∂	∂
Квантор существования	∃	∃

Таблица П2.3 *(продолжение)*

Описание символа	**Обозначение**	**Вид**
Пустое множество	∅	∅
Набла	∇	∇
Принадлежит множеству	∈	∈
Не принадлежит множеству	∉	∉
Является членом	∋	∋
N-арное произведение	∏	∏
N-арная сумма	∑	∑
Знак минус	−	−
Оператор звездочка	∗	∗
Радикал	√	√
Пропорционально	∝	∝
Бесконечность	∞	∞
Угол	∠	∠
Логическое И	∧	∧
Логическое ИЛИ	∨	∨
Пересечение	∩	∩
Объединение	∪	∪
Интеграл	∫	∫
Следовательно	∴	∴
Оператор тильда	∼	∼
Приблизительно равно	≅	≅
Асимптотически равно	≈	≈
Не равно	≠	≠
Тождественно равно	≡	≡
Меньше или равно	≤	≤
Больше или равно	≥	≥
Подмножество	⊂	⊂
Надмножество	⊃	⊃
Не подмножество	⊄	⊄
Подмножество или равно	⊆	⊆

Таблица П2.3 *(окончание)*

Описание символа	**Обозначение**	**Вид**
Надмножество или равно	`⊇`	⊇
Прямая сумма	`⊕`	⊕
Векторное произведение	`⊗`	⊗
Перпендикулярно	`⊥`	⊥
Оператор точка	`⋅`	⋅
Левый верхний угол	`⌈`	⌈
Правый верхний угол	`⌉`	⌉
Левый нижний угол	`⌊`	⌊
Правый нижний угол	`⌋`	⌋
Левая угловая скобка	`⟨`	〈
Правая угловая скобка	`⟩`	〉
Ромб	`◊`	◊
Пики	`♠`	♠
Трефы	`♣`	♣
Червы	`♥`	♥
Бубны	`♦`	♦

Приложение 3

Названия и коды цветов в HTML

Далее приведены допустимые значения атрибута `color` в HTML. Для каждого цвета указаны его наименование, шестнадцатеричный RGB-код и десятичное представление, удобное для работы в графических редакторах. Пометка (!) после названия цвета означает, что наименование стандартно и упомянуто в спецификации HTML. Символы (*) показывают, что поддержка цвета есть лишь в некоторых браузерах, и применять его не следует. В общем случае рекомендуется предпочитать наименованиям цветов их шестнадцатеричные RGB-коды.

Aliceblue (#F0F8FF) (240, 248, 255)	Antiquewhite (#FAEBD7) (250, 235, 215)	Aqua (!) (#00FFFF) (0, 255, 255)	Aquamarine (#7FFFD4) (127, 255, 212)
Azure (#F0FFFF) (240, 255, 255)	Beige (#F5F5DC) (245, 245, 220)	Bisque (#FFE4C4) (255, 228, 196)	Black (!) (#000000) (0, 0, 0)
Blanchedalmond (#FFEBCD) (255, 235, 205)	Blue (!) (#0000FF) (0, 0, 255)	Blueviolet (#8A2BE2) (138, 43, 226)	Brown (#A52A2A) (165, 42, 42)
Burlywood (#DEB887) (222, 184, 135)	Cadetblue (#5F9EA0) (95, 158, 160)	Chartreuse (#7FFF00) (127, 255, 0)	Chocolate (#D2691E) (210, 105, 30)
Coral (#FF7F50) (255, 127, 80)	Cornflowerblue (#6495ED) (100, 149, 237)	Cornsilk (#FFF8DC) (255, 248, 220)	Crimson (#DC143C) (220, 20, 60)
Cyan (#00FFFF) (0, 255, 255)	Darkblue (#00008B) (0, 0, 139)	Darkcyan (#008B8B) (0, 139, 139)	Darkgoldenrod (#B8860B) (184, 134, 11)
Darkgray (#A9A9A9) (169, 169, 169)	Darkgreen (#006400) (0, 100, 0)	Darkkhaki (#BDB76B) (189, 183, 107)	Darkmagenta (#8B008B) (139, 0, 139)

Darkolivegreen
(#556B2F)
(85, 107, 47)

Darksalmon
(#E9967A)
(233, 150, 122)

Darkturquoise
(#00CED1)
(0, 206, 209)

Dimgray
(#696969)
(105, 105, 105)

Forestgreen
(#228B22)
(34, 139, 34)

Gold
(#FFD700)
(255, 215, 0)

Greenyellow
(#ADFF2F)
(173, 255, 47)

Indigo
(#4B0082)
(75, 0, 130)

Lavenderblush
(#FFF0F5)
(255, 240, 245)

Lightcoral
(#F08080)
(240, 128, 128)

Lightgray (*)
(#D3D3D3)
(211, 211, 211)

Lightseagreen
(#20B2AA)
(32, 178, 170)

Lightsteelblue
(#B0C4DE)
(176, 196, 222)

Linen
(#FAF0E6)
(250, 240, 230)

Mediumblue
(#0000CD)
(0, 0, 205)

Darkorange
(#FF8C00)
(255, 140, 0)

Darkseagreen
(#8FBC8B)
(143, 188, 143)

Darkviolet
(#9400D3)
(148, 0, 211)

Dodgerblue
(#1E90FF)
(30, 144, 255)

Fuchsia (!)
(#FF00FF)
(255, 0, 255)

Goldenrod
(#DAA520)
(218, 165, 32)

Honeydew
(#F0FFF0)
(240, 255, 240)

Ivory
(#FFFFF0)
(255, 255, 240)

Lawngreen
(#7CFC00)
(124, 252, 0)

Lightcyan
(#E0FFFF)
(224, 255, 255)

Lightgreen
(#90EE90)
(144, 238, 144)

Lightskyblue
(#87CEFA)
(135, 206, 250)

Lightyellow
(#FFFFE0)
(255, 255, 224)

Magenta
(#FF00FF)
(255, 0, 255)

Mediumorchid
(#BA55D3)
(186, 85, 211)

Darkorchid
(#9932CC)
(153, 50, 204)

Darkslateblue
(#483D8B)
(72, 61, 139)

Deeppink
(#FF1493)
(255, 20, 147)

Firebrick
(#B22222)
(178, 34, 34)

Gainsboro
(#DCDCDC)
(220, 220, 220)

Gray (!)
(#808080)
(128, 128, 128)

Hotpink
(#FF69B4)
(255, 105, 180)

Khaki
(#F0E68C)
(240, 230, 140)

Lemonchiffon
(#FFFACD)
(255, 250, 205)

Lightgoldenrod (*)
(#EEDD82)
(238, 221, 130)

Lightpink
(#FFB6C1)
(255, 182, 193)

Lightslateblue (*)
(#8470FF)
(132, 112, 255)

Lime (!)
(#00FF00)
(0, 255, 0)

Maroon (!)
(#800000)
(128, 0, 0)

Mediumpurple
(#9370DB)
(147, 112, 219)

Darkred
(#8B0000)
(139, 0, 0)

Darkslategray
(#2F4F4F)
(47, 79, 79)

Deepskyblue
(#00BFFF)
(0, 191, 255)

Floralwhite
(#FFFAF0)
(255, 250, 240)

Ghostwhite
(#F8F8FF)
(248, 248, 255)

Green (!)
(#008000)
(0, 128, 0)

Indianred
(#CD5C5C)
(205, 92, 92)

Lavender
(#E6E6FA)
(230, 230, 250)

Lightblue
(#ADD8E6)
(173, 216, 230)

Lightgoldenrodyellow
(#FAFAD2)
(250, 250, 210)

Lightsalmon
(#FFA07A)
(255, 160, 122)

Lightslategray
(#778899)
(119, 136, 153)

Limegreen
(#32CD32)
(50, 205, 50)

Mediumaquamarine
(#66CDAA)
(102, 205, 170)

Mediumseagreen
(#3CB371)
(60, 179, 113)

Mediumslateblue (#7B68EE) (123, 104, 238)	Mediumspringgreen (#00FA9A) (0, 250, 154)	Mediumturquoise (#48D1CC) (72, 209, 204)	Mediumvioletred (#C71585) (199, 21, 133)
Midnightblue (#191970) (25, 25, 112)	Mintcream (#F5FFFA) (245, 255, 250)	Mistyrose (#FFE4E1) (255, 228, 255)	Moccasin (#FFE4B5) (255, 228, 181)
Navajowhite (#FFDEAD) (255, 222, 173)	Navy (!) (#000080) (0, 0, 128)	Navyblue (*) (#A0B0E0) (160, 176, 224)	Oldlace (#FDF5E6) (253, 245, 230)
Olive (!) (#808000) (128, 128, 0)	Olivedrab (#6B8E23) (107, 142, 35)	Orange (#FFA500) (255, 165, 0)	Orangered (#FF4500) (255, 69, 0)
Orchid (#DA70D6) (218, 112, 214)	Palegoldenrod (#EEE8AA) (238, 232, 170)	Palegreen (#98FB98) (152, 251, 152)	Paleturquoise (#AFEEEE) (175, 238, 238)
Palevioletred (#DB7093) (219, 112, 147)	Papayawhip (#FFEFD5) (255, 239, 213)	Peachpuff (#FFDAB9) (255, 218, 185)	Peru (#CD853F) (205, 133, 63)
Pink (#FFC0CB) (255, 192, 203)	Plum (#DDA0DD) (221, 160, 221)	Powderblue (#B0E0E6) (176, 224, 230)	Purple (!) (#800080) (128, 0, 128)
Red (!) (#ff0000) (255, 0, 0)	Rosybrown (#BC8F8F) (188, 143, 143)	Royalblue (#4169E1) (65, 105, 225)	Saddlebrown (#8B4513) (139, 69, 19)
Salmon (#FA8072) (250, 128, 114)	Sandybrown (#F4A460) (244, 164, 96)	Seagreen (#2E8B57) (46, 139, 87)	Seashell (#FFF5EE) (255, 245, 238)
Sienna (#A0522D) (160, 82, 45)	Silver (!) (#C0C0C0) (192, 192, 192)	Skyblue (#87CEEB) (135, 206, 235)	Slateblue (#6A5ACD) (106, 90, 205)
Slategray (#708090) (112, 128, 144)	Snow (#FFFAFA) (255, 250, 250)	Springgreen (#00FF7F) (0, 255, 127)	Steelblue (#4682B4) (70, 130, 180)
Tan (#D2B48C) (210, 180, 140)	Teal (!) (#008080) (0, 128, 128)	Thistle (#D8BFD8) (216, 191, 216)	Tomato (#FF6347) (255, 99, 71)
Turquoise (#40E0D0) (64, 224, 208)	Violet (#EE82EE) (238, 130, 238)	Violetred (*) (#D02090) (208, 32, 144)	Wheat (#F5DEB3) (245, 222, 179)
White (!) (#FFFFFF) (255, 255, 255)	Whitesmoke (#F5F5F5) (245, 245, 245)	Yellow (!) (#FFFF00) (255, 255, 0)	Yellowgreen (#9ACD32) (154, 205, 50)

Приложение 4

Полезные ссылки Интернета

Далее приводятся ссылки на популярные русско- и англоязычные ресурсы, посвященные Интернету, HTML, CSS и созданию Web-сайтов.

Русскоязычные ресурсы

- **html.manual.ru** — справочник по HTML;
- **www.htmlbook.ru** — материалы по HTML, CSS, дизайну, графике и созданию сайтов;
- **www.citforum.ru** — ведущий аналитический портал об информационных технологиях;
- **community.livejournal.com/ru_html** — российское интернет-сообщество по вопросам, связанным с созданием сайтов;
- **www.i2r.ru** — библиотека ресурсов интернет-индустрии;
- **www.fid.ru** — фонд развития Интернета;
- **www.netplanet.ru** — журнал "Планета Интернет";
- **www.webmascon.com** — Webmascon — журнал для Web-мастеров;
- **dweb.ru** — портал Web-мастера;
- **www.firststeps.ru** — первые шаги, материалы по информационным технологиям для начинающих;
- **mega.km.ru** — электронная мегаэнциклопедия;
- **ru.wikipedia.org/wiki** — открытая энциклопедия, которую может редактировать каждый.

Англоязычные ресурсы

- **www.w3.org** — World Wide Web Consortium, разработка стандартов, спецификаций и справочных материалов для WWW;
- **www.icann.org** — The Internet Corporation for Assigned Names and Numbers, технический координатор Сети;
- **www.iso.org** — International Organization for Standardization, международная федерация национальных стандартизующих организаций;
- **www.idc.com** — The Premier Global Market Intelligence Firm, аналитическая компания, мировой лидер в области интеллектуального обеспечения технологического роста;
- **www.htmlgoodies.com** — HTML Goodies, ресурсы по HTML и другим технологиям;
- **www.htmlcodetutorial.com** — Free Reference Guide for Help with HTML, справочник по HTML;
- **www.htmlhelp.com** — HTML Help by The Web Design Group, помощь по HTML;
- **www.livinginternet.com** — сайт об истории и развитии Сети;
- **www.pageresource.com** — A Web Development Tutorial and Information Site, материалы по Web-разработке;
- **www.webopedia.com** — Online Computer Dictionary for Computer and Internet Terms and Definitions, электронный словарь терминов и определений Интернета;
- **www.webreference.com** — The Webmaster's Reference Library, портал для Web-разработчиков;
- **www.w3schools.com** — Online Web Tutorials, HTML, CSS и многие другие технологии.

Приложение 5

Описание компакт-диска

Папки и файлы	Описание	Главы
Glava_4,…,Glava_15	Электронные копии всех примеров из книги, включая готовые шаблоны сайтов и их элементов	4–15
Pictures	Цветные оригиналы рисунков из книги	1, 5, 7–14
Docs	Спецификации HTML и CSS по-русски	4–15
HTMLtags.html	Электронный справочник с примерами по тегам и атрибутам HTML	4–15

Предметный указатель

<

<a> 81, 144, 302
<abbr> 73, 302
<acronym> 73, 303
<address> 79, 303
<applet> 261, 303
<area> 178, 191, 304
<b> 71, 304
<base> 60, 83, 190, 304
<basefont> 70, 142, 304
<bdo> 305
<bgsound> 249, 305
<big> 71, 305
<blink> 305
<blockquote> 73, 305
<body> 54, 141, 189, 305

 65, 306
<button> 202, 306
<caption> 113, 306
 77, 306
<cite> 73, 307
<code> 73, 307
<col> 123, 307
<colgroup> 123, 307
<dd> 104, 308
<del> 73, 308
<dfn> 73, 308
<dir> 308
 77, 309
<dl> 103, 309
<dt> 103, 309
 73, 309
<embed> 249, 309
<fieldset> 206, 310
<font> 69, 142, 310
<form> 198, 311
<frame> 186, 312
<frameset> 185, 312
<h1>, <h2>, <h3>, <h4>, <h5>, <h6> 77, 276, 313
<head> 53, 313
<hr> 79, 313
<html> 53, 313
<i> 71, 314
<iframe> 192, 314
<img> 156, 314
<input> 200, 315
<ins> 73, 316
<isindex> 316
<kbd> 73, 316
<label> 201, 317
<legend> 207, 317
 98, 100, 317
<link> 60, 317
<listing> 317
<map> 178, 318
<marquee> 318
<menu> 318
<meta> 318
<nobr> 66, 319
<noframes> 187, 319
<noindex> 288
<noscript> 255, 319
<object> 243, 319
 101, 320
<optgroup> 205, 320
<option> 204, 321
<p> 63, 321
<param> 247, 261, 321
<plaintext> 321
<pre> 75, 321
<q> 73, 322
<s> 71, 322
<samp> 73, 322
<script> 255, 322
<select> 204, 323
<small> 71, 323
 77, 323
<strike> 71, 323
 73, 323
<style> 216, 324
<sub> 71, 324
<sup> 71, 324
<table> 110, 324
<tbody> 122, 325
<td> 114, 326
<textarea> 203, 326
<tfoot> 122, 326
<th> 114, 327
<thead> 121, 327
<title> 54, 276, 327
<tr> 113, 327
<tt> 71, 328
<u> 71, 328
 98, 328
<var> 73, 328
<wbr> 67, 328
<xmp> 328

A

Action 198
ActiveX 245
AJAX 36, 46
Align 63, 79, 111, 113, 158, 192, 244
Alink 142
Alt 157, 180, 277
Apache 10
ASP 35

B

Background 163
Bgcolor 111, 113, 142, 165
Border 110, 160, 165, 244

C

Cellpadding 110
Cellspacing 110
CGI 33
CHTML 42
Class 45, 55, 78
Color 69, 79, 142

Cols 185, 203
Colspan 118
CSS 39, 213

D

DHTML 40, 240
Dir 56
Disabled 203
DNS 15
DOM 29, 40, 240

E, F

Enctype 198
Face 69
FOAF 44
Frame 111
Frameborder 187

G, H

GUID 245
Height 110, 157, 239, 244
Hidden 207
Href 81, 165, 180
Hspace 160, 244
HTML 16, 38, 51
HTTP 15

I

Id 54
IIS 11
Internet Explorer 19
IP-адрес 12

J

Java 30, 260
JavaScript 28, 254
JSP 36

L

Label 204
Lang 55
Link 142
LiveAudio 250

M

Macromedia Flash 263
Marginheight 187
Marginwidth 187
Maxlength 200
Method 198
MIME-типы 245
Mozilla Firefox 21
Multiple 204

N

Name 85, 160, 186, 198, 245
Netscape Navigator 22
Nohref 180
Noresize 186
Nowrap 118

O, P

Opera 20
OWL 44
PageRank 286
Perl 33
PerlScript 30
PHP 34
Python 35

Q, R

QuickTime 250
RDF 43
Readonly 203
Real color 138, 152
Rel 45
RGB 138
RIA 37
Rows 185, 203
Rowspan 118
RSS 44
Rules 111

S

Safari 22
Scrolling 186
Selected 204
Semantic Web 43
SGML 38
Shape 179
Size 69, 79, 200, 204
Span 124
Src 156, 165, 186
SSI 36
Start 101
Style 55

T

Target 81, 83, 190, 198
TCL 30
Text 142
Title 55, 277
True color 152
Type 100, 101, 200

U

URL 17, 80
Usemap 180, 245

V

Valign 113
Value 101, 200, 204
VBScript 30
Vlink 142
VRML 42
Vspace 160, 245

W

W3C 11
WAP 42
Web 2.0 46
Web-редактор:
  визуальный 135
  текстовый 134
Web-сервер 10
Web-страница 16, 51
  динамическая 27
Web-трансляция 253
Width 79, 110, 124, 157, 239, 244
WML 42
World Wide Web 15, 38

X

X3D 42
XHTML 41
XML 40
XSL 41

А

Абзац 63
Анимация 154
Апплет 30, 260
Архитектура
  "клиент – сервер" 10
Атрибут 52

Б

Базовая линия 159
Баннер 258
Блок сайта 128
Браузер 10, 18

В

Выравнивание:
  в таблице 112, 117
  раздела 77
  рисунков 158
  текста 63

Г

Гамма-коррекция 156
Гиперссылка 80
Гипертекст 16
Глубина цвета 152
Градиент 164
Графика 148
  векторная 149
  оптимизация 175
  растровая 149
  сжатая 151
Графические
  форматы 153
  BMP 151
  GIF 154
  JPEG 155
  PNG 156
  WMF 149
Графическое
  изображение 156

Д

Декларация 215
Директива CSS 215
Доменное имя 13

З

Заголовок 77
Заголовок документа 53
Закладка 85

И

Индекс
  цитирования 285
Индексирование 281
Интерпретация 28, 31
Информер 259

К

Кадр 183
Карта сайта 92, 258
Каскадные таблицы
  стилей 213
Каталог ссылок 94, 287
  "белый" 288
Кернинг 67
Класс 32
  CSS 218
Клиент 10
Кнопка 201
  обзора 202
  пересылки 202
Кодек 251
Комментарий
  CSS 215
  HTML 58
Компиляция 28, 31, 260
Корневой каталог 17
Кэш 294

Л

Логин 11
Локальная сеть 11

М

Маркер 98
Метаданные 43
Метатег 58
Микроформаты 45
Модем 11

Н

Навигация по сайту 89
  графическая 165
  на JavaScript 257
Наследование 214
Начертание 68

О

Обои 163
Объект 32
Объектная модель
  документа 240

П

Палитра 152
  безопасная 141
Переключатель 201
Пиксел 148
Плагин 25
Поле ввода 200
  многострочное 203
  пароля 200
Полиморфизм 214
Потомок 91, 214
Правило CSS 215
Предварительно
  отформатированный
  текст 75
Предок 91, 214
Пробел
  неразрывный 67
Провайдер 9
Прозрачность 154
Пропорциональная
  ширина 117
Протокол 10, 12
  FTP 88
  TCP/IP 12
  прикладного уровня 87
Псевдокласс 221
Псевдотег 45
Псевдоэлемент 221
Пункт 67

Р

Радиокнопка 201
Раздел 77

Размер 110
относительный 185
Размеры:
CSS 223
шрифта 67, 70
Разрешение монитора 148
Ранжирование 281
Реестр 245
Релевантность 282
Рисунок:
векторный 149
кадрирование 174
картированный 177
масштабирование 174
обрамление 160
плавающий 158
растровый 149
фоновый 163
эскиз 169

С

Сайт 18
Свойства:
границы
и заполнителя 232
рамок 233
текста 227
цвета и фона 229
шрифта 225
Свойства CSS 215
Селектор 218
Сервер 10
Сервлет 36
Скрипт 27, 33, 254
Слой 236
Событие 56, 198, 203
Содержимое 52
Специальные символы 67, 75, 86
Список 97
маркированный 98
многострочный 204
многоуровневый 102
нумерованный 100
определений 103
Ссылка 38, 80
абсолютная 82
внешняя 86
внутренняя 85
относительная 83
Стиль 39, 213
внешний 216
внутренний 216
Строка декларации 53
Субдомен 14
Сценарий 27

Т

Таблица 108
вложенная 125
группировка
столбцов 123
группировка строк 121
форматирование 117
Тег 16, 38, 51
блочный 52
контейнерный 52
текстовый 52
унарный 52
Текст
альтернативный 157
Текстура 163
Тело документа 54
Триплет 43

Ф

Файл:
видео 251
звуковой 247
Фильтры 241
Флажок 200
Форма 198
Форматирование:
текста 62
логическое 72
физическое 70
Фрейм 183
вложенный 188
плавающий 192
родительский 191

Х, Ц

Хост 13
Цвет:
RGB-код 139
наименование 139
фоновый 141, 144
шрифта 142

Ч, Ш

Чересстрочная
загрузка 154
Шаблон страницы 130
Шрифт:
векторный 67
моноширинный 68, 75
растровый 67
родовой 226

Э

Элемент 52
дочерний 214
корневой 214
родительский 214

Я

Язык
сценариев 254
Якорь ссылки 80