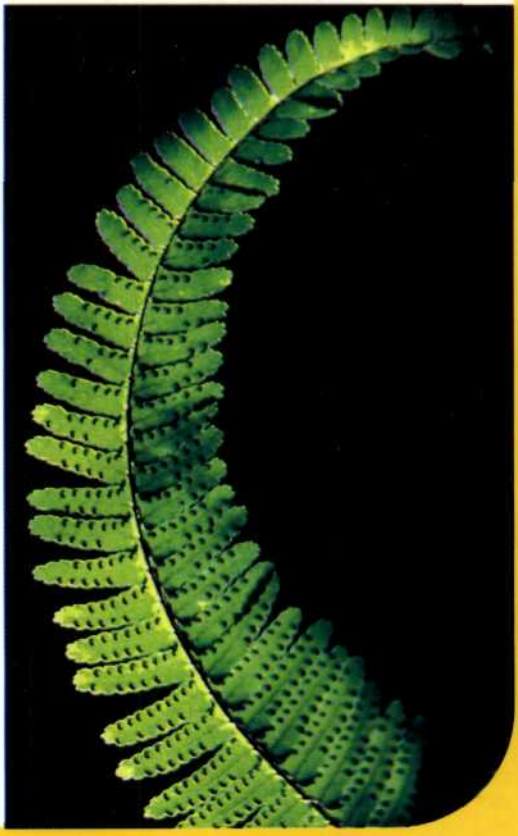
Самоучитель №1
по освоению языка
HTML

САМОУЧИТЕЛЬ

Язык HTML

С помощью этого руководства вы быстро изучите язык HTML и узнаете как

- Создавать и редактировать гипертекстовые документы
- Структурировать, форматировать и размечать HTML-страницы
- Создавать списки и таблицы
- Помещать в HTML-документы графику
- Применять технологию каскадных листов стилей
- Публиковать HTML-документы в Web



ДИАЛЕКТИКА

Е.Л.Полонская

САМОУЧИТЕЛЬ

Язык HTML

САМОУЧИТЕЛЬ

Язык HTML

Е.Л. Полонская

ДИАЛЕКТИКА

Москва • Санкт-Петербург • Киев
2003

ББК 32.973.26-018.2.75
П52
УДК 681.3.07

Компьютерное издательство "Диалектика"

Зав. редакцией *А.В. Слепцов*

По общим вопросам обращайтесь в издательство "Диалектика" по адресу:
info@dialektika.com, http://www.dialektika.com

Полонская Е.Л.

П52 Язык HTML. Самоучитель. : — М. : Издательский дом "Вильяме", 2003. — 320 с. : ил.

ISBN 5-8459-0466-8 (рус.)

Книга предназначена для самостоятельного изучения базового языка создания документов в World Wide Web - языка HTML. Ее главная задача состоит в том, чтобы помочь новичкам освоиться в сложной среде современных интерактивных документов, изучить правила их оформления и имеющиеся инструментальные средства, приобрести навыки, необходимые для успешной публикации своих материалов в Web. В книге рассматриваются основы языка HTML, особенности структуры интерактивных документов, а также самые современные Web-технологии.

Благодаря лаконичному, понятному изложению материал книги будет доступен для восприятия людям любых профессий и возрастов. Книга снабжена многочисленными врезками, а также тематическими контрольными вопросами, способствующими усвоению прочитанного.

ББК 32.973.26-018.2.75

ISBN 5-8459-0466-8 (рус.)

Оглавление

Содержание

Как читать эту книгу

Если эта книга послужит вам приятным послеобеденным чтением, — что ж, буду рада за вас. Но если вы всерьез собрались освоить HTML, то ее нужно читать иначе.

Каждая глава несет в себе информацию трех видов: теоретическую, практическую и справочную. Как усваивать теорию — дело ваше. Одним людям достаточно просто вдумчиво прочесть главу. Другие ведут конспект. Третьи делают нечто наподобие школьной шпаргалки, которой потом пользуются в работе. Подберите тот вариант, который вам удобнее и больше нравится. Здесь все зависит от привычек и особенностей вашей памяти.

Но в любом случае будет не лишним проверить себя: правильно ли вы поняли прочитанное? Для этого в конце каждой главы приводится раздел *Тесты*. Там вы найдете вопросы с несколькими вариантами ответа на каждый. Отметьте те варианты, которые вы считаете правильными, и проверьте свои ответы по *Приложению А*. Обратите внимание, что среди множества предлагаемых вариантов может не оказаться ни одного правильного или несколько правильных ответов. Кроме того, встречаются ответы частично правильные, например, вариант кода, который в принципе работает, но не является оптимальным. В общем, все как в жизни... Так что будьте внимательны.

Однако основные навыки приобретаются, конечно же, на практике. Поэтому настоятельно советую, после того как прочитаете главу, выполнить все описанные в ней примеры самостоятельно. Для этого вам понадобится компьютер, на котором установлены HTML-редактор и броузер. Желательно также иметь доступ к Internet. Но если у вас его нет, не беда. Он нам будет нужен не так часто. При необходимости можете сходить в ближайшее Internet-кафе.

Наконец, в каждой главе содержится информация справочного характера, такая как описание дескрипторов и их параметров. На мой взгляд, зазубривать эти сведения ни к чему. В конце учебника вы найдете сводную таблицу дескрипторов HTML (см. *Приложение Б)*. Можете пользоваться ею, а можете по ходу чтения составить собственную.

Некоторые главы снабжены разделами, озаглавленными *Технические подробности*. В них содержатся более углубленные технические сведения, касающиеся темы, рассматриваемой в данной главе.

Некоторые заметки и рекомендации в книге снабжены пиктограммами. Ответы на наиболее типичные вопросы начинающих помечены пиктограммой *Совет*. Пиктограммой *Внимание* выделены важные моменты, о которых следует помнить при создании Web-страниц. Под пиктограммой *На заметку* приводятся удобные практические приемы, наблюдения — результат многолетнего опыта многих Web-дизайнеров. Это не правила и не стандарты. Вы можете согласиться с ними и взять на вооружение, а можете поспорить, перепроверить и только потом решить, насколько они вам подходят.

В любом случае, я желаю вам успехов в изучении HTML и удовольствия от создания собственных HTML-документов.

Елена ПОЛОНСКАЯ,
автор

Введение. Первые "почему"

Уважаемый читатель! Позвольте задать вам вопрос: зачем вам понадобилось изучить HTML? Для того чтобы стать Web-дизайнером? Чтобы создать виртуальную книгу, учебник, презентацию? Сдать зачет? Для общего развития?

Как бы то ни было, какая-то причина все же побудила вас взять в руки эту книгу, пролистать первые страницы. Пойдете ли вы до конца? Решитесь ли подарить себе удовольствие нового знания, умения? Не берусь судить. Но, говорят, тот, кто однажды отказал себе в таком удовольствии, потом долго об этом жалеет...

Домашняя Web-страничка — это, конечно, далеко не всегда источник дохода (хотя нередко бывает и так). Гораздо чаще это виртуальная визитная карточка, средство общения, "живой" учебник или презентация, которые можно без особых хлопот изготовить самому, не прибегая к помощи программиста. Нередко это первая проба Web-дизайнерского пера, за которой потом появляются другие, лучшие, профессиональные... И всегда — средство самовыражения.

Начинать всегда немного боязно и страшно. Боязно — как нырять в холодную воду после того, как угреешься на теплом песке. Страшно — как все новое. Тебя удивляют правила новой среды. А старожилы удивляются твоим вопросам. Но постепенно образуется некий общий знаменатель. А потом начинает *получаться*. И тогда приходит удовольствие. Как от всего, что делаешь своими руками и делаешь хорошо, — будь то гениальный роман или ровно вбитый гвоздь.

Но поначалу все кажется немного странным. Особенно до тех пор, пока не найдешь ответы на основные "почему".

Почему HTML

Языков программирования и так в последнее время расплодилось довольно много. Зачем было изобретать еще одну тарабарскую систему кодов, на сей раз для Web? Неужели нельзя было взять обычный формат текстовых документов, например Word, со всеми его шрифтами, таблицами, картинками и, само собой, гиперссылками, и написать для него программу-просмотрщик? Зачем было изобретать какой-то язык разметки гипертекста и набирать в куцем Notepad нечто жуткое, не имея возможности даже увидеть, как это выглядит на самом деле? Ведь вот он — Word, под рукой. И все видно, и все понятно, и его документы везде открываются. Похоже, эти интернетчики — просто ненормальные. Создают сложности на ровном месте себе и другим...

У вас никогда не возникало подобных мыслей?

У меня — возникали.

Оставим в покое историю развития Internet. Вряд ли вас удовлетворит ответ в духе "все всегда так делали и мы будем делать так же". Займемся днем сегодняшним. И самыми практическими его вопросами.

Почему объем HTML-страницы так важен? Говоря о месте на жестком диске, мы давно оперируем гигабайтами или, на худой конец, сотнями мегабайт. А тут трясемся над каждым десятком килобайт!

В чем причина? В каналах связи.

Подавляющее большинство отечественных пользователей Internet подключаются к сети по коммутируемой телефонной линии через модем. Повысить качество такой связи они обычно могут только в очень ограниченных пределах, так как определяется оно не столько модемом, сколько бабушкой АТС, из которой весь песок уже давно высыпался.

Попробуем приблизительно оценить, какой объем должна иметь HTML-страница, чтобы посетитель не заскучал и не ушел дальше бродить по Internet, не дожидаясь конца ее загрузки.

Скорость загрузки страницы на компьютер ее посетителя определяется многими факторами. Учесть их все при таком поверхностном расчете не представляется возможным. Поэтому будем опираться только на один из них, который нам известен, — скорость модема. Предположим, она составляет 33,6 Кбит/с. Это значит, что (подчеркнем, не учитывая массу других факторов!) за секунду модем передает 33,6 Кбит / 8 = 4,2 Кбайт данных. Но количество полезных данных меньше, так как на линии часто возникают ошибки, отчего данные приходится передавать повторно. Насколько оно меньше, точно сказать нельзя. Предположим, что в полтора раза. Тогда получится, что пользователь принимает 4,2 / 1,5 = 2,8 Кбайт полезной информации в секунду. Сколько времени он будет терпеливо ждать загрузки страницы? Конечно, это зависит и от темперамента пользователя, и от того, насколько ему нужна именно эта страница, и еще от массы причин. Но предположим, что это время составляет 1 минуту. Тогда, как нетрудно подсчитать, максимальный размер Web-страницы составляет 2,8 * 60 = 168 Кбайт. Нетрудно подсчитать, что для скорости 14,4 кбит/с этот размер составит 72 Кбайт, а для скорости 54,6 Кбит/с — 273 Кбайт.

Конечно, мы многого не учли в нашем примитивном расчете. Однако он позволяет сказать, что размер Web-страницы должен составлять 100—150 Кбайт и уж, во всяком случае, не должен превышать 250 Кбайт.

Но ведь файлы с HTML-кодом гораздо меньше, скажете вы. Действительно, размер самого кода редко превышает несколько десятков килобайт. Но следует помнить, что вместе с кодом на компьютер посетителя страницы передаются файлы с картинками и, возможно, другими нетекстовыми объектами. А их размер может значительно превышать определенный нами лимит. О том, как поступать в таких случаях, вы узнаете из главы 8.

Следует отметить, что это довольно жесткое ограничение. Оно не позволяет использовать на Web-страницах интенсивную графику, аудио- и видеоэффекты. Для того чтобы такие вещи, как просмотр через Internet видеофильмов и телеканалов, стали реальностью, нужны гораздо более мощные каналы передачи данных, — такие, которые позволили бы оперировать не сотнями килобайт, а гигабайтами или хотя бы сотнями мегабайт.

Скачайте из Internet и сохраните на диске любую Web-страницу. Можно даже без картинок. Откройте ее, скажем, в Word 2000 или XP и сохраните как документ Word. Готово? Теперь откройте Explorer (Проводник) или какой-нибудь Commander и сравните размеры обоих файлов. Ну как? Могу поспорить, что файл с кодом HTML не просто меньше, а раз в десять меньше. Или даже в пятьдесят... А теперь припомните свои впечатления, когда вы последний раз загружали объемистую Web-страницу и минут пять глазели на одинокий баннер. Не мелькала ли у вас тогда мысль заставить автора этой страницы частично оплатить вам доступ к Internet?

А у того — свои заботы. Каждый хозяин Web-собственности желает, чтобы его страницу в Internet посещали часто и с удовольствием. Иначе зачем было городить весь этот виртуальный огород? А раз так — страницы должны быть, во-первых, интересными, а во-вторых, *быстро открываться*. За первое HTML ответственности не несет. Зато за второе — отвечает головой. И, поверьте, неплохо справляется. Вы могли убедиться в этом сами, сравнив объем HTML-страницы с объемом документа Word.

Идем дальше. У вас какой компьютер? Pentium II? Прекрасно. P-IV? Ого! "Двойка"? Смотрите-ка, жива ведь старушка... Мака ни у кого нет? Ладно, шучу. А впрочем, вон кто-то тянет руку в заднем ряду. Карманный ПК... Мобильный телефон... Телевизор с Web-приставкой... Что? Холодильник?! Н-да, дожили...

А какая у вас операционная система? Windows 98... NT... 2000... XP... "Могучая кучка" линуксоидов с фанатично горящими глазами... UNIX? Вы, наверное, системный администратор или скоро им будете... Владельцам Mac и карманных ПК выбирать особо не из чего... Плюс то, что заменяет ОС в мобильном телефоне, Web-приставке и... гм, в холодильнике.

То, что нас так много, — это еще ничего. Это даже очень хорошо. Хуже, что у нас такие разные *платформы* — компьютеры и операционные системы. Поэтому нам, *сообществу Internet,* для обмена информацией нужен такой формат, который бы "понимали" *все* броузеры *всех* производителей и *всех* операционных систем, настоящие и по возможности будущие. В PC, Macintosh, карманном ПК и холодильнике. Можем ли мы положиться на двоичный формат отдельно взятого приложения, разработанный отдельно взятой — пусть даже очень большой — компанией? Голосуем: кто за то, чтобы вверить формат наших Web-страниц заботам г-на Гейтса? Заснули, что ли... Ладно, кто против? Нет, похоже, не заснули.

Раз так — возникает следующий вопрос: какой формат самый универсальный? Очевидно, такой, который прочтет и поймет не только программа, но и, в случае чего, человек. Какой формат для этого лучше всего подходит? Правильно — *текстовый.*

Если взять обычный текстовый файл и разметить в нем фрагменты, которые нужно выделить, скажем, курсивом или цветом, обозначить нумерованные и маркированные списки, места вставки рисунков и т.п. (рисунки и другие нетекстовые объекты придется поместить в отдельных файлах), то любой мало-мальски толковый программист практически за неделю "нарисует" вам программку, которая бы отображала такие файлы на экране в более или менее приятно читаемом виде. Вот вам и броузер. Осталось разработать единую систему разметки текста. Или, точнее, *гипертекста,* поскольку, кроме собственно текста — и не простого, а форматированного — в нем ведь есть еще картинки, ссылки на другие документы и файлы, а иногда аудио- и даже ви-

деовставки. Что у нас получится? Язык разметки гипертекста, или, по-английски, HyperText Markup Language — HTML. Вот мы и изобрели велосипед...

И последний, но немаловажный аргумент в пользу HTML. К сожалению, в силу — как бы это сказать помягче? — некоторых специфических традиций отечественного рынка программного обеспечения этот аргумент не вполне очевиден. Вы когда-нибудь интересовались, *сколько стоит* текстовый процессор Microsoft Word? Нет? Поинтересуйтесь. За эту сумму можно устроить себе неплохой летний отпуск. А текстовых редакторов для ASCII — пруд пруди. Помимо Notepad и редакторов, встроенных во всяческие "Коммандеры", на Web-узлах бесплатного ПО их сотни мегабайт. И это не считая специализированных HTML-редакторов, среди которых тоже немало бесплатных и условно-бесплатных. Если бы вся Всемирная Паутина перешла на стандарт Word, она бы, пожалуй, быстро лишилась многих небогатых, но законопослушных авторов. А поскольку слишком много народу уже жизни не мыслит без Internet, то происходит как раз обратное: Word и весь Microsoft Office, да и другие офисные пакеты, дружно принимают HTML в виде еще одного текстового формата.

Почему Notepad

Notepad, он же Блокнот Windows — излюбленный редактор Web-дизайнеров. Хотя есть много других хороших и не менее уважаемых ими специализированных редакторов для HTML, таких как MacroHTML или HomeSite. Уважают их, главным образом, за простоту, доступность, компактность. А еще за невмешательство в творческие дела дизайнера. Все очень просто: вы вводите в редакторе HTML-код и любуетесь результатом в окне параллельно работающего броузера. Редактор в лучшем случае ускорит ввод ключевых слов и поможет с синтаксической проверкой. Остальное — за вами.

Другой тип HTML-редакторов — *визуальные,* такие как известный Microsoft FrontPage Express или гораздо более любимый Web-дизайнерами Macromedia DreamWeaver. Их интерфейс построен по тому же принципу, что и интерфейс текстового процессора, скажем Word. Кстати, Word 2000 или XP вполне сойдет в качестве визуального HTML-редактора.

Преимущество визуальных HTML-редакторов в том, что для работы с ними можно вообще не знать HTML. К сожалению, визуальные редакторы страдают тем же недостатком, что и трансляторы языков программирования, преобразующие текст на C или Pascal в ассемблерный код. А именно — неоптимальностью этих самых кодов. Спросите любого программиста, и он вам скажет, что ассемблерный код, написанный вручную, в несколько раз меньше кода такой же программы, написанной, скажем, на Pascal и преобразованной в ассемблерный код с помощью транслятора. Удивляться нечему: решения человека-специалиста всегда красивее решений машины. Зато и труда, и времени они требуют несравнимо больше. Впрочем, вероятно, недалек тот день, когда при загрузке страницы разница в несколько килобайтов или даже мегабайтов не будет так заметна, как сейчас. Тогда визуальные HTML-редакторы займут на компьютере Web-дизайнера такое же достойное место, какое сейчас занимают на компьютере программиста визуальные среды разработки, такие как, например, Delphi.

Увы, этим грехи визуальных редакторов не исчерпываются. Часто страница выглядит в броузере совсем не так, как до этого она выглядела в окне редактора. Приходится лезть в длинный, запутанный код, искать там ошибки и вручную их исправлять.

Когда-то давно мне встретились в одной книжке десять шуточных заповедей программиста. Одна из них гласила: "Не возжелай программы ближнего твоего!". В применении к нашей задаче она звучит так: "Не возжелай HTML-кода ближнего твоего, в особенности если этот ближний — FrontPage, а ты собрался изучать HTML".

Где взять хороший HTML-редактор?

HTML-редакторов множество, и у каждого из них есть свои преимущетсва и недостатки. Поэтому однозначно сказать, какой именно вам подойдет лучше всего, нельзя. Одним нравится HomeSite, другим — 1st Page, третьим — Arachnophilia или Easy HTML. Или другой из десятков им подобных. Здесь все в конечном счете сводится к личным вкусам.

Для того чтобы подобрать редактор по душе, советую посетить несколько Web-хранилищ бесплатного и условно-бесплатного программного обеспечения и там подобрать себе редактор по душе и по карману. Поэтому вместо того, чтобы сравнивать редакторы (тем более что регулярно появляются новые версии с новыми функциями), я лучше приведу несколько адресов, по которым стоит обратиться.

- 1st Page: www.evrsoft.com
- Arachnophilia: www.arachnoid.com
- HomeSite: www.macromedia.com
- Easy HTML: www.ukrwest.net
- Также виртуальные архивы ПО: FreeSoft (www.freesoft.ru), Freeware.ru (www.freeware.ru), ListSoft (www.listsoft.ru), Softarea (www.softarea.ru) и др.

Почему HTML не является языком программирования

Многие ошибочно называют HTML языком программирования. К сожалению, это не так. И дело здесь даже не в том, что у HTML нет компилятора, а только встроенный в броузер интерпретатор: BASIC и JavaScript тоже обходятся одними интерпретаторами, но это не мешает им называться языками программирования. Дело в том, что в HTML отсутствует главный атрибут, присущий любому языку программирования, — *команды.* На HTML нельзя задать *последовательность действий,* а можно только описать, как броузер должен вводить на экран тот или иной документ. Если же на Web-странице действительно должно что-то выполняться, например вестись форум, то используются настоящие языки программирования, такие как Java и JavaScript (см. главу 16). Поэтому говорить "программа на HTML" не вполне корректно. Мы будем пользоваться термином "HTML-код".

Резюме

Язык HTML, или универсальный язык разметки гипертекста, используется для создания самых разных интерактивных документов с гиперссылками и элементами мультимедиа — Web-страниц, интерфейсов, презентаций, электронных книг и учебных пособий. Файлы с HTML-кодом — это обычные текстовые файлы, доступные для чтения как программе, так и человеку. Благодаря этому HTML-страницы можно редактировать и просматривать на любом компьютере и в любой операционной системе.

Для создания HTML-страниц можно пользоваться любым текстовым редактором, но существуют и специализированные программы. Эти HTML-редакторы делятся на два типа: визуальные и невизуальные. Визуальные HTML-редакторы обладают интуитивно понятным интерфейсом и не требуют много времени на освоение, но генерируют очень длинный, неоптимальный и малопонятный HTML-код, который потом трудно редактировать. К редакторам этого типа относится, например Microsoft FrontPage Express.

Невизуальные редакторы требуют знания HTML, но лишены недостатков визуальных редакторов. Кроме того, это, как правило, небольшие, компактные программы, бесплатные или условно-бесплатные. К HTML-редакторам этого типа относятся, например MacroHTML и HomeSite.

HTML не является языком программирования. Поэтому для того чтобы на HTML-странице что-то выполнялось, например, выводился какой-то текст или менялась картинка в ответ на определенные действия пользователя, необходимо пользоваться специальными средствами, расширяющими возможности HTML, такими как язык программирования JavaScript.

Тесты

1. Что из нижеперечисленного является броузером?
   - **а)** Microsoft Word
   - б) Microsoft Internet Explorer
   - в) Microsoft FrontPage Express
   - **г)** HomeSite
   - д) Netscape Navigator
   - **е)** Opera
   - ж) Mozilla
   - **з)** Notepad
   - и) PhotoShop
   - к) Google.com
2. Что из нижеперечисленного можно использовать для просмотра HTML-страницы?

а) Microsoft Word

б) Microsoft Internet Explorer

в) Microsoft FrontPage Express

г) HomeSite

д) Netscape Navigator

е) Opera

ж) Mozilla

з) Notepad

и) Windows

к) Linux

л) PhotoShop

м) Google.com

3. Что из нижеперечисленного можно использовать для просмотра кода HTML-страницы?

а) Microsoft Word

б) Microsoft Internet Explorer

в) Microsoft FrontPage Express

г) HomeSite

д) Netscape Navigator

е) Opera

ж) Mozilla

з) Notepad

и) Windows

к) Linux

л) PhotoShop

м) Google.com

Глава 1

Основные принципы HTML

В этой главе...

- Первый опыт "общения" с броузером
- HTML — язык дескрипторов
- Первый опыт форматирования: курсив
- Жирный шрифт и подчеркивание

Первый опыт "общения" с броузером

Итак, Notepad открыт, рукава засучены, в глазах — огонь... Приступим.

Прежде всего давайте разберемся: что "понимает" и чего "не понимает" броузер? Для этого создадим в Notepad обычный текстовый файл. Можно написать там что-то вроде:

```
Я хочу, чтобы ЭТО можно было прочесть в броузере!
```

И сохраним. Назовем его, скажем, testhtml.

Теперь откроем созданный файл в броузере. Это можно сделать двумя способами: с помощью команды Файл ⇨ Открыть (File ⇨ Open), либо перетащив пиктограмму из Проводника или файлового менеджера в окно броузера.

Несмотря на все усилия сообщества Internet, жесткого стандарта HTML до сих пор не существует. Некоторые конструкции языка отображаются в разных броузерах по-разному. А иногда не воспринимаются вовсе, и тогда посетители Web-страницы видят на экране нечто невообразимое... Поэтому хороший Web-дизайнер обязательно проверяет свои работы в нескольких броузерах. В качестве "джентльменского набора" можно предложить Internet Explorer, Netscape Navigator, Opera. Причем желательно *предпоследние* версии: как бы ни была привлекательна идея "идти в ногу со временем", самые новые версии, как правило, содержат ошибки, которые будут исправлены через несколько месяцев, да и немногие пользователи успели их установить.

Получилось? Повезло вам с броузером...

Не текст, а абракадабра? Смените кодировку. С помощью команды Кодировка (Coding) или чего-то похожего, что находится в меню Вид (View). Более точное название команды зависит от броузера. Попробуйте разные виды кириллицы, начиная с Windows и КОИ8-Р. Забегая вперед, отметим, что, вообще-то, первая

обязанность Web-дизайнера — избавить посетителя от необходимости подобных действий. И не такая уж обременительная — нужно всего лишь поставить в начале документа специальный дескриптор...

HTML — язык дескрипторов

Что-что поставить в начале?..

Дескриптор.

"Вехи", которыми размечают гипертекст, называются дескрипторами (tags). В обычном, некомпьютерном английском языке слово *tag* означает "признак", "метка", "маркер". Дескриптор в HTML — это некое ключевое слово или сокращение, которое служит признаком того или иного форматирования данной части документа. Для того чтобы дескриптор можно было отличить от остального текста, его заключают между знаками "больше" и "меньше": <ДЕСКРИПТОР>.

Многие дескрипторы принадлежат к *контейнерному типу:* в них, как в контейнер, заключается фрагмент текста, форматирование которого определяется этим дескриптором, — например, выделение курсивом или "привязка" гиперссылки. Контейнерные дескрипторы всегда парные: каждому *открывающему* дескриптору соответствует одноименный *закрывающий,* причем закрывающий дескриптор отличается от открывающего тем, что начинается с косой черты (заметьте — обычной, а не обратной, к которой вы, возможно, привыкли, работая в Windows):

```
Я хочу, чтобы <ДЕСКРИПТОР1> ЭТО </ДЕСКРИПТОР1> можно было прочесть в броузере!
```

Встречаются и неконтейнерные, одинарные дескрипторы. Они не содержат текста, который нужно так или иначе отформатировать, а определяют характеристики документа в целом или описывают объект, который нужно вставить в данной точке документа, например картинку. Такие дескрипторы не имеют закрывающих "двойников":

```
Я хочу, чтобы ЭТО <ДЕСКРИПТОР2> можно было прочесть в броузере!
```

Контейнерные дескрипторы, как правило, могут быть *вложенными* друг в друга, как матрешки:

```
Текст1 <ДЕСКРИПТОР1> текст2 <ДЕСКРИПТОР2> текст3 </ДЕСКРИПТОР2> текст4
</ДЕСКРИПТОР1> текст5
```

В результате текст1 и текст5 выводятся без форматирования, текст2 и текст4 — с форматированием, описываемым дескриптором 1, а текстЗ — с форматированием, описываемым дескрипторами 1 и 2. Таким образом, свойства двух дескрипторов как бы накладываются друг на друга, в результате чего достигаются разнообразные эффекты. Пример такого эффекта будет рассмотрен в конце этой главы.

Главное правило при использовании вложенных дескрипторов — соблюдать последовательность: первым закрывается дескриптор, открытый последним. Собственно, как и при сборке куклы-матрешки...

Первый опыт форматирования: курсив

А теперь попробуем что-нибудь отформатировать с помощью дескрипторов.

Вспомним: как мы, например, привыкли в Word делать текст курсивным? Выделяем текст и щелкаем на соответствующей кнопке, или просто нажимаем <Ctrl+I>. Кстати, в Word с английским интерфейсом на кнопке тоже нарисована буква I, а не К. Почему I? Потому что по-английски курсив — *Italic*. В HTML для того же действия существуют дескрипторы `<I>` и `</I>`. Текст, помещенный между ними, отображается в броузере курсивом.

Попробуем?

Заключим, например, в "курсивные" дескрипторы слово ЭТО:

```
Я хочу, чтобы <I>ЭТО</I> можно было прочесть в броузере!
```

Сохраним файл и "освежим" картинку в броузере. Если вы его еще не закрыли, можете воспользоваться командой Вид о Обновить (View ⇨ Refresh). В некоторых броузерах для этого есть специальная клавиша. Например, в IE это <F5>.

Ну как, нравится (рис. 1.1)? Лично мне — не очень.

Похоже, броузер по-прежнему отказывается принимать наш текст за HTML. Попробуем его перехитрить — изменим расширение файла, ничего не меняя внутри. Вы, наверное, сохранили файл с тем расширением, что предлагал Notepad – .txt. А нужно — `.htm` или `.html`. Не ищите HTML в списке типов Notepad — тип-то все равно *текстовый*. Просто при сохранении введите не только имя файла, но и расширение: `testhtml.html`. Ну вот, теперь другое дело (рис. 1.2).

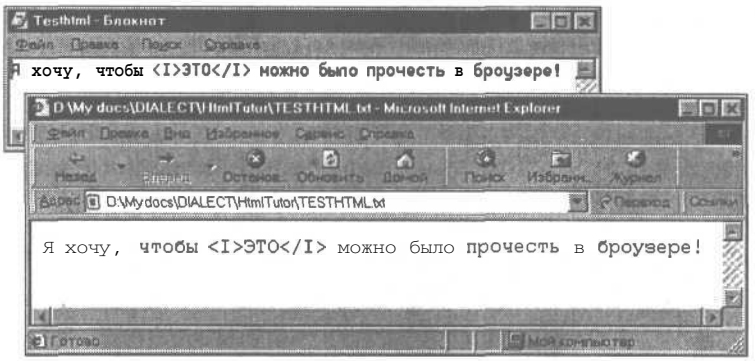


Рис. 1.1. Для того чтобы в броузере отображался отформатированный текст, одних дескрипторов недостаточно

Теперь, пожалуй, слово *это* можно написать обычными, строчными буквами. К слову сказать, посетители Web-страниц не любят, когда с ними "говорят" БОЛЬШИМИ БУКВАМИ. Да и у вас, вероятно, такой стиль вызывает раздражение: подсознательно он воспринимается как крик. Зачем КРИЧАТЬ, когда можно мягко *расставить акценты*?

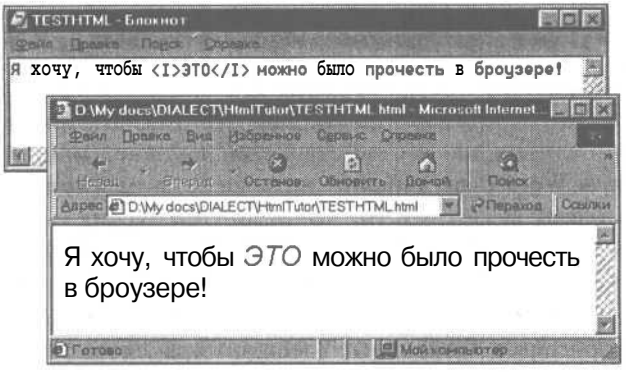


Рис. 1.2. Для того чтобы броузер воспринимал содержимое файла как HTML-код, этот файл должен иметь расширение `.htm` или `.html`

Жирный шрифт и подчеркивание

Как вы думаете, какие дескрипторы соответствуют жирному шрифту и подчеркиванию? По аналогии с Word, жирному должен соответствовать дескриптор <B> (от слова *Bold),* а подчеркиванию — <U> (от *Underline).* Так и есть. Конечно же, эти дескрипторы, как и <I>, являются контейнерными, т.е. имеют соответствующие закрывающие "пары".

Впрочем, увлекаться дескриптором <U> не рекомендуется. Ведь на Web-страницах подчеркиванием традиционно обозначаются гиперссылки (см. главу 9), и посетители к этому привыкли. Вряд ли имеет смысл обманывать их ожидания только лишь красивости ради.

А что нужно сделать, чтобы слово это выводилось жирным шрифтом, а первая буква э — еще и курсивом? Правильно, использовать вложенные дескрипторы <I> и <B> (рис. 1.3).

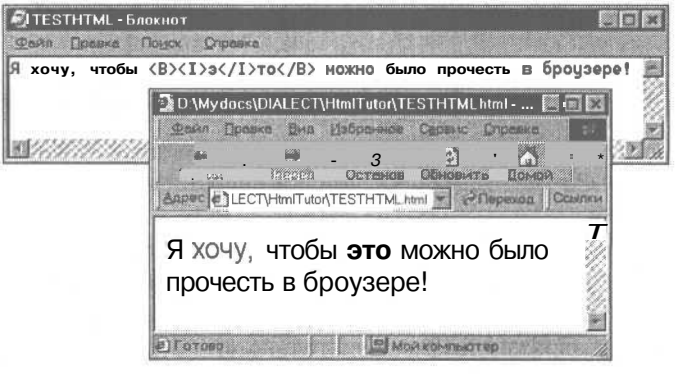


Рис. 1.3. С помощью вложенных дескрипторов <I> и <B> достигается эффект полужирного курсива

Резюме

Для того чтобы броузер воспринимал содержимое текстового файла как HTML-код, этот файл должен иметь расширение .htm или .html.

Разметка гипертекста средствами HTML происходит путем вставки в текст *дескрипторов* — специальных кодовых слов, определяющих элементы форматирования. Для того чтобы дескрипторы отличались от остального текста, их заключают между знаками "больше" и "меньше", например: <I>, <U>, **<B>.**

Дескрипторы бывают контейнерные (парные) и неконтейнерные (одинарные), причем первые встречаются чаще. Контейнерный дескриптор состоит из двух частей — открывающего и закрывающего дескрипторов, — между которыми находится форматируемый текст, например:

<I> текст курсивом </I>

Закрывающий дескриптор отличается от открывающего наличием косой черты. Контейнерные дескрипторы бывают вложенными, например:

<I> курсив <B> жирный курсив </B> курсив </I>

При использовании вложенных дескрипторов нужно следить за их последовательностью: дескриптор, открытый первым, закрывается последним.

Тесты

1. Какой Или какие из следующих фрагментов HTML-кода содержат ошибку?
   а) Зачем КРИЧАТЬ, когда можно мягко <I>расставить акценты</I>?
   б) Текст курсивом: <I></I>
   в) Многие дескрипторы принадлежат к <\I>контейнерному типу<I>
2. Какой или какие из следующих фрагментов HTML-кода не содержат ошибки?
   а) Увлекаться дескриптором <U> не рекомендуется
   б) С помощью вложенных дескрипторов «I» и «B» достигается эффект полужирного курсива
   в) Контейнерные дескрипторы бывают вложенными, например: <I><U></U></I>
   г) Разметка гипертекста средствами HTML происходит путем вставки в текст <I><U>дескрипторов</I></U>
3. Как будет выглядеть в окне броузера следующий фрагмент HTML-страницы?

   Для того чтобы броузер воспринимал содержимое файла как HTML-код, этот файл должен иметь расширение <U><I>.htm</I> или <B>.html</B></U>

   а) Для того чтобы броузер воспринимал содержимое файла как HTML-код, этот файл должен иметь расширение *.htm* или **.html**
   б) Для того чтобы броузер воспринимал содержимое файла как HTML-код, этот файл должен иметь расширение *.htm* или **.html**
   в) Для того чтобы броузер воспринимал содержимое файла как HTML-код, этот файл должен иметь расширение ***.htm*** *или* ***.html***

Глава 2

Абзацы

В этой главе...

- Как заставить броузер перейти на новую строку
- Абзац с параметрами
- Разрыв строки

Как заставить броузер перейти на новую строку

Отлично! Похоже, в первом приближении у нас есть все, что нужно для простенькой странички. Правда, черно-белой и без картинок. Ну, да невелика важность. Сейчас попробуем написать что-нибудь побольше, например:

```
Я памятник себе ВОЗДВИГ нерукотворный,
К нему не зарастет народная тропа.

(C) <I><B>A.C. П</B>ушкин</I>
```

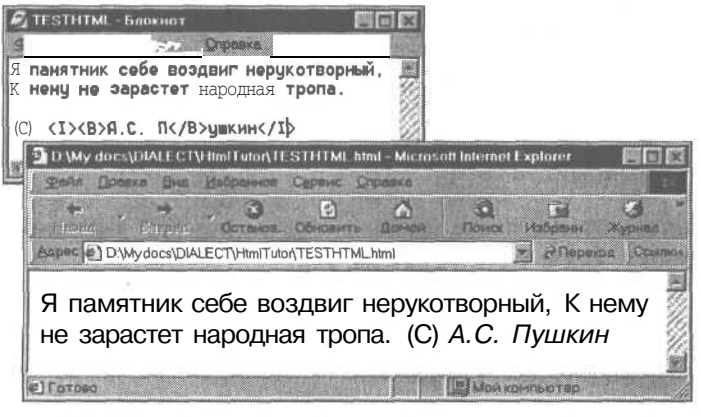


Рис. 2.1. Попытка разбить текст на абзацы: первый блин комом

По идее, подпись "A.C. Пушкин" должна получиться курсивной, а инициалы и первая буква фамилии — еще и жирными. Так и есть (рис. 2.1). Но где переход на новую строку после запятой? Где пустая строка между стихами и подписью? И самое главное — как сделать, чтобы все это было?

Броузеру безразлично, какой разделитель стоит между словами в HTML-коде: переход на новую строку, табуляция или пробел. Безразлично ему и количество разделителей. Сколько ни ставь пробелов между словами в коде, на экране слова все равно будут разделены одним пробелом. Сколько ни ставь в коде абзацев, в окне броузера переход на следующую строку будет происходить тогда, когда текст приблизится к границе окна.

Для того чтобы разделить текст на абзацы, в HTML применяется дескриптор <P> — от английского *paragraph*, что в переводе означает вовсе не "параграф" (как было бы логично с точки зрения русского языка), а именно "абзац". Что ж, попробуем иначе:

```
<P>Я памятник себе воздвиг нерукотворный,</P>
<P>К нему не зарастет народная тропа.</P>
<P>(c) <I><B>A.C. П</B>ушкин</I></P>
```

Уже лучше (рис. 2.2).

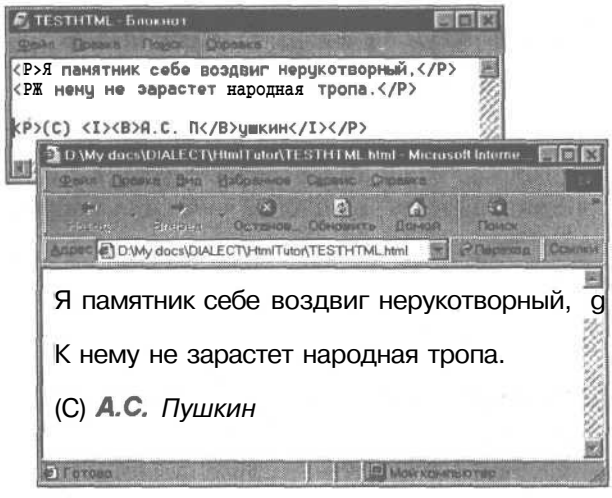


Рис. 2.2. Абзацы в HTML размечаются с помощью дескриптора **<P>**

А как увеличить отступ после стихов перед подписью? Наверное, нужно вставить пустую строку, как в Word? Скажем, так:

```
<P>Я памятник себе воздвиг нерукотворный,</P>
<P>К нему не зарастет народная тропа.</P>
<P></P>
<P>(c) <I><B>A.C. П</B>ушкин</I></P>
```

М-да... Похоже, "пустой" абзац для броузера просто не существует (рис. 2.. А что, если абзац будет не пустым? Если заполнить его чем-то видимым для броузера, но невидимым для нас? Скажем, пробелом или табуляцией (см. рис. 2.3):

```
<P>Я памятник себе воздвиг нерукотворный,</P>
<P>К нему не зарастет народная тропа.</P>
<P>     </P>
<P>(c) <I><B>A.C. П</B>ушкин</I></P>
```

Похоже, нас опять не понимают...

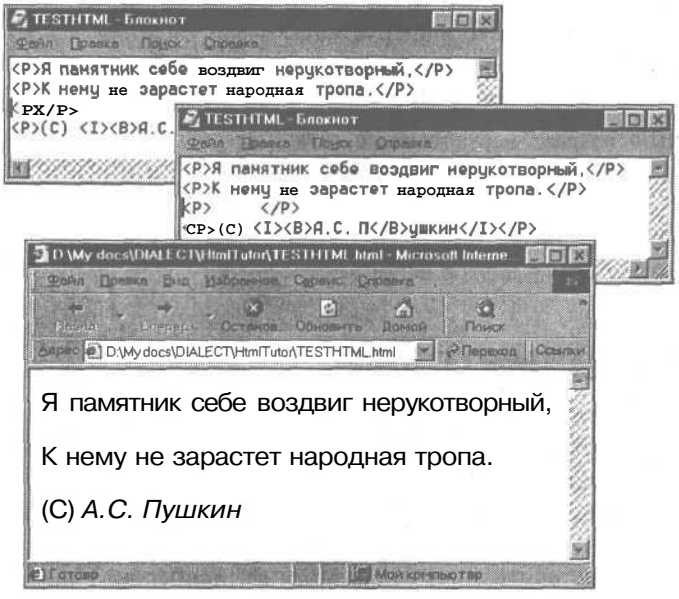


Рис. 2.3. Простая вставка "пустой" строки в коде не позволяет увеличить отступ между строками в броузере. Заполнение ее пробелами также не дает желаемого результата

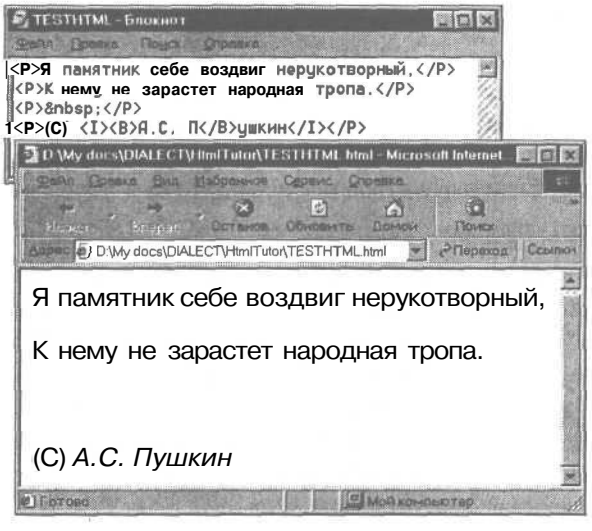


Рис. 2.4. Для того чтобы вставить в текст "пустую" строку, используйте символ неразрывного пробела

Придется пойти на хитрость: заменить обычный пробел *неразрывным*. Основное назначение неразрывного пробела — разделять слова, но запрещать в этом месте переход на новую строку. Но здесь мы воспользуемся другим его свойст-

вом: если обычный пробел (а также, кстати, и табуляция) расценивается броузером как *отсутствие символа,* то неразрывный пробел для него — полноценный символ, хоть и невидимый. Что и требуется.

Но как ввести н7еразрывный пробел в Notepad? С помощью специального обозначения —   (подробнее об этом читайте в главе 3):

```
<P>Я памятник себе воздвиг нерукотворный,</P>
<P>К нему не зарастет народная тропа.</P>
<P> </p>
<P>(c) <I><B>A.C. П</B>ушкин</I></P>
```

Вот теперь (рис. 2.4) — другое дело!

Абзац с параметрами

Действительно, неразрывный пробел — полезная вещь: его можно использовать и для "красных строк", и вообще везде, где нужно изобразить отступ или оставить немного свободного места.

А если много? Как, например, разместить подпись *А.С. Пушкин* не с левого, а с правого края строчки?

Конечно, можно вставить много неразрывных пробелов. Примерно так поступают, когда печатают на пишущей машинке. Но так работать на компьютере — это, по меньшей мере, непрофессионально. Было бы просто нелепо предполагать, что при разработке стандарта HTML не предусмотрели возможность выравнивания текста по правому краю.

Конечно же, такая возможность есть — с помощью *параметров* дескриптора <P>.

Параметры для дескрипторов HTML — обычное дело. У большинства дескрипторов HTML есть параметры. Благодаря им язык разметки гипертекста становится более гибким, появляется больше возможностей при меньшем количестве дескрипторов. Параметры и их значения вписываются между именем дескриптора и закрывающей угловой скобкой. Если параметров несколько, то они разделяются пробелами:

```
<ДЕСКРИПТОР параметр1=значение1 параметр2=значение2 ...>
```

У дескриптора <P> всего один параметр — align. Он принимает одно из четырех значений — left, right, center или justify. Как вы уже, вероятно, догадались, этот параметр "отвечает" за выравнивание текста — соответственно по левому или правому краю, по центру или по обоим краям сразу. Впрочем, значение justify поддерживается не всеми броузерами, хотя в IE проблем не возникает.

По умолчанию обычно текст выравнивается по левому краю, что соответствует коду <P align=right>. Попробуем отформатировать наш пример по-другому (рис. 2.5):

```
<P>Я памятник себе воздвиг нерукотворный,</P>
<P>К нему не зарастет народная тропа.</P>
<P> </p>
<P align=right>(c) <I><B>A.C. П</B>ушкин</I></P>
```

Ура, заработало!

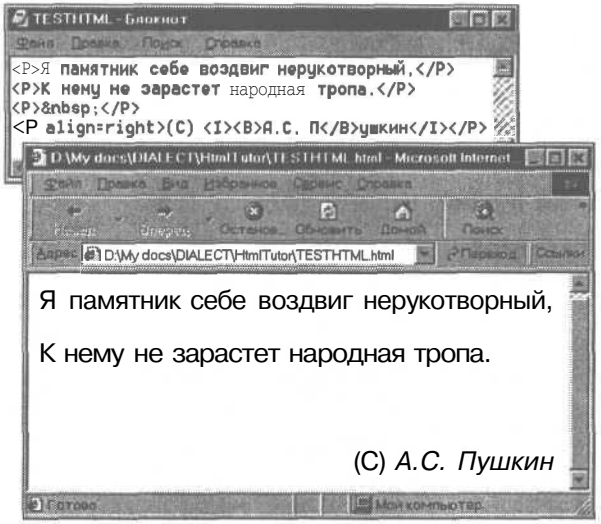


Рис. 2.5. Выравнивание абзаца по правому краю с помощью параметра align

Вероятно, вы уже заметили, что абзацы в HTML выглядят иначе, чем в книгах. В "бумажной" литературе абзац — это текст, который начинается с новой строки, причем первая строка абзаца, как правило, "красная", т.е. начинается с некоторым отступом относительно остальных строк.

Такова традиция русской литературы. В англоязычной литературе традиция другая: вместо "красной строки" абзацы разделяются большими интервалами между строками. Именно эта традиция сохранилась в электронных документах. Конечно, главная причина этого — та, что Internet родом из англоязычных стран. Но, кроме того, читать текст с экрана труднее, чем с листа бумаги. Для того чтобы облегчить этот процесс, дизайнеры прилагают все усилия, чтобы как-то разбивать информацию на блоки, всеми силами избегают больших и однородных массивов текста. И абзац с отступами — одно из простейших средств добиться этого.

Разрыв строки

А нельзя ли как-нибудь уменьшить расстояние между строками в нашем примере? Великовато оно. Но в дескрипторе **<P>** подходящего параметра нет.

Воспользуемся другим дескриптором —
. Означает он переход на новую строку, но не на новый абзац — то, что в Word называется "разрыв строки". При этом:

- по умолчанию в новой строке сохраняется тип выравнивания, присвоенный всему абзацу;
- расстояние между новой строкой и предыдущими строками абзаца равно расстоянию между остальными строками абзаца.

Именно это нам и требуется. Теперь лишние абзацы, в том числе и "невидимый", можно убрать (рис. 2.6):

```
<P>Я памятник себе воздвиг нерукотворный,<BR>
    К нему не зарастет народная тропа.</P>
<P align=right>(c) <I><B>A.C. П</B>ушкин</I></P>
```

Как раз то, что нам нужно!

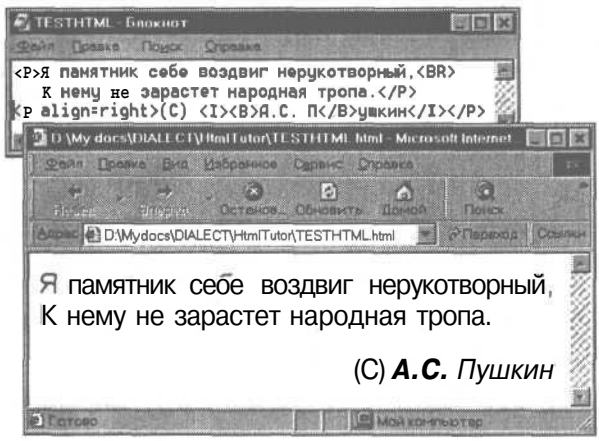


Рис. 2.6. Для того чтобы уменьшить расстояние между строками, замените дескриптор <P> дескриптором

Обратите внимание: дескриптор
 — одиночный. У него нет парного закрывающего дескриптора. И неудивительно: ведь этот дескриптор "работает" не на участке текста, а в той точке, где он поставлен.

Интересно, что и для дескриптора <P> указывать парный закрывающий дескриптор не обязательно. Броузер сам "закрывает" предыдущий абзац, когда начинается следующий. Но полагаться на броузер в таких делах можно далеко не всегда. Попробуйте не закрыть дескриптор <I> или <в> — сами увидите, что получится.

Понятно, что код HTML-страницы читают не только броузеры, но и люди. Вы уже завтра захотите что-нибудь добавить или исправить. Чтобы было проще разбираться в хитросплетениях дескрипторов, используют следующие приемы.

Правильная расстановка отступов и абзацев. Как вы уже знаете, любое количество пробелов, абзацев и табуляций, поставленных в HTML-коде подряд, воспринимается броузером как один разделитель между словами. Зато для тех, кто будет читать и править этот код, они по-прежнему выполняют свое основное назначение. Эта особенность позволяет разделять логически законченные фрагменты кода, не меняя вид страницы в броузере. Обратите внимание, что пробелы можно использовать и внутри дескрипторов. Броузер одинаково воспримет <P align=right> и <P align = right>.

Выделение дескрипторов большими буквами. Броузеру безразлично, в каком регистре набраны дескрипторы: <P ALIGN=RIGHT>, <P Align=Right> или <p align=right>. Некоторые предпочитают писать дескрипторы прописными буквами, чтобы код HTML-страницы было удобнее читать.

Комментарии. Для той же цели — удобства чтения и правки — служат *комментарии*. Комментарий — это часть HTML-кода, которую броузер не выводит на экран. Комментарии обозначаются так:

```
<!-- Здесь мог быть ваш комментарий -->
```

Комментарии служат двум целям: в одних случаях они позволяют снабдить пояснениями малопонятный код, а в других — скрыть для броузера код, который он обрабатывать не должен. Это может быть сценарий, написанный на каком-нибудь языке программирования, или временно не используемый фрагмент HTML-кода.

Резюме

Символы перехода на новую строку, используемые для разбиения на абзацы обычного текста, в коде HTML воспринимаются как обычные пробелы. Поэтому для создания абзацев в HTML-страницах используются специальные дескрипторы — <P>.

Для изменения выравнивания абзаца — по левому краю, по правому краю, по центру и по ширине — используется параметр дескриптора <P> — align, — принимающий значение left, right, center и justify, соответственно. Так, для выравнивания абзаца по центру используется следующий код:

```
<P align = center> Какой-нибудь заголовок
```

Следует отметить, что, несмотря на то, что дескриптор <P> — контейнерный, указывать для него закрывающий дескриптор не обязательно: ведь начало следующего абзаца — это одновременно и конец предыдущего.

С помощью дескрипторов <P> текст разбивается на абзацы, согласно традициям англо-американской полиграфии — без красной строки и с увеличенным отступом между абзацами. Для создания красной строки используют символы неразрывного пробела , а для уменьшения отступов — дескриптор разрыва строки <BR>. В последнем случае при переходе на новую строку сохраняется выравнивание, заданное параметром align предыдущего дескриптора <P>.

Тот факт, что броузер не различает пробелы, табуляции и абзацы в коде HTML-страницы, а также не реагирует на несколько этих символов, поставленных подряд, позволяет сделать код читабельнее, не меняя вид страницы в броузере.

Кроме того, для удобства чтения используют также тот факт, что HTML-код нечувствителен к регистру символов. Часто дескрипторы набирают прописными буквами, чтобы они четче выделялись среди остального текста.

Наконец, для удобства чтения и правки используются комментарии. Комментарий в HTML-коде заключается между символами <!-- и -->:

```
<!-- комментарий -->
```

Тесты

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а) <P> Для того чтобы перейти на новую строку, используется тег <P>

   б) <P> Для того чтобы перейти на новую строку, используется тег <P></P>

   в) <P> Для того чтобы перейти на новую строку, используется тег «P»</P>

   г) <R> Для того чтобы перейти на новую строку, используется тег «P»</R>

   д) «P» Для того чтобы перейти на новую строку, используется тег P

2. В каком случае абзац будет выровнен по левому краю?

   а) <P align = center> Этот абзац выровнен по левому краю?

   б) <P align = left> Этот абзац выровнен по левому краю?

   в) <P align = right> Этот абзац выровнен по левому краю?

   г) <P align = justify> Этот абзац выровнен по левому краю?

   д) <P> Этот абзац выровнен по левому краю?

   е) <P align> Этот абзац выровнен по левому краю?

3. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а) <P align=center> Этот текст выровнен по центру

   б) <P ALIGN=CENTER> Этот текст выровнен по центру

   в) <P_align=center> Этот текст выровнен по центру

   г) <P
   align = center> Этот текст выровнен по центру

   д) <P aLiGn=cEnTeR> Этот текст выровнен по центру

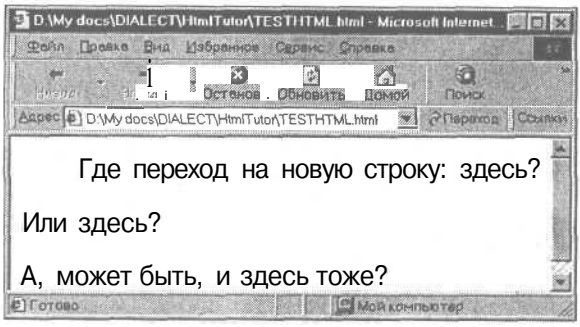
   е) <P center = align> Этот текст выровнен по центру

4. Как будет выглядеть в окне броузера следующий фрагмент HTML-страницы?

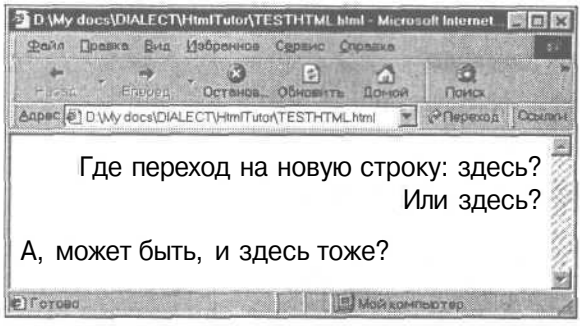
   <p align = right> Где переход на новую строку:
   здесь?
 Или здесь?
   <P> А может быть, и здесь тоже?

   (выберите один из вариантов ответа, предлагаемых на рис. 2.7)

(а)



(б)



(в)

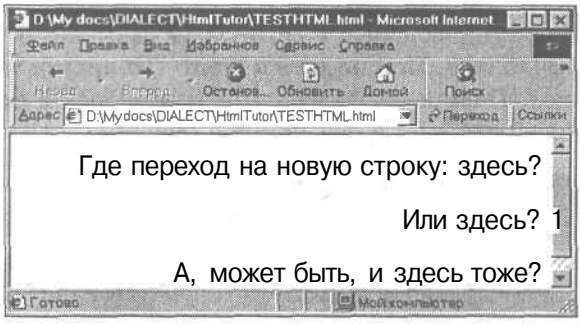


Рис. 2.7. Какой из вариантов соответствует коду, приведенному в тесте 4?

Глава 3

Служебные и "непечатные" символы

В этой главе...

- Специальные символы
- О переносах и их отсутствии

Специальные символы

Некоторые знаки, такие как < и >, являются служебными символами HTML. Другие, такие как неразрывный пробел и длинное тире, просто нельзя набрать в программе Блокнот. А на HTML-странице они должны быть — по правилам орфографии или просто для красоты.

Попробуем, например, вставить в HTML-код кавычки. В программе Блокнот (Notepad) их, конечно, нет. Но они есть в Word. Скопируем их в буфер обмена, а оттуда — в Блокнот. Посмотрим теперь, что получилось в броузере. Иногда это срабатывает. Но чаще нас ждет разочарование (рис. 3.1).

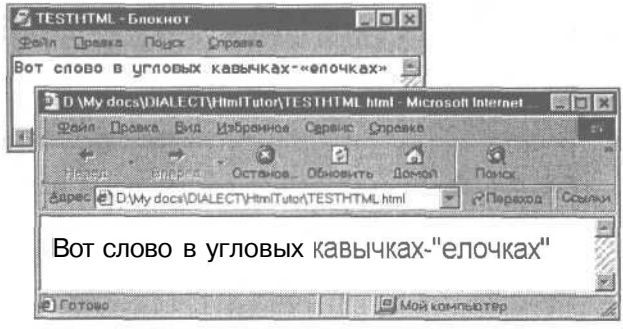


Рис. 3.1. Простой перенос специальных символов в программу Блокнот с помощью буфера обмена обычно не приводит к желаемым результатам

Чтобы обойти это досадное ограничение, "строптивые" символы заменяются в HTML-коде на соответствующие Esc-последовательности (читается "эскейп-последовательности"). Esc-последовательностью называется ASCII-код символа, перед которым стоят символы &#. Именно по этим символам броузер распознает

Esc-последовательность. Например, открывающей и закрывающей угловым скобкам соответствуют Esc-последовательности &#60 и &#62.

Для того чтобы не обращаться то и дело к таблице ASCII-кодов, некоторым наиболее часто используемым символам присвоены специальные мнемонические коды, состоящие из знака &, английского сокращенного названия этого символа и точки с запятой. Их проще запомнить, чем "голые" цифры. (Разумеется, если знать английский язык.) Например, тем же угловым скобкам соответствуют обозначения &lt (от *less than* — "меньше") и &gt (от *greater than* — "больше"), неразрывному пробелу —   (от *non-breaking space*), амперсанду (&) — &amp, а левой и правой двойным кавычкам ("елочкам") — &laquo и &raquo, соответственно.

Как видим, существует несколько видов кавычек и тире. В издательском деле — и бумажном, и электронном — приняты четкие правила их использования.

Кавычки. На клавиатуре компьютера есть только один вид кавычек — ". Точнее, это и не кавычки вовсе, а двойной штрих — знак дюйма и секунд. Именно он, а также знаки “ меньше" (<) и "больше" (>) зачастую используются в электронных публикациях вместо правой и левой кавычек. Но такое применение этих символов некорректно.

В полиграфии существует три вида кавычек: "елочки" (« »), верхние "лапки" (“ ”) и рукописные "лапки" („ “). По традиции российской полиграфии основным видом кавычек являются "елочки". Правой и левой "елочкам" в HTML соответствуют мнемонические коды « и » соответственно. Во многих изданиях вместо "елочек" используются также верхние "лапки" — это традиция западной полиграфии. Кроме того, верхние "лапки" применяются для вложенных кавычек (рис. 3.2). Левой и правой верхним "лапкам" соответствуют мнемонические коды “ и ” соответственно, а нижней левой "лапке" — „.

Дефис и тире. В полиграфии существует три знака: длинное тире, короткое тире и дефис. Из них на компьютерной клавиатуре есть только один — последний. Именно его зачастую и используют во всех случаях, когда в тексте нужно поставить дефис или тире. Однако существуют четкие правила пунктуации относительно того, какой из этих знаков когда используется.

Так, дефис ставится только внутри слов или между цифрами. Длинное тире, или просто тире (—), ставится между словами в предложении и отделяется от этих слов пробелами. Ему соответствует мнемонический код —. Короткое тире (–) ставится между цифрами без букв или между словами, набранными прописными буквами, а также используется в формулах в качестве знака "минус". Между этим знаком и словами пробел не ставится. Короткому тире соответствует мнемонический код –. Применение дефиса и тире показано на рис. 3.3.

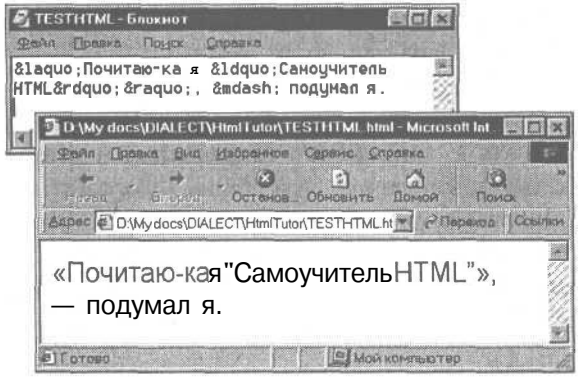


Рис. 3.2. Правильное использование кавычек: в HTML-коде указываются мнемонические имена символов « » и “ ”

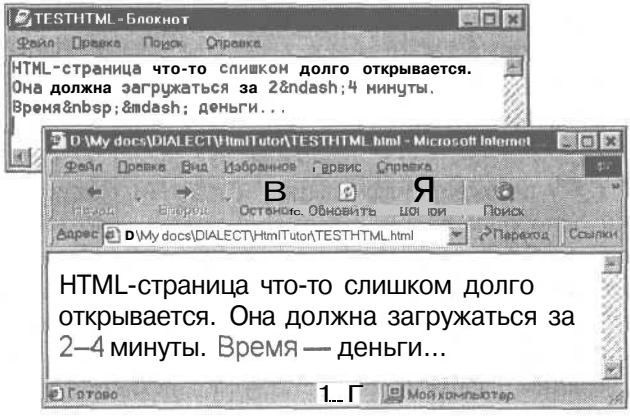


Рис. 3.3. Примеры правильного использования дефиса, длинного и короткого тире

О переносах и их отсутствии

Поскольку размер окна броузера непостоянен, переход на новую строку в абзаце происходит автоматически. При этом броузер руководствуется тем соображением, что разрывать строку можно в любом месте, где стоит пробел или дефис.

Однако, согласно правилам орфографии, существуют языковые конструкции, которые нельзя разрывать при переходе на другую строку несмотря на то, что в них используются именно эти знаки. К конструкциям с неразрывным пробелом, в частности, относятся:

фамилии с инициалами;

длинные тире с предшествующим им словом;

односложные слова с последующим словом;

цифры с последующими единицами измерения.

В этих случаях, как вы могли догадаться, в HTML-коде обычно используется неразрывный пробел (рис. 3.4).

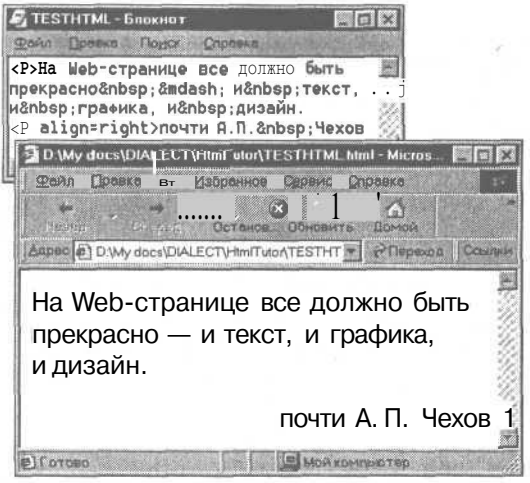


Рис. 3.4. Некоторые языковые конструкции требуют использования неразрывного пробела

В качестве примера конструкций с неразрывным дефисом можно привести диапазоны значений, сокращения и составные слова. Как быть с ними? Ведь символа "неразрывный дефис" в HTML не существует.

Зато в HTML есть дескриптор <NOBR> (от англ. *no break* — "без разрыва"). Все, что находится внутри него, должно поместиться в одной строке броузера. Другими словами, броузеру попросту запрещается переход на новую строку, пока он не встретит закрывающий дескриптор </NOBR>.

Что делает броузер, если содержимое конструкции <NOBR>... <NOBR> не помещается в одной строке? Перейти на новую строку нельзя. Остается только вывести горизонтальную полосу прокрутки (рис. 3.5).

Используя дескриптор <NOBR>, следует учитывать особенности того, как мы привыкли читать. Чем длиннее строчка, тем труднее переходить на следующую: ведь ее еще нужно найти. Именно поэтому текст на широких газетных листах разбивается на колонки. Именно поэтому аналогично поступают дизайнеры многих сайтов, особенно новостийных.

И именно поэтому рекомендуется приложить все усилия, чтобы на вашей странице никогда не появилась горизонтальная полоса прокрутки. Из-за нее не только намного труднее читается текст, но и нарушается целостный образ страницы, складывающийся в представлении ее посетителя.

Бывает так, что на вашем компьютере страница выглядела нормально, но после того, как она появилась в Internet, посетители начали жаловаться: появляется горизонтальная прокрутка, неудобно читать! В чем дело? А дело в том, что на разных компьютерах установлено разное разрешение экрана, и у разных пользователей установлен разный размер шрифта в броузере. Если вы, например, просматривали страницу при среднем размере шрифта, а кто-то предпочитает круп-

ный, то у него в строке просто помещается меньше текста. Поэтому, верстая HTML-страницу, рекомендуется проверять, как она выглядит при разных разрешениях экрана и разных размерах шрифта.

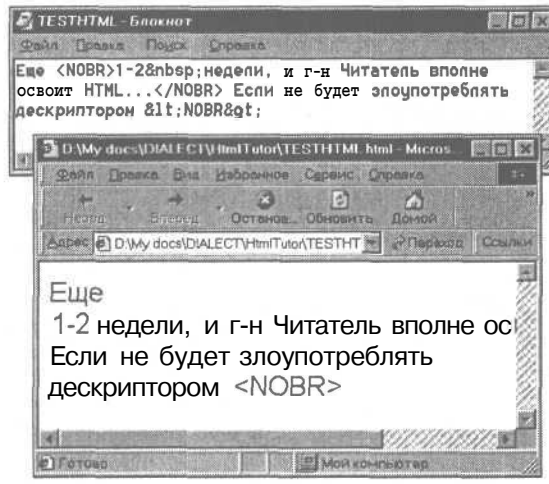


Рис. 3.5. Если конструкция <NOBR>... <NOBR> не помещается в одной строке, появляется горизонтальная полоса прокрутки

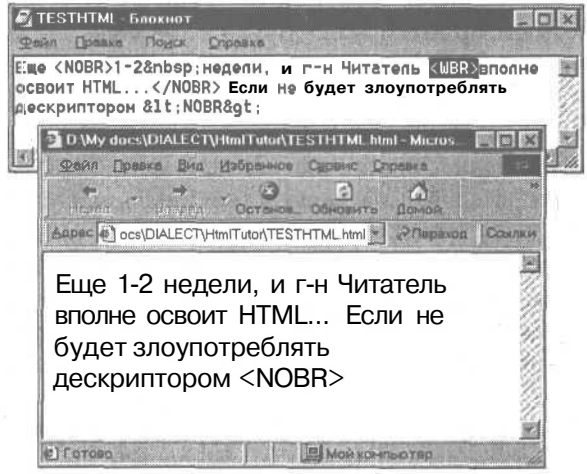


Рис. 3.6. Дескриптор <WBR> позволяет разметить места возможного перехода на новую строку внутри конструкции <NOBR>... <NOBR>

Конечно, лучший способ избежать горизонтальной прокрутки — делать конструкции <NOBR>... <NOBR> как можно короче. Но как быть, если это не получается? Например, если в тексте встретилось очень длинное слово?

В Microsoft Word и других текстовых процессорах есть специальное средство – "мягкий" перенос. Это некий символ, который вставляется в том месте слова, где его при необходимости можно разорвать. Обычно этот символ не виден. Но как только слово попадает на конец строки, в этом месте ставится дефис, и происходит перенос части слова на следующую строку.

Аналогичное средство есть и в HTML. Правда, это не символ, а дескриптор. Называется он <WBR>, от английского *word break* — "разрыв слова". Только это не совсем точно: если поставить <WBR> в середине длинного слова, перехода на новую строку в этом месте не будет. Другое дело, если поставить этот дескриптор внутри конструкции <NOBR>... <NOBR>. Собственно, там ему самое место (рис. 3.6).

Резюме

Символы, не входящие в стандартный набор ASCII, обозначаются в HTML-коде с помощью Esc-последовательностей или мнемонических имен. Esc-последовательность представляет собой ASCII-код символа, перед которым стоят знаки &#, а после — точка с запятой. Мнемоническое имя символа — это некое сокращение, образованное от его английского названия, перед которым стоит амперсанд, а после — точка с запятой. Так, левую и правую угловые кавычки (« и ») в HTML-коде можно обозначить как « и » или же как « и ». Таким же способом в HTML-код вносят служебные символы HTML. Например, если на страницу потребуется вывести угловые скобки, обычно используемые как признак дескрипторов HTML, то можно воспользоваться Esc-последовательностями < и > или же мнемоническими именами < и >.

В Web-дизайне, как и в "бумажной" полиграфии, существуют определенные традиции и правила по использованию разных видов кавычек, а также дефисов, длинных и коротких тире. Эти правила следует знать и следовать им.

Кроме того, существуют правила орфографии относительно того, в каком месте допустим переход на новую строку, а где он недопустим. Для того чтобы эти правила выполнялись на Web-странице, необходимо пользоваться символом неразрывного пробела (или), а также дескриптором <NOBR>.

Все, что находится внутри конструкции <NOBR>... </NOBR>, выводится броузером в одной строке. Если оно там не помещается, появляется горизонтальная полоса прокрутки. Последнее крайне нежелательно. Для того чтобы разметить места возможных переходов на новую строку, внутри конструкции <NOBR>... </NOBR> используются дескрипторы <WBR>.

Тесты

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?
   а) Есть три типа кавычек: «елочки», "верхние лапки" и „рукописные лапки".
   б) Есть три типа кавычек: «елочки», "верхние лапки" и ‚рукописные лапки'.

в) Есть три типа кавычек: «елочки», “верхние лапки” и „рукописные лапки”.

г) Есть три типа кавычек: &171;елочки&187;, &8220;верхние лапки&8221; и &8222;рукописные лапки&8221;.

д) Есть три типа кавычек: «елочки», “верхние лапки” и „рукописные лапки”.

е) Есть три типа кавычек: »елочки», “верхние лапки” и „рукописные лапки”.

2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) HTML-страница не должна содержать ошибок - в том числе орфографических. И ее объем не должен превышать 5-10 Кбайт.

б) HTML-страница не должна содержать ошибок ‐ в том числе орфографических. И ее объем не должен превышать 5‐10 Кбайт.

в) HTML-страница не должна содержать ошибок – в том числе орфографических. И ее объем не должен превышать 5—10 Кбайт.

г) HTML-страница не должна содержать ошибок — в том числе орфографических. И ее объем не должен превышать 5–10 Кбайт.

д) HTML-страница не должна содержать ошибок—в том числе орфографических. И ее объем не должен превышать 5 – 10 Кбайт.

3. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <P align=center> По-моему, весь этот текст должен выводиться в одной строке</P>

б) <NOBR align=center> По-моему, весь этот текст должен выводиться в одной строке</NOBR>

в) По-моему, весь этот текст должен выводиться в одной строке.

г) <NBSP> По-моему, весь этот текст должен выводиться в одной строке</NBSP>

4. В каких точках *(а, б, в,* г, *д)* следующего фрагмента возможен переход на новую строку:

<NOBR> Где пере<WBR>$^{(а)}$ход на <WBR>$^{(б)}$ новую стро-$^{(в)}$ку: здесь?<WBR>$^{(г)}$Или здесь? <P>$^{(д)}$А может быть, и здесь тоже?</NOBR>

Глава 4

Шрифты

В этой главе...

- Дескриптор <FONT>
- Гарнитура
- Размер
- Цвет
- Шрифт с несколькими параметрами
- Параметры шрифта по умолчанию

Дескриптор <FONT>

Как броузер определяет, каким шрифтом нужно выводить текст? Обычно он использует тот шрифт, который задан в его настройках по умолчанию. Но эти значения можно изменить. Больше того, страница, набранная только одним шрифтом, одного размера и одного цвета, мало кому понравится. Такую и читать неприятно, и делать неинтересно.

В текстовых процессорах эту проблему решают просто: заходят в диалоговое окно настройки шрифта и выставляют там нужные параметры — гарнитуру, размер, цвет и др. В программах с английским интерфейсом это окно обычно так и называется — *Font,* т.е. "Шрифт".

Дескриптор, отвечающий за параметры шрифта в HTML, тоже называется <FONT>. Разумеется, он парный и влияет на вид текста, заключенного внутрь конструкции `<FONT>...` </FONT>.

В отличие от текстовых процессоров, где в диалоговом окне Font можно менять многие характеристики шрифта, у дескриптора <FONT> только три параметра: гарнитура, размер и цвет.

Гарнитура

За эту характеристику — пожалуй, главную при описании шрифта — в HTML "отвечает" параметр face. Попробуем им воспользоваться (рис. 4.1).

Вообще-то, в моем броузере по умолчанию установлен Arial, <FONT face=Curier> но этот текст должен выводиться моноширинным шрифтом</FONT>... если только я ничего не путаю

Что-то здесь не так... Ах, да! Ну конечно же: шрифт называется не Curier, а Courier! Всего одна буква — а какая разница (рис. 4.2)...

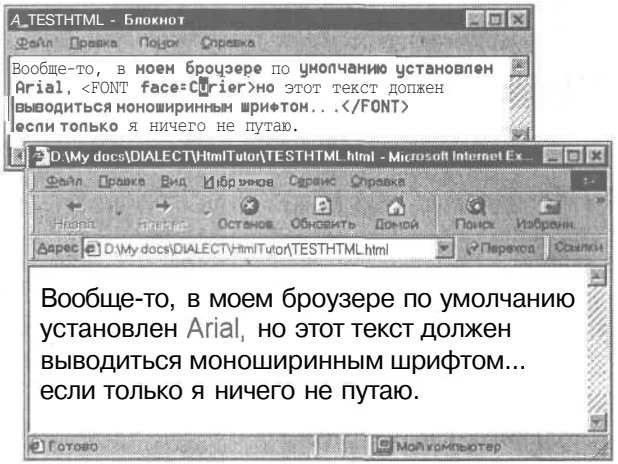


Рис. 4.1. Дескриптор <FONT> игнорирует ошибки в имени шрифта

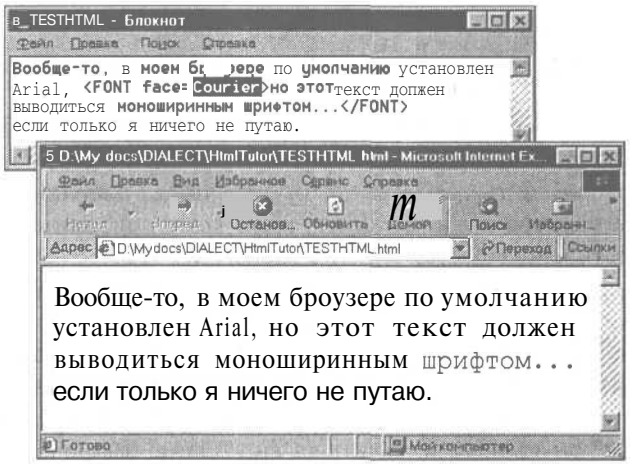


Рис. 4.2. Вот теперь все правильно

Есть существенное различие между тем, как мы воспринимаем печатный и электронный текст. Замечено, что на бумаге удобнее читать текст, набранный шрифтом с засечками, таким как Times или Baltica. С экрана же комфортнее читается шрифт без засечек, такой как Arial или Verdana.

Замечательно! Ну-ка, попробуем подставить что-нибудь еще:

```
Вообще-то, в моем броузере по умолчанию установлен Arial, <FONT face=Arial Black>
но этот текст должен выводиться шрифтом Arial Black</FONT>
```

Кажется, мы опять что-то напутали (рис. 4.3). Сверим название шрифта... Нет, здесь вроде все в порядке. Тогда где "собака зарыта"? В чем дело?

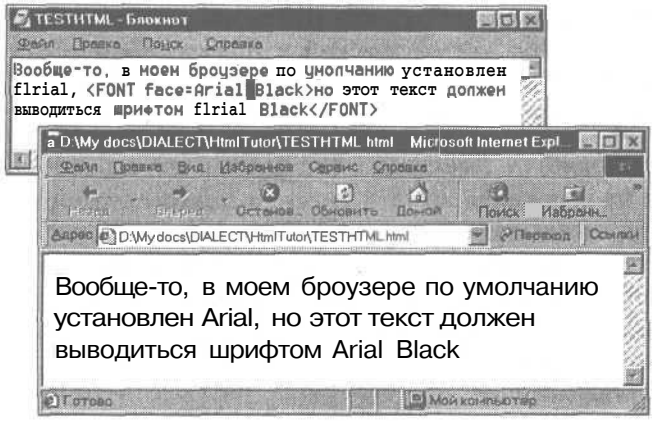


Рис. 4.3. Этот код должен работать... но почему-то не работает

В кавычках.

Пробелы, абзацы, табуляции внутри дескриптора броузер воспринимает как разделители между названием дескриптора и параметрами. Поэтому если значение параметра содержит пробелы, оно заключается в кавычки. Именно так следует поступить в нашем случае, так как название шрифта — Arial Black — состоит из двух слов:

```
Вообще-то, в моем броузере по умолчанию установлен Arial, <FONT face="Arial
Black"> но этот текст должен выводиться шрифтом Arial Black</FONT>
```

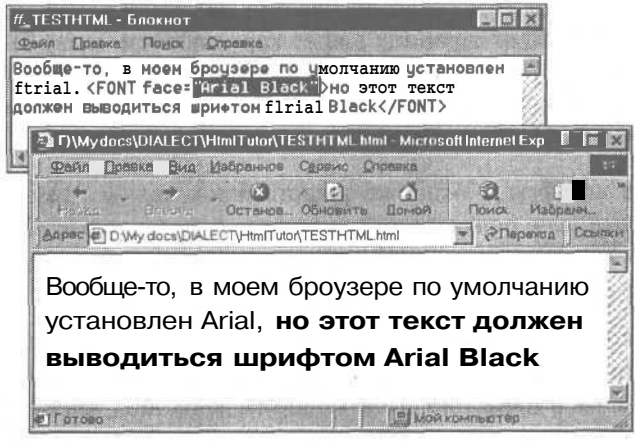


Рис. 4.4. Если значение параметра содержит пробел, его нужно заключить в кавычки

Вот теперь все работает правильно (рис. 4.4).

В Web-дизайне есть правило, которое можно кратко сформулировать так: много кавычек не бывает. Другими словами, если вы не уверены, нужно ли заключать значение параметра в кавычки, смело ставьте их. Если забыть поставить кавычки в нужном месте, броузер отобразит

страницу с ошибками, а если поставить их лишний раз — ничего не произойдет. При автоматической и полуавтоматической разметке Web-страниц в HTML-редакторах все значения параметров заключаются в кавычки. В общем, экономить на кавычках не стоит.

При задании атрибута face следует помнить, что шрифт не сохраняется в HTML-файле и не передается по сети вместе с текстом страницы. И нет никакой гарантии, что на компьютере посетителя будут все необходимые шрифты. Как тогда поступает броузер? Просто использует шрифт, установленный по умолчанию. Поэтому если вы использовали подобным образом какой-нибудь экзотический шрифт, желая добиться определенного визуального эффекта, то пользователь, скорее всего, ваших стараний не оценит.

Как выйти из положения?

Прежде всего, параметру face можно присвоить в качестве значения не один шрифт, а целый список из нескольких похожих шрифтов. Авось у пользователя найдется хоть один из них. Поскольку броузер просматривает список слева направо, то в начале списка обычно указывают наиболее подходящий шрифт, а в конце — такой, который наверняка должен быть в наличии:

```
<FONT face="Cotton, Impact, Arial"> Какой-нибудь из этих шрифтов наверняка должен быть на компьютере посетителя моей странички </FONT>
```

Вообще-то, многие Web-дизайнеры не жалуют параметр face именно за то, что результат его применения сильно зависит от наличия у пользователя нужного шрифта. Если вид надписи сильно зависит от гарнитуры, а она ну очень экзотическая, то надпись делают в графическом редакторе и помещают на Web-страницу в виде картинки (см. главу 8). Если же гарнитура не очень существенна, то вместо конструкции <FONT face=...> многие предпочитают использовать дескрипторы логического форматирования (см. главу 6).

Многие шрифты в Windows представлены целыми семействами: отдельный файл для полужирного шрифта, отдельный — для курсива и т.д. Можно ли, например, написать так:

<FONT face="Arial Italic"> Это курсив? </FONT>

Нет, нельзя. В качестве значения параметра face следует использовать не имена файлов со шрифтами и не те имена, под которыми эти шрифты указаны в окне шрифтов Windows, а те, под которыми они фигурируют в броузере. (См. также раздел "Другие параметры".)

Размер

Не знаю, как у кого, но со мной при первом знакомстве с правилами задания шрифтов в HTML случился легкий шок... Почему? Сейчас поймете.

Как задается размер шрифта в текстовом процессоре? Очень просто, в пунктах. Неважно, что в лучшем случае один из сотни пользователей знает, что такое

пункт. Главное, что даже новичок уже через десять минут ориентируется в этих размерах и может выбрать нужный.

В дескрипторе <FONT> также можно задать размер шрифта с помощью параметра size. Но как! Не в пунктах и не в миллиметрах, а в неких безымянных единицах от 1 до 7. Это напоминает размеры одежды: все знают, что L больше, чем M, и меньше, чем XL — и только. Так и в HTML, size =1 — самый мелкий шрифт, size=7 — самый крупный. Ничего более определенного сказать нельзя.

Наконец, для того чтобы завершить картину хаоса, напомним, что размер шрифта зависит еще и от желания пользователя. Обычно в броузерах предусмотрено несколько вариантов отображения текста. Конечно, это удобно: для того чтобы сэкономить бумагу при печати странички, выставляем самый мелкий шрифт, а для того чтобы читать с экрана длинный текст, — шрифт покрупнее. Но какой после этого смысл в параметре size (рис. 4.5)?

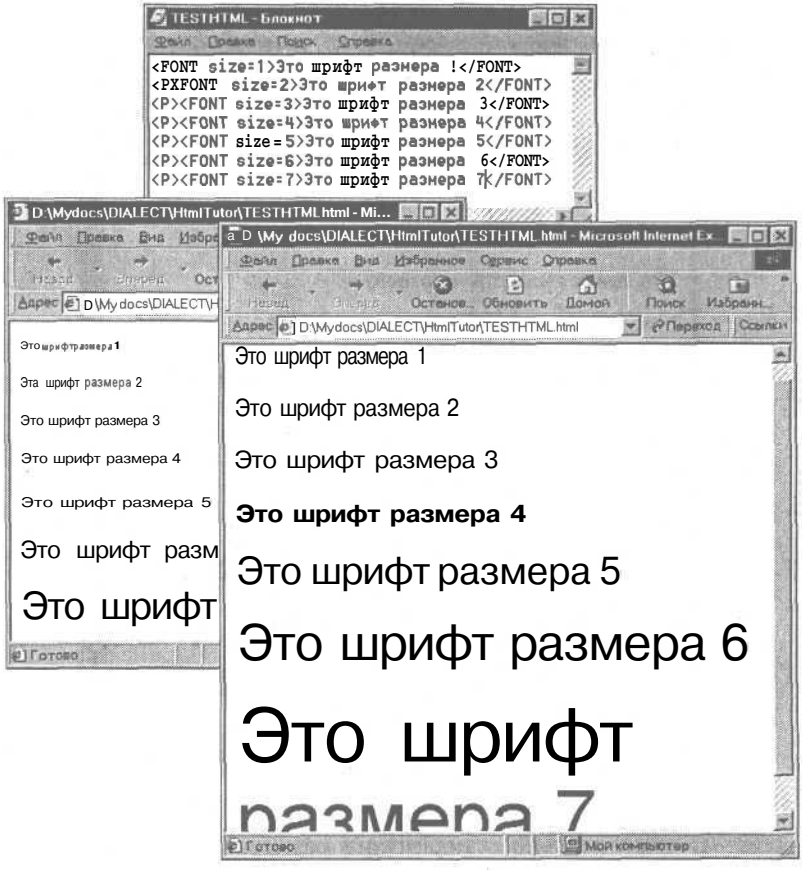


Рис. 4.5. Код одинаковый, а размер шрифта разный. Так получается потому, что параметр size определяет размер шрифта относительно того, который установлен в броузере

Что ж, наведем некоторый порядок. Очевидно, указывать абсолютный размер шрифта, например <FONT size=2>, имеет смысл далеко не всегда. Гораздо перспективнее выглядит возможность указать *относительный* размер — относительно того, что установлен по умолчанию. Для этого нужно перед цифрой поставить знак. Так, например, <FONT size=-2> соответствует шрифту, который меньше текущего на две единицы, а <FONT size=+3> — большему текущего на три единицы (рис. 4.6). Кстати, с помощью такого кода можно без труда убедиться, что по умолчанию шрифт броузера соответствует size=3.

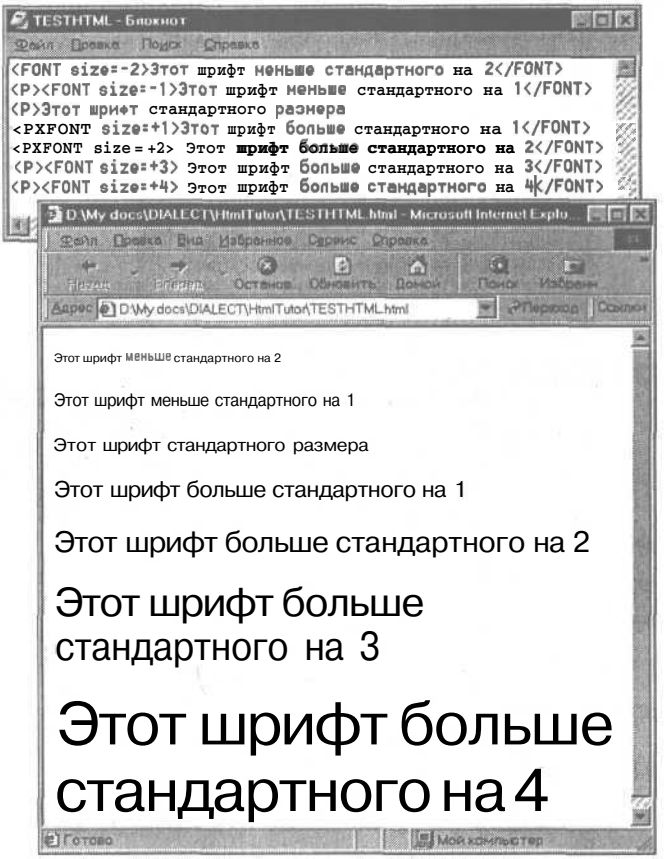


Рис. 4.6. Такой код позволяет убедиться, что по умолчанию шрифт броузера соответствует size=3

А что будет, если приращение слишком большое? Например, <FONT size=+5> или <FONT size=-4> относительно стандартного размера? Попробуем и посмотрим. Ничего особенного: просто броузер "округляет" значение до ближайшего приемлемого, т.е. до +4 и -3, соответственно (рис. 4.7).

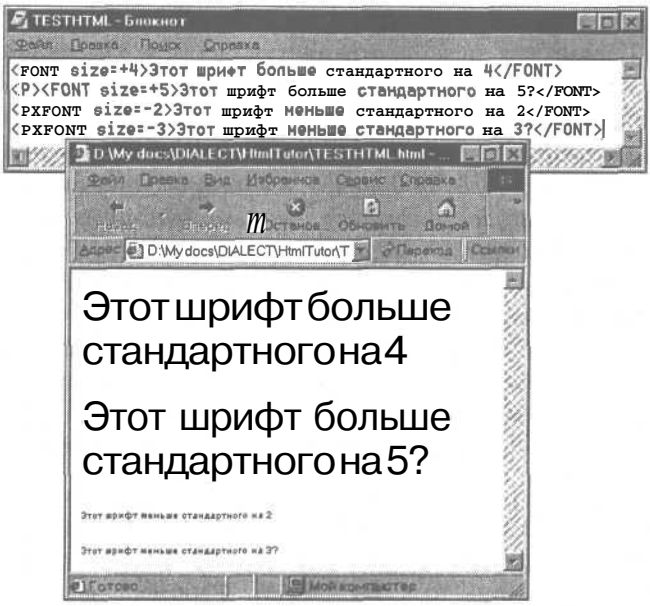


Рис. 4.7. Если значение параметра `size` превышает допустимый диапазон, то оно "округляется" до ближайшего приемлемого значения

Цвет

По умолчанию текст Web-страницы выводится черным по белому. Так проще, так привычнее, так легче читать. Но ведь очень часто хочется сделать иначе: что-то сделать более ярким, где-то изменить цвет "под настроение".

Для того чтобы изменить цвет текста, в дескрипторе <FONT> предусмотрен параметр color:

```
<FONT color=... >
```

Стоп.

Как задать цвет в текстовом HTML-коде?

Представьте себе ситуацию: вы беседуете по телефону с марсианином и хотите описать ему восход солнца над морем. А тот понятия не имеет, как выглядит солнце над нашей планетой, какого цвета утренняя морская вода и небо над ней. И вообще, у него глаза устроены иначе. Поэтому вместо красивых слов "нежно-голубой", "изумрудный", "золотистый" придется диктовать длины электромагнитных волн...

С броузером — примерно такая же история. Каждый цвет в HTML-коде обозначается шестнадцатеричным числом, перед которым стоит знак #, например:

```
<FONT color=#66CDAA> Текст цвета морской волны </FONT>
```

Чем не "марсианский" язык?

Впрочем, разобраться в этом "языке" не так сложно, как кажется на первый взгляд. Цвет пикселя на экране монитора составляется так же, как и в телевизоре —

из красной, зеленой и синей точек разной яркости. Именно яркость этих трех точек задается кодом. Первые два знака соответствуют яркости красной точки, вторые - зеленой, третьи — синей. Поэтому формат цвета в HTML часто обозначают как #RRGGBB (от английских слов *Red* — красный, *Green* — зеленый и *Blue* — синий). Яркость каждой составляющей измеряется целым числом, которое в десятичной системе исчисления находится в пределах от 0 до 255, а в шестнадцатеричной — от 00 до FF.

Даже если вы когда-нибудь занимались программированием, вряд ли вам понравится подбирать цвета "на ощупь", меняя цифры и просматривая результат в броузере. Тем более вам не захочется заучивать на память все 16 миллионов цветов, которые можно закодировать таким образом. А точнее, даже больше: $16^6 = 16777216$.

Но это и не нужно. Многие цвета, кроме кода, имеют собственные названия. Это шестнадцать основных цветов и еще 124 дополнительных, которыми тоже вполне можно пользоваться. Например, наш "цвет морской волны" называется mediumaquamarine. Пожалуй, этого достаточно, чтобы удовлетворить далеко не самые скромные запросы.

Если же понадобится что-то совсем необычное, можно воспользоваться одним простым приемом. Заключается он в следующем. В любом графическом редакторе, будь то Corel DRAW, Photoshop или обычный Windows Paint, есть "палитра" для смешивания цветов. Из нее можно узнать интенсивность трех составляющих любого цвета. Правда, в десятичной форме. Для того чтобы получить их шестнадцатеричные значения, можно воспользоваться калькулятором Windows, переведя его в "инженерный" режим.

Проведем небольшой эксперимент. Проверим, действительно ли цвету mediumaquamarine соответствует код 66CDAA. Для этого выполним следующие действия.

1. Создадим HTML-страничку с таким кодом:

```
<FONT color=mediumaquamarine size=5><B> Проверка цвета: mediumaquamarine
</B></FONT>
```

2. Запустим Windows Paint и откроем нашу страничку в броузере.
3. Скопируем окно броузера в буфер обмена с помощью комбинации клавиш <Alt + Print Screen> и поместим его в Paint с помощью комбинации клавиш <Ctrl+Ins> или <Ctrl+V>.
4. Выберем инструмент "пипетка" и щелкнем на интересующем нас цвете.
5. Выберем команду Палитра о Изменить палитру и в появившемся диалоговом окне щелкнем на кнопке Определить цвет. А вот и то, что нам нужно, интенсивности составляющих: красный — 102, зеленый — 205, синий — 170.
6. Запускаем калькулятор и с помощью команды Вид ⇨ Инженерный переводим его в расширенный режим.
7. Вводим интенсивность, например красного цвета (она у нас равна 102), и переходим на шестнадцатеричное исчисление (режим Hex). Что написано в "окошечке"? Правильно, 66.

8. Возвращаемся в десятеричный режим (Dec) и повторяем то же для зеленой и синей составляющих. Получаем CD и AA соответственно. Сошлось?

   Или не сошлось?

   Вспомним: код цвета определяется из расчета, что всего у нас больше 16 миллионов цветов. Это соответствует режиму монитора True Color. Если режим другой, то цвет, указанный в HTML-коде, отсутствует в палитре монитора, и броузер автоматически заменяет его другим, ближайшим. Отсюда и возможное несоответствие "измеренного" и закодированного цветов.

   Естественно, чем меньше цветов отображает монитор, тем заметнее подобные несоответствия. Как же теперь быть: ведь у разных посетителей сайта разные мониторы?

   Как правило, различия не настолько существенны, чтобы серьезно влиять на качество дизайна и тем более на интерес к странице. А для тех случаев, когда это все-таки важно, существует специальная таблица из 216 "рекомендованных" цветов, разработанная компанией Netscape, — так называемая "палитра Netscape". Эти цвета одинаково отображаются во всех режимах монитора — от 256-цветного до True Color.

Шрифт с несколькими параметрами

А как описать в HTML-коде шрифт, который отличается от стандартного и цветом, и размером, и гарнитурой? Если речь идет об одном фрагменте, то, разумеется, используется дескриптор <FONT> с несколькими параметрами:

```
<FONT face=Courier size=+2> Текст крупным шрифтом Courier</FONT>
```

Порядок перечисления параметров значения не имеет. С тем же успехом мы могли бы написать:

```
<FONT size=+2 face=Courier> Текст крупным шрифтом Courier</FONT>
```

Помните, что разделителями между параметрами дескриптора HTML служат пробел, табуляция или символ перехода на новую строку. Разделение параметров запятыми или точками с запятой (как это делают по привычке программисты) приведет к неверному отображению страницы в броузере.

Если же разные элементы форматирования распространяются на различные фрагменты текста, то можно воспользоваться вложенными дескрипторами <FONT>. Предположим, нам нужно разметить текст следующим образом: выделить фразу моноширинным шрифтом, в нем несколько слов увеличить на две единицы, и еще некоторые буквы, например, те, на которые падает ударение, вывести другим, более ярким цветом. Тогда код будет выглядеть так:

```
<FONT color=red>Э</FONT>то об<FONT color=red>ы</FONT>чный текст. <FONT
face=Courier>А <FONT color=red>э</FONT>тот текст напеч<FONT color=red>а</FONT>тан
шр<FONT color=red>и</FONT>фтом <FONT size=+2>Couri<FONT
color=red>e</FONT>r</FONT></FONT>
```

Помните, что каждый вложенный дескриптор нуждается в собственном закрывающем дескрипторе, даже если эти дескрипторы одинаковые. Нельзя "закрыть" несколько дескрипторов <FONT> одним </FONT>. Код наподобие

```
Это обычный текст. <FONT face=Courier>А этот текст напечатан шрифтом
<FONT size=+2>Courier</FONT>
```

приведет к тому, что внутренний дескриптор <FONT size=+2> будет закрыт, а внешний <FONT face=Courier> — нет, и весь остальной текст на странице будет выводиться шрифтом Courier.

Обратите внимание, что дескриптор можно размещать в середине слова. В результате слово не разобьется на части, если только не поставить справа или слева от дескриптора пробелы. Пробелы внутри дескриптора не влияют на слитное или раздельное написание слова.

Другие параметры

Как вы уже могли заметить, дескриптор <FONT> невыгодно отличается от соответствующей функции текстовых редакторов своими спартанскими, чтобы не сказать попросту бедными, возможностями.

Некоторые из таких возможностей восполняются с помощью специальных дескрипторов. С тремя такими дескрипторами — <B>, <I> и <U> — вы уже знакомы.

Для зачеркнутого текста используются дескрипторы <STRIKE> и <S>. Их название произошло от английского слова *strikethrough*, что в переводе означает... правильно, зачеркивание (ну никакой фантазии, правда?). Оба дескриптора реализуют одно действие: вывод текста, перечеркнутого одинарной горизонтальной чертой (рис. 4.8).

Рис. 4.8. Для отображения перечеркнутого текста используются дескрипторы <STRIKE> и <S>

Зачем нужны два одинаковых дескриптора с разными названиями?

Здесь все до смешного просто. Стандарт HTML, как любой живой язык, — неважно, компьютерный или разговорный — развивается. То, что в процессе использования оказалось неудобным, изменяется, разные части "притираются" друг к другу. Вспомните: то, что раньше называли "кинематографом", сейчас зовут просто "кино", вместо длинного "таксомотор" говорят просто "такси". Если вы сегодня скажете приятелю, что посетили кинематограф и на обратном пути взяли таксомотор, он, возможно, удивится, но вас поймет. То же и в HTML: вначале был придуман длинный дескриптор <STRIKE>. Потом его название показалось чересчур длинным, и его сократили до **<S>.** Но если вы напишете в коде <STRIKE>, броузер вас поймет... И даже не удивится.

А как реализовать на Web-странице другие спецэффекты —объемные буквы, мигающий текст, "малые прописные"? К сожалению, многое из того, что в текстовых редакторах обычно сосредоточено в одном диалоговом окне, в HTML относится к разным элементам языка. В частности, многое вы найдете в каскадных таблицах стилей (см. главу 15). А если чего-то не окажется и там, придется восполнять пробелы одного языка — HTML — возможностями другого — JavaScript (см. главу 16). Наконец, если все эти средства не обеспечат нужного эффекта, остается одно: нарисовать желаемое в графическом редакторе и вставить надпись на Web-страницу в виде картинки.

Параметры шрифта по умолчанию

Бывают случаи, когда нужно изменить шрифт больших массивов текста, например одного или нескольких абзацев. В таких случаях вместо дескриптора <FONT> можно использовать дескриптор <BASEFONT>.

Дескриптор <BASEFONT> определяет параметры шрифта по умолчанию. В отличие от <FONT>, дескриптор <BASEFONT> не является контейнерным и не имеет закрывающего дескриптора: его действие распространяется до конца Web-страницы или до следующего дескриптора <BASEFONT> (рис. 4.9). Поэтому, если <BASEFONT> определяет параметры шрифта для всей страницы, его часто выносят в заголовок (см. главу 14).

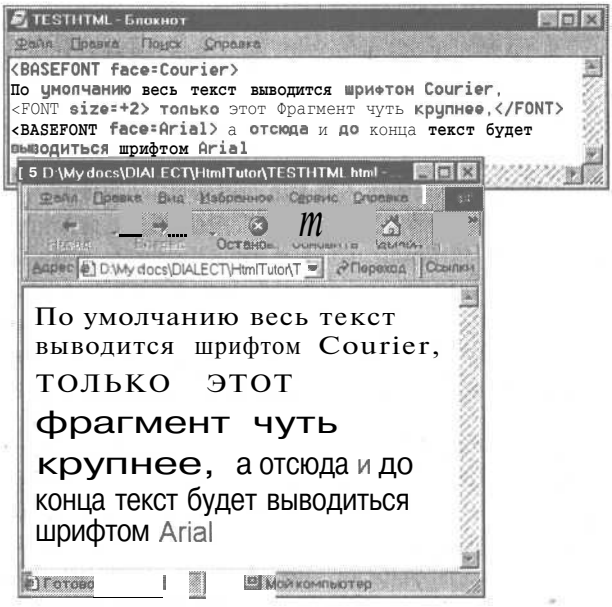


Рис. 4.9. Использование дескрипторов <FONT> и <BASEFONT>

Резюме

Параметры шрифта — гарнитура, размер, цвет — в HTML задаются с помощью дескриптора <FONT>. Это контейнерный дескриптор, определяющий свойства заключенного в него текста.

Гарнитура задается с помощью параметра face. Кроме собственно имени шрифта, значением face может служить список имен шрифтов. В этом случае броузер выбирает первый из списка шрифт, который есть на компьютере пользователя. Если же такого шрифта нет, параметр face игнорируется. Поэтому рекомендуется все надписи с нестандартной гарнитурой делать в графическом редакторе и вставлять в HTML-страницу в виде изображений (см. главу 8).

Размер шрифта определяется параметром size и выбирается из ряда от 1 до 7. По умолчанию используется шрифт размера 3. Реальный размер текста в окне броузера зависит от размера шрифта, установленного пользователем для просмотра страниц. Поэтому чаще в дескрипторе <FONT> задается относительный размер шрифта. Так, например, запись <FONT size=+2> означает, что дальше будет использоваться шрифт на 2 единицы крупнее предыдущего.

Цвет шрифта определяется параметром color и задается по формуле #RRGGBB, где RR, GG и BB — двузначные шестнадцатеричные числа, обозначающие интенсивность красной, зеленой и синей составляющих, соответственно. Кроме того, для стандартных 140 цветов существуют мнемонические имена, которые также можно использовать при определении параметра color.

Если необходимо указать несколько параметров дескриптора <FONT>, то они перечисляются в произвольном порядке через пробел, например: <FONT color=red size=+3>. Кроме того, дескрипторы <FONT> могут быть вложенными.

Для того чтобы описать форматирование больших массивов текста, удобно использовать дескриптор <BASEFONT>. Это одиночный дескриптор, имеющий те же параметры, что и <FONT> и определяющий вид текста до конца страницы или до следующего дескриптора <BASEFONT>.

Тесты

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?
   - **а)** <FONT> Этот текст набран другим шрифтом</FONT>
   - **б)** <FONT face=Arial Cyr> Этот текст набран другим шрифтом</FONT>
   - **в)** <FONT face="Arial Cyr"> Этот текст набран другим шрифтом</FONT>
   - **г)** <FONT face="ARIAL CYR"> Этот текст набран другим шрифтом</FONT>
   - д) <FONT face="Arial Bold"> Этот текст набран другим шрифтом</FONT>
2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?
   - **а)** <FONT size=3> Этот текст набран другим размером </FONT>
   - **б)** <FONT size=8 pt> Этот текст набран другим размером </FONT>
   - **г)** <FONT size=-3> Этот текст набран другим размером </FONT>
   - д) <FONT size=+7> Этот текст набран другим размером </FONT>
3. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?
   - **а)** <FONT color=#123456> Этот текст набран другим цветом</FONT>
   - **б)** <FONT color=#&123456> Этот текст набран другим цветом</FONT>
   - **в)** <FONT color=#ABCDEF> Этот текст набран другим цветом</FONT>
   - **г)** <FONT color=#GHIKLM> Этот текст набран другим цветом</FONT>
   - **д)** <FONT color=#red> Этот текст набран другим цветом</FONT>
   - **е)** <FONT color=#&red> Этот текст набран другим цветом</FONT>
   - **ж)** <FONT color=red> Этот текст набран другим цветом</FONT>
   - **з)** <FONT color-red> Этот текст набран другим цветом</FONT>
4. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?
   - **а)** Этот текст набран мелким шрифтом и красным цветом
   - **б)** Этот текст набран красным цветом и мелким шрифтом

в) <color=red font size=1> Этот текст набран красным цветом и мелким **шрифтом**

г) **<font="red, 1">** Этот текст набран красным цветом и **мелким шрифтом**

д) Этот текст набран красным цветом и мелким шрифтом**

е) Этот текст набран красным цветом и мелким шрифтом**

ж) <font color=red> Этот текст набран красным цветом и <font **size=1>мелким шрифтом</font** size>

5. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) **<BASEFONT** color=blue> Синий текст <FONT **color=yellow>** Зеленый текст </FONT>

б) <BASEFONT color=blue> Синий текст **<FONT** color=yellow> Желтый текст </FONT>

в) <BASEFONT color=blue> Синий текст </BASEFONT>

Глава 5

Заголовки

В этой главе...

- Уровни заголовков
- Параметры заголовка

Уровни заголовков

Во многих текстовых процессорах есть такая функция: если некоторый текст служит заголовком, его можно выделить соответствующим образом, использовав для этого один из специально предусмотренных стилей. Вместо того чтобы каждый раз выполнять одни и те же действия — выделять текст более крупным и жирным шрифтом, выравнивать его по центру страницы и т.п., — достаточно присвоить этому фрагменту стиль соответствующего заголовка.

В HTML такая возможность тоже есть. Здесь предусмотрено 6 уровней заготовков. Первый уровень соответствует самому крупному заголовку, шестой — самому мелкому. Для разметки заголовков используются дескрипторы вида <Нл>, где н — первая буква английского слова *header* (заголовок), а л — номер заголовка, от 1 до 6. Каждый заголовок выводится шрифтом своего размера и начинается с новой строки. Дескрипторы заголовков являются парными: текст заголовка помещается внутри конструкции <Нл>... </Нл> (рис. 5.1).

Как и большинство парных дескрипторов HTML, дескрипторы заголовков нужно закрывать. Если этого не сделать, возможны самые неожиданные и, как правило, нежелательные эффекты.

Зато ставить дескриптор абзаца для перехода на новую строку не нужно: броузер сам понимает, что заголовки — такая вещь, которая всегда начинается с новой строки и после которой опять нужно перейти на новую строку. Вот вам преимущества логической разметки!

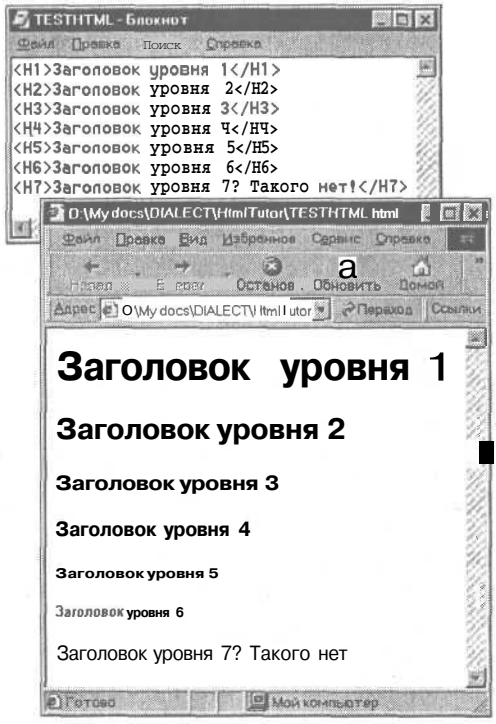


Рис. 5.1. В HTML предусмотрено 6 уровней заголовков

Вообще-то, заголовки — вещь важная и нужная. В первую очередь для того, чтобы текст проще читался. Сравните сами, что приятнее глазу: страница или экран со сплошным "слепым" текстом, без единого цветного пятна или крупной буквы, где буквально не за что "зацепиться"? Или текст, разбитый на небольшие, логически законченные части с помощью заголовков разного уровня? Вопрос, по-моему, риторический. Поэтому, если страница содержит большие массивы текста, рекомендуется разбивать их с помощью заголовков так, чтобы при любом положении на экране находились 1—3 заголовка.

Но не кажется ли вам, что уровней заголовков многовато? В книгах и электронных публикациях такой глубокой вложенности не встретишь. Действительно: когда многочисленные заголовки вложены друг в друга, как матрешки, в них трудно разобраться. Вместо того чтобы вносить ясность, многочисленные мелкие заголовки только запутывают читателя. Обычно на Web-страницах используют 1-2 уровня заголовков, хотя в отдельных случаях, — например, если речь идет о большой книге, — вложенность может доходить до четырех уровней.

Но в HTML таких уровней шесть! Для чего они?

Запас никогда не помешает. Например, для того, чтобы использовать заголовки не подряд, а через один. Скажем, если нам нужна двухуровневая вложенность, то мы вовсе не обязаны использовать обязательно заголовки <H1> и <H2>. Вместо этого можно воспользоваться, например, <H2> и <H4> или любыми другими, и таким образом подобрать нужное соотношение размеров.

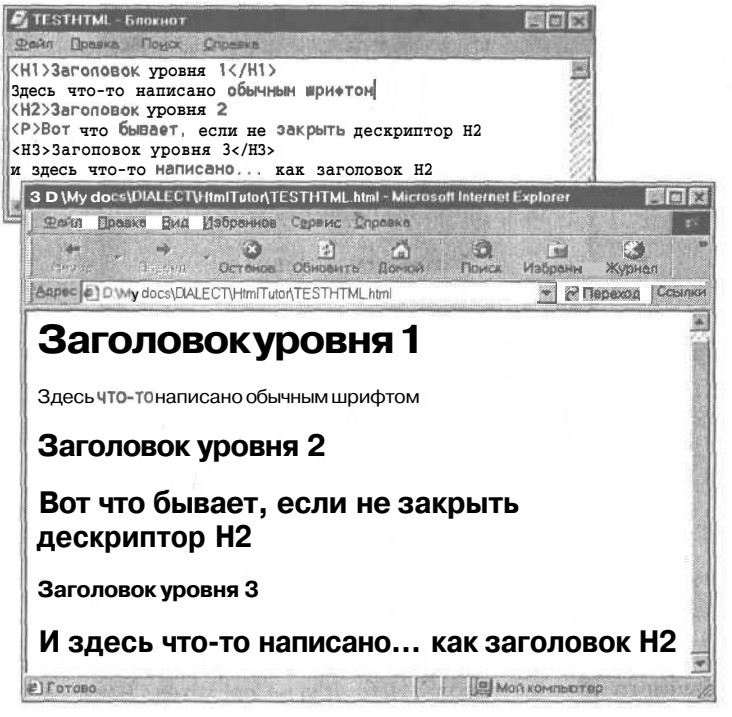


Рис. 5.2. Если не закрыть дескриптор заголовка, возможны самые неожиданные эффекты. Но если он закрыт, следующий текст всегда начинается с новой строки

Параметры заголовка

Итак, у нас есть средство для разметки заголовков и подзаголовков страницы. Но не слишком ли оно однообразно? В конце концов, это всего лишь вариации на тему размера шрифта в пределах одного абзаца.

Есть ли у дескрипторов <Нл> параметры?

Разумеется. И это те же параметры, что и у дескриптора <P>. Точнее, один параметр абзаца — выравнивание (align). Как и обычные абзацы, заголовки по умолчанию выравниваются по левому краю. Но с помощью параметра align их можно выровнять по правому краю или по центру (рис. 5.3).

А как же с другими параметрами? Ведь заголовки часто отличаются не только размером шрифта и выравниванием, но и цветом, и гарнитурой, да мало ли еще чем?

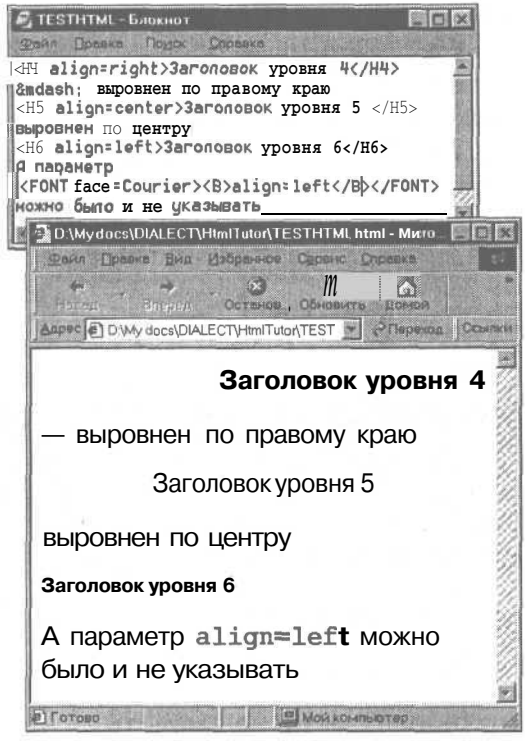


Рис. 5.3. К дескрипторам заголовков применим тот же параметр align, что и к обычным абзацам

Что ж, ответ на этот вопрос такой же, что и на вопрос об аналогичном форматировании обычного абзаца: используйте другие средства. Благо в HTML их хватает. Например, средства уже известного вам дескриптора <FONT>. Или же средства пока неизвестных вам каскадных таблиц стилей (см. главу 15). Наконец, если заголовки представляют собой нечто уж вовсе экзотическое, можно заменить текст рисунком (см. главу 8).

Однако есть еще один аргумент в пользу применения дескрипторов <Нл>, а не, например, графики или обычного текста. Текст, заключенный внутри заголовочной конструкции, пользуется особым вниманием поисковых серверов. Именно по нему они определяют тему страницы и принимают решение о помещении ее в тот или иной раздел электронных каталогов. Подробнее об этом вы узнаете из главы 14.

Резюме

Для выделения заголовков более крупным и жирным шрифтом в HTML используются дескрипторы <Нл>, где л — цифра от 1 до 6. Заголовок 1 уровня выводится самым крупным шрифтом, заголовок 6 уровня — самым мелким.

Несмотря на наличие шести уровней заголовков, для хорошей структуризации страницы обычно достаточно двух-трех. Более глубокая вложенность часто лишь мешает читателю составить четкое представление *о* структуре документа.

Заголовки почти во всем похожи на абзацы, размечаемые с помощью дескриптора <P>: сами они и следующий за ними текст всегда начинается с новой строки, а для более точного форматирования используется тот же параметр, что и у дескриптора <P> — align. Однако, в отличие от дескриптора <P>, в заголовках предусмотрены только три значения этого параметра — left, right и center. Остальные параметры текста, как и в обычном абзаце, определяются другими средствами, в частности с помощью дескриптора <FONT>.

Кроме того, дескрипторы заголовков, в отличие от дескрипторов абзаца <P>, обязательно нужно закрывать. Иначе весь текст до конца страницы по умолчанию будет считаться заголовком соответствующего уровня.

Тесты

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?
   а) <H1> Это заголовок первого уровня
   б) <H1> Это заголовок первого уровня </H1>
   в) <H1> Это заголовок первого уровня </H2>
2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?
   а) <H1> Это заголовок
 из двух строк </H1>
   б) <H1> Это заголовок <P> из двух строк </H1>
   в) <H1> Это заголовок <H2> из двух строк </H2x/H1>
   г) <H1> Это заголовок </H1><H2> из двух строк </H2>
3. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?
   а) <H1>Это заголовок, выровненный по левому краю</H1>
   б) <Hl align=left>Это заголовок, выровненный по левому краю</H1>
   в) <Hl align=right>Это заголовок, выровненный по правому краю</H1>
   г) <H1 align=center>Это заголовок, выровненный по центру</H1>
   д) <H1 align=justify>Это заголовок, выровненный по ширине</H1>

Глава 6

Логическая разметка гипертекста

В этой главе...

- "Логические" дескрипторы
- Расстановка акцентов
- Цитаты
- 4 Верхние и нижние индексы
- Имитация бурной деятельности
- Сокращения
- Обратный адрес
- Зачем все это?
- HTML придумал программист...

"Логические" дескрипторы

Откровенно говоря, понятия "логической" и "физической" разметки Web-страницы довольно условны. Говоря так, противопоставляют жесткую привязку к "физическим" особенностям страницы выделению на ней логически цельных элементов.

Точный вид этих элементов, соответствующим образом размеченных в коде Web-страницы, на экране зависит от броузера, установленного у посетителя страницы. Но от него, похоже, и так зависит слишком много... Так не лучше ли окончательно положиться на то, что броузер так или иначе выделит нужные нам элементы текста, и не пытаться навязать ему свое видение?

Собственно, с дескрипторами обоих типов вы уже знакомы. Попробуйте, не читая следующую фразу, отнести уже известные вам дескрипторы к той или иной категории. Можете даже составить два списка — "физический" и "логический". Готово? Тогда давайте сравним.

Действительно, дескрипторы, "отвечающие" только за внешний вид текста, — <U>, <B>, **<I>,** <FONT>, <P>, <BR>, <WBR>, <NOBR>, — относятся к средствам "физической" разметки. Однако дескрипторы заголовков однозначно попадают в разряд "логических": ведь они не столько описывают видимые свойства текста, сколько обозначают его роль на странице.

Действительно, до сих пор мы в основном имели дело с дескрипторами "физического" типа. Восполним этот пробел и рассмотрим другие логические дескрипторы языка HTML.

Расстановка акцентов

Что мы делаем, когда хотим обратить внимание читателя на что-то важное? Правильно, выделяем важное курсивом. Или полужирным. Или подчеркиваем. Или еще как-нибудь.

Но страница, где встречается сразу несколько вариантов выделения, — и подчеркивание, и курсив, и еще что-нибудь — выглядит, мягко говоря, любительски, чтобы не сказать, — неопрятно. Избежать беспорядка будет гораздо проще, если вместо разнообразных вариантов выделения использовать логическую разметку.

Для этого в HTML предусмотрено несколько средств (рис. 6.1).

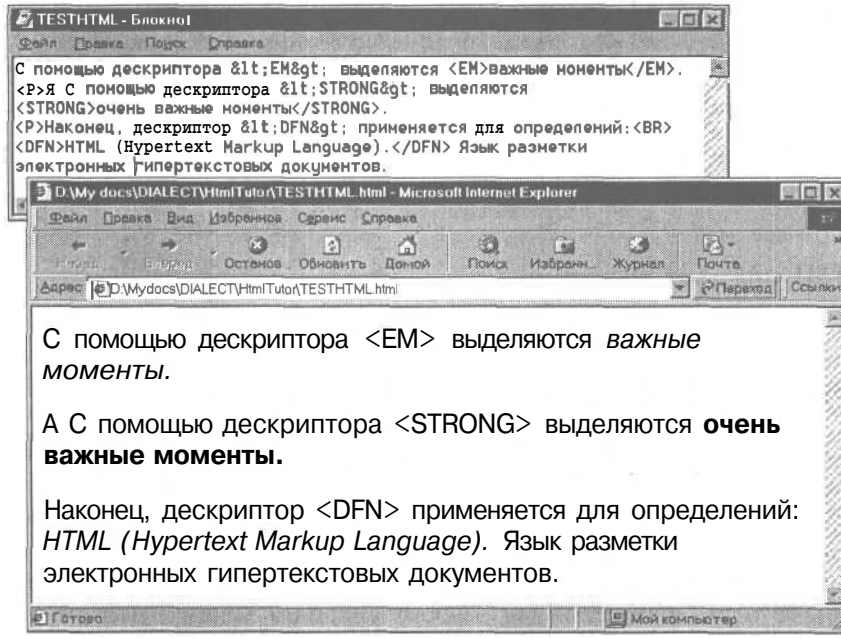


Рис. 6.1. Варианты выделения важных фрагментов текста с помощью логических дескрипторов HTML

Как поступают в книгах, когда хотят сакцентировать внимание читателя на нескольких важных словах? Правильно, выделяют эти слова курсивом. Для этого в HTML есть "физический" дескриптор <I>. Но, кроме него, специально для выделения важных моментов имеется дескриптор <EM>. Его название происходит от английского слова *EMphasis*— постановка ударения, выделение, подчеркивание. Дескриптор <EM> является парным. Все, что заключено внутри конструкции <EM>... </EM>, обычно выводится курсивом.

Действительно, из всего прочитанного текста в первую очередь запоминается то, что было напечатано курсивом (разумеется, при условии, что остальной текст был обычным). А как сделать, чтобы какое-то слово или несколько слов буквально бросались в глаза еще прежде, чем читатель примется за чтение? Обычно для этого используют более крупный или, по крайней мере, жирный шрифт. Как это делается на "физическом" уровне, вы уже знаете — с помощью дескриптора <B>. Если же вам не хочется вникать в подробности того, как именно будет выделен данный текст — лишь бы заметно выделялся, — можно воспользоваться дескриптором <STRONG>. Этот дескриптор, как и <EM>, парный: все, что заключено между <STRONG> и </STRONG>, обычно выводится на экран жирным шрифтом.

Наконец, бывает не просто важная информация, а нечто такое, чему в последующих фразах дается определение. Например, как в словарях. Для логической разметки таких терминов используется дескриптор <DFN> (от английского *DeFiNition* — "определение"). Обычно содержимое конструкции <DFN>... <DFN> отображается курсивом.

Цитаты

Согласно правилам русского языка, цитаты заключаются в кавычки. Это в электронных публикациях нам придется делать самостоятельно так же, как если бы мы писали ручкой на бумаге в линеечку.

Но, кроме кавычек, цитаты обычно выделяются форматированием. Для коротких — не больше одного предложения — цитат, как правило, достаточно курсива. Более длинные — на несколько предложений — цитаты часто выделяют в отдельный абзац, чтобы текст лучше воспринимался.

Для разметки коротких цитат применяется дескриптор <CITE>. Все, что находится внутри конструкции <CITE>... </CITE>, обычно выделяется курсивом. Для больших цитат используется дескриптор <BLOCKQUOTE>. Все, что находится между <BLOCKQUOTE> и </BLOCKQUOTE>, выносится в отдельный абзац с отступом (рис. 6.2).

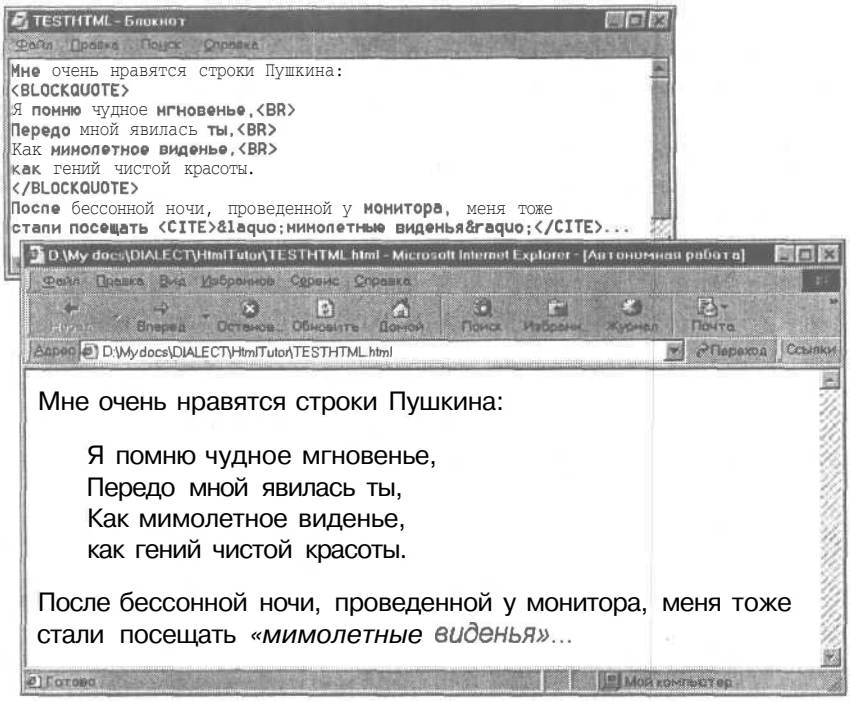


Рис. 6.2. Короткие цитаты обычно выделяются курсивом с помощью дескриптора <CITE>, длинные — выносятся в отдельный абзац с помощью дескриптора <BLOCKQUOTE>

Верхние и нижние индексы

Как известно, текст верхних и нижних индексов выводится более мелким шрифтом и располагается соответственно выше или ниже уровня строки. Конечно, можно изобразить нечто отдаленно похожее на нижний индекс, используя просто мелкий шрифт (рис. 6.3), а если приложить еще больше усилий, — то даже верхний индекс. Но так будет еще менее аккуратно.

Для того чтобы упростить Web-дизайнеру жизнь, в HTML предусмотрены дескрипторы <SUB> и <SUP>. Их названия происходят от английских слов *subscript* и *superscript,* что означает "нижний индекс" и "верхний индекс", соответственно. Оба эти дескриптора парные. Все, что попадает внутрь конструкции _{...}, выводится более мелким шрифтом и на полстроки ниже основного текста, а все, что попадает внутрь конструкции ^{...}, — на полстроки выше основного текста и также более мелким шрифтом. Сравните код, где используются дескрипторы <SUB> и <SUP> (рис. 6.4), с кодом, представленным на рис. 6.3.

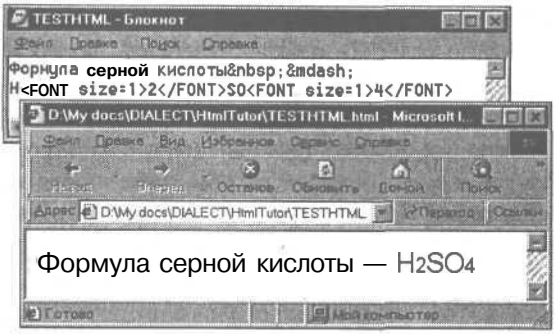


Рис. 6.3. Нечто похожее на нижний индекс можно изобразить с помощью более мелкого шрифта

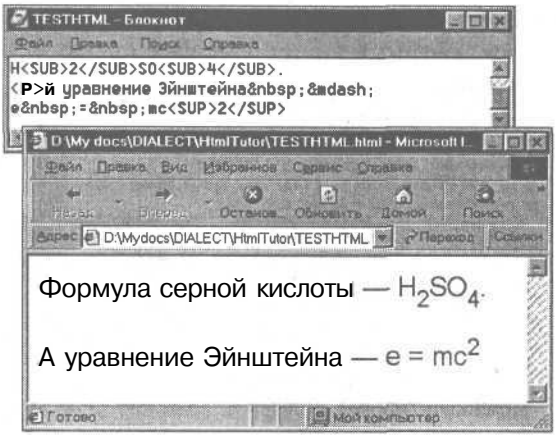


Рис. 6.4. Но с помощью дескрипторов <SUB> и <SUP> можно добиться большего эффекта

Некоторые символы, часто играющие роль верхнего и нижнего индекса, такие как ², ³, °, ™, встречаются в распространенных шрифтах. Что предпочесть: их или специальные дескрипторы индексов — решать вам. Главное, чтобы соответствующий шрифт нашелся и на компьютере посетителя вашей странички (см. также главу 4).

Имитация бурной деятельности

Следующие средства позволяют создать у посетителя впечатление вашей интенсивной работы над обновлением страницы.

Регулярность обновления страницы — вопрос весьма важный. Если страница не обновляется месяцами, то ее станут посещать все реже и однажды вовсе прекратят это делать. Однако слишком частое обновление содержимого страницы может потребовать от вас неоправданных затрат времени и сил. О том, как часто стоит обновлять страницу, читайте в главе 13.

Что обычно является первым признаком того, что над бумажным документом как следует поработали? Поздно сдан? В какой-то степени, да, но нам это не подходит. Измятая бумага? Хорошая идея: имитировать потертости, вмятины и жирные пятна за счет соответствующей графики... Но об этом позже. Ага, вот оно: многочисленные исправления! Их-то можно без особого труда внести и в электронный документ.

Да, но ведь на то он и электронный, чтобы даже после основательной переделки не содержать помарок! Их остается только имитировать.

Как вы уже знаете, в HTML-коде зачеркнутый текст размечается дескрипторами <STRIKE> и <S>. На этом можно было бы и остановиться. Но идея логической разметки заключается в том, чтобы не столько описать вид текста на экране, сколько обозначить его роль в документе. Броузер, как хороший актер, сам разыграет эту роль для посетителя страницы так, чтобы тот все понял правильно.

Для разметки удаленного текста используется дескриптор <DEL>. Его название происходит от английского слова *delete* (удалить). Внешне удаленный текст выглядит так же, как текст, заключенный внутри конструкции <S>... <S> (рис. 6.5).

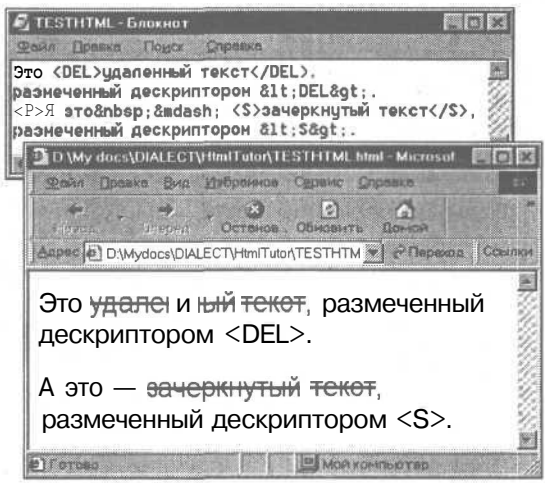


Рис. 6.5. Текст, размеченный как удаленный, выглядит на экране так же, как текст, заключенный внутрь конструкции <S>... <S>

Хорошо, с зачеркнутым текстом мы разобрались. А как имитировать вставки? В "бумажных" документах такие места обычно выглядят мало привлекательными: в лучшем случае над строкой появляется широкая "птичка", а над ней — то, что нужно вставить. О том, как выглядят менее аккуратные вставки, лучше не вспоминать...

В электронных документах даже помарки выглядят аккуратнее. Например, в Microsoft Word исправления, вставленные в текст, выделяются подчеркиванием.

Как разметить подчеркнутый текст, мы уже знаем: с помощью дескриптора <U>. Но это — простое подчеркивание. Для того чтобы указать, что это не просто так отформатированный фрагмент, а текст, вставленный после долгих раздумий, используется логический дескриптор <INS> (от английского *insert* — "вставка"). <INS> — парный дескриптор; текст, помещенный внутри конструкции <INS>... </INS>, считается вставленным и отображается на экране как подчеркнутый (рис. 6.6).

Рис. 6.6. Разметка вставленных и удаленных фрагментов позволяет сделать работу над сайтом видимой для посетителя

Сокращения

Вам, вероятно, знакома ситуация: вы читаете текст, изобилующий сокращениями, их значение разъясняется где-то далеко вначале, искать все это лень, а иначе непонятно, о чем речь...

В Web-документах такая проблема решается с помощью дескриптора <ACRONYM>. Сам по себе он не дает визуальных эффектов и используется главным образом ради параметра title. Значение этого параметра выводится на экран рядом с указателем мыши, если "навести" последний на интересующее вас слово (рис. 6.7).

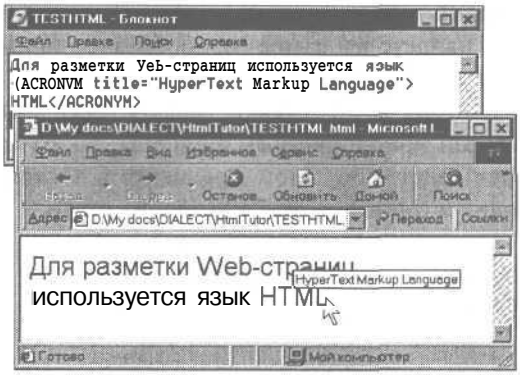


Рис. 6.7. Вывод контекстной подсказки с помощью дескриптора <ACRONYM>

Обратный адрес

Одним из дежурных элементов, кочующих из статьи в письмо, из письма — в новостное сообщение, является подпись. Иногда с указанием званий, должностей и прочих регалий. Часто с адресом, когда — электронным, а когда и обычным, по которому данный субъект получает зарплату или проедает оную.

Для разметки подобных элементов используется дескриптор <ADDRESS>. Это парный дескриптор. То, что заключено между <ADDRESS> и </ADDRESS>, как правило, выводится курсивом и отдельным абзацем (рис. 6.8).

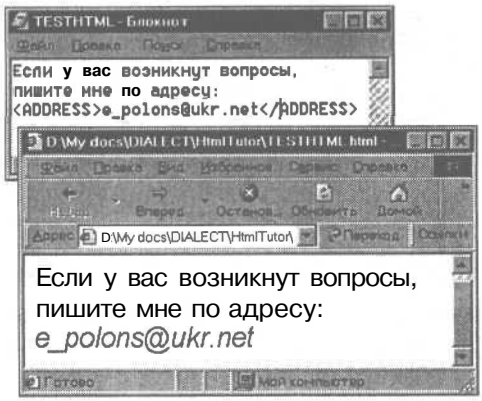


Рис. 6.8. Для разметки подписи, в которую часто входит и адрес, используется дескриптор <ADDRESS>

HTML придумал программист...

Именно такая мысль приходит в голову, когда встречаешь в HTML целый "букет" средств для описания различных программных кодов и результатов их работы. Впрочем, использовать эти средства по назначению вовсе не обязательно. В своей Web-дизайнерской практике (а я надеюсь, со временем она станет весьма богатой!) вы наверняка найдете им самое разнообразное применение.

Небольшой фрагмент программного кода, встречающийся в тексте, можно разметить с помощью дескриптора <CODE>. Текст, заключенный между <CODE> и </CODE>, обычно выводится на экран моноширинным шрифтом, таким как Courier.

А как быть, если речь идет о большом фрагменте в несколько строк?

Первое, что приходит в голову, — код наподобие этого:

```
<CODE>
<P> первая строка программы
<P> вторая строка программы
<P> ...
</CODE>
```

Ради экономии места можно заменить дескрипторы **<P>** на <BR>. Наконец, комбинируя абзацы с разрывами строк, можно разделять логически обособленные фрагмента кода (рис. 6.9).

Но в программных кодах, кроме пустых строк, часто встречаются табуляции и другие отступы, "разбавляющие" тарабарский текст и позволяющие хотя бы посвященным понять, что там написано. Заменять эти табуляции и пробелы на коды соответствующих символов — занятие слишком неблагодарное, чтобы ушлые Web-дизайнеры не попытались от него избавиться.

И успешно избавились, введя дескриптор <PRE> (от английского *preformatted*— "с сохранением форматирования"). Этот дескриптор замечателен тем, что внутри конструкции <PRE>... </PRE> все пробелы, табуляции и переходы на новую строку сохраняются в том виде, в котором они были введены в HTML-редакторе.

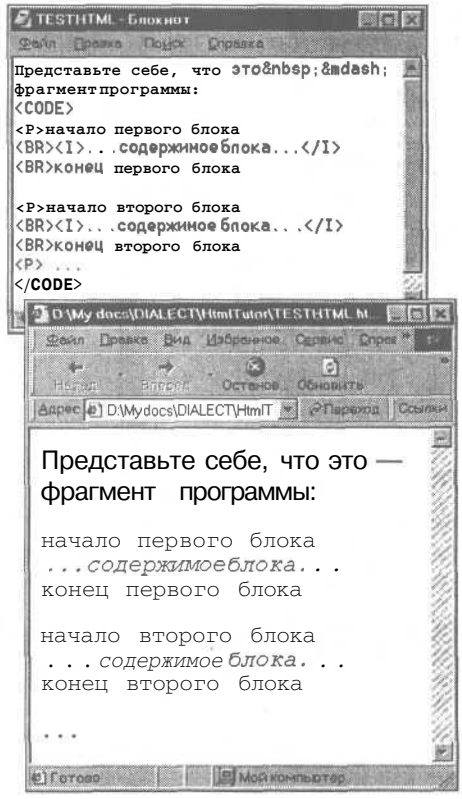


Рис. 6.9. Один из вариантов отображения больших фрагментов кода — с помощью дескрипторов <CODE>, <P> и

Однако не стоит думать, судя по названию дескриптора, что внутри него сохраняются также и другие элементы форматирования, такие как гарнитура и начертание шрифта или спецсимволы. Для отображения этих элементов, как и прежде, приходится пользоваться соответствующими дескрипторами HTML, а для спецсимволов — их кодами и мнемоническими именами (рис. 6.10).

Содержимое конструкции <PRE>... </PRE> всегда выводится на экран с новой строки. Поэтому предварять такой текст дескриптором <P> не обязательно.

Кстати, есть такое правило: если в тексте — например, по физике или математике — встречается переменная, то она выводится курсивом. Мы могли бы привести в соответствие с этим правилом код, представленный на рис. 6.8, с помощью дескрипторов <I>. Но ведь *x* и *y* — не просто какие-то "курсивные места".

Это переменные. А в HTML для разметки переменных есть специальный дескриптор <VAR> (от английского *variable* — "переменная"). Воспользуемся им (рис. 6.11). Что получилось в результате? Внешне — тот же курсив. Но, кроме того, такая разметка сообщает броузеру о назначении данного фрагмента.

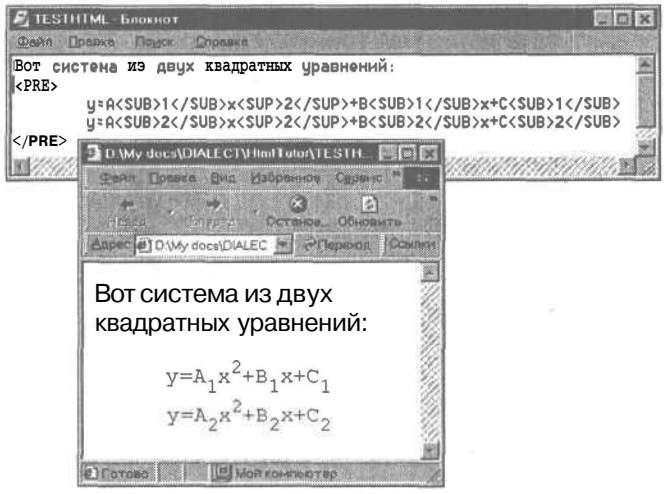


Рис. 6.10. Внутри дескриптора <PRE> сохраняются отступы и абзацы, но для специальных символов по-прежнему используются их коды, а для элементов форматирования — соответствующие дескрипторы

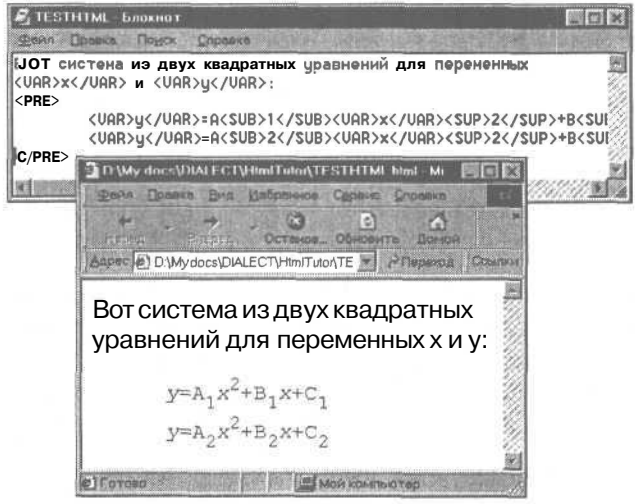


Рис. 6.11. Благодаря дескриптору <VAR> имена переменных выделяются. В данном случае, курсивом

Какие еще есть элементы в описаниях программ, кроме фрагментов кодов и переменных? Правильно, текст, вводимый пользователем и примеры текста, выводимого программой на экран. Первый размечается с помощью дескриптора

<KBD> (от английского *keyboard* — "клавиатура"), второй — с помощью дескриптора <SAMP> (от английского *sample* — "образец", "пример"). В окне броузера оба вида текста отображаются моноширинным шрифтом.

Зачем все это?

Зачем же столько дескрипторов, если результат зачастую одинаковый?
Уточним: *пока* одинаковый.

Кто знает, как будут обходиться с логическими дескрипторами будущие броузеры и что они вообще станут с ними делать? Например, они могут "проговаривать" некоторые элементы текста, используя систему синтеза речи... И вообще, кто знает, что нам готовит будущее! Вероятно, именно поэтому многие средства HTML сделаны как бы "на вырост", и в справочниках рядом с ними то и дело встречаешь пометки: "пока не используется ни в одном броузере"...

"Физические" и "логические"... Часто, когда нам встречаются подобные диаметрально противоположные пары свойств или вещей, хочется спросить: "А какие лучше?".

И не менее часто ответом на этот вопрос служит фраза: "Смотря что вам нужно".

В действительности сейчас существует тенденция постепенной замены физической разметки гипертекста на логическую. Главной причиной этого является желание унифицировать вид электронных документов в соответствии с устоявшимися традициями и правилами, а также — что более важно — упростить анализ таких документов поисковыми системами. Задача таких систем — понять, о чем говорится в тексте и поместить его в соответствующий раздел электронного каталога. Если в тексте уже на стадии верстки будут выделены главные элементы, эта задача значительно упростится. Подробнее об этом читайте в главе 14.

Резюме

Все дескрипторы HTML можно условно разделить на две большие категории – "физические" и "логические", т.е. такие, которые определяют только внешний вид документа, и такие, которые несут информацию о его логической структуре.

К "физическим" дескрипторам относятся такие, как уже знакомые нам <U>, **<I>, <S>,** <FONT>, а к "логическим" - дескрипторы выделения главного <EM> и <STRONG>, определений <DFN>, цитат <CITE> и <BLOCKQUOTE>, нижнего и верхнего индекса <SUB> и <SUP>, удаленного и вставленного текста <DEL> и <INS>, сокращений <ACRONYM>, адресов <ADDRESS>, программных фрагментов <CODE>, переменных <VAR>, ввода с клавиатуры и вывода на экран <KBD> и <SAMP>.

Несмотря на то, что многие "логические" дескрипторы дублируют друг друга в смысле внешних эффектов, они полезны: благодаря им, во-первых, упрощается анализ страницы поисковыми системами Internet (см. главу 14), а во-вторых, достигается единообразие оформления страниц и их соответствие полиграфическим правилам и традициям.

Тесты

1. Дескриптор **<S>** является аналогом:
   - **а)** дескриптора <STRONG>;
   - **б)** дескриптора <EM>;
   - **в)** дескриптора **<STRIKE>**;
   - г) дескриптора **<DEL>**.
2. Какой или какие из следующих фрагментов кода содержат ошибки?
   - **а)** **<BLOCK QUOTE>Тише едешь — дальше будешь</BLOCK** QUOTE>. Народная мудрость
   - **б)** **<BLOCKQUOTE>Тише едешь — дальше будешь</BLOCKQUOTE>.** Народная мудрость
   - **в)** **<BLOCK quote="Тише едешь — дальше будешь">.** Народная мудрость
   - **г)** **<ACKRONYM title="язык** разметки **гипертекста">HTML</ACKRONYM>**
   - **д)** **<ACKRONYM>HTML</ACKRONYM> —** язык разметки гипертекста
   - **е)** Формула этилового **спирта — C₂H₅OH**
   - **ж)** Формула этилового **спирта — C²H⁵OH**
3. Какой из следующих фрагментов состоит из двух абзацев?
   - **а)** **«<CITE>Тише едешь —** дальше будешь**</CITE>» —** народная мудрость
   - б) **<CITE>Тише едешь —** дальше **будешь</CITE>. <ADDRESS>Народная мудрость</ADDRESS>**
   - **в)** **<BLOCKQUOTE>Тише едешь —** дальше **будешь</BLOCKQUOTE>. <ADDRESS>Народная мудрость</ADDRESS>**
   - **г)** **<BLOCKQUOTE>Тише едешь — дальше будешь</BLOCKQUOTE>.** Народная мудрость
   - д) **<PRE>Тише едешь — дальше будешь</PRE>.** Народная мудрость
4. Какие из следующих пар фрагментов отображаются в броузере одинаково?

а)	**<DFN>HTML</DFN> —** язык разметки гипертекста	**<DEL>HTML</DEL> —** язык разметки гипертекста
б)	**<DFN>HTML</DFN> —** язык разметки гипертекста	<ACKRONYM **title="язык** разметки **гипертекста">HTML</ACKRONYM>**
в)	**<DFN>HTML</DFN> —** язык разметки гипертекста	**<B>HTML</B> —** язык разметки гипертекста
г)	**<DFN>HTML</DFN> —** язык разметки гипертекста	**<I>HTML</I> —** язык **разметки** гипертекста

5. Какие из следующих пар фрагментов отображаются в броузере одинаково?

а)	Сама + **<DEL>Маша</DEL>** **<INS>Наташа</INS>** = Дружба	Саша + <B>Маша</B> **<I>Наташа</I>** = Дружба

б) Саша + <DEL>Маша</DEL> <INS>Наташа</INS> = Дружба

Саша + <S>Маша</S> <U>Наташа</U> = Дружба

в) Саша + <DEL>Маша</DEL> <INS>Наташа</INS> = Дружба

Саша + <STRIKE>Маша</STRIKE> <STRONG>Наташа</STRONG> = Дружба

6. Какие из следующих пар фрагментов отображаются в броузере одинаково?

а) Вот фрагмент HTML-кода: <PRE>
<I>курсив<I>
<U>подчеркнутый<U>
</PRE>

Вот фрагмент HTML-кода: <PRE>
курсив
подчеркнутый
</PRE>

б) Вот фрагмент HTML-кода: <PRE>
<I>курсив</I>
<U>подчеркнутый</U>
</PRE>

Вот фрагмент HTML-кода: <CODE>
<I>курсив</I>
<U>подчеркнутый</U>
</CODE>

в) Вот фрагмент HTML-кода: <PRE>
<I>курсив</I>
<U>подчеркнутый</U>
</PRE>

Вот фрагмент HTML-кода: <CODE>
<I>курсив</I>
<U>подчеркнутый</U>
</CODE>

г) Вот фрагмент HTML-кода: <PRE>
<I>курсив</I>
<U>подчеркнутый</U>
</PRE>

Вот фрагмент HTML-кода: <CODE>

   <I>курсив</I>

   <U>подчеркнутый</U>
</CODE>

д) Вот фрагмент HTML-кода: <PRE>
<I>курсив</I>
<U>подчеркнутый</U>
</PRE>

Вот фрагмент HTML-кода: <FONT face="Courier New">

   <I>курсив</I>

   <U>подчеркнутый</U>
</FONT>

7. Какие из следующих пар фрагментов отображаются в броузере одинаково?

а) Дескриптор <VAR><SUP></VAR> предназначен для отображения верхнего индекса

Дескриптор <I><SUP></I> предназначен для отображения верхнего индекса

б) Дескриптор <VAR><SUP></VAR> предназначен для отображения верхнего индекса

Дескриптор <FONT face="Courier New"><SUP></FONT> предназначен для отображения верхнего индекса

в) Дескриптор <KBD><SUP></KBD> предназначен для отображения верхнего индекса

Дескриптор <FONT face="Courier New"><SUP></FONT> предназначен для отображения верхнего индекса

г) Дескриптор <SAMP><SUP></SAMP> предназначен для отображения верхнего индекса

Дескриптор <FONT face="Courier New"><SUP></FONT> предназначен для отображения верхнего индекса

Глава 7

Списки

В этой главе...

- Концепция списков в HTML
- Нумерованные списки
- Маркированные списки
- Параметры элемента списка
- Многоуровневые списки
- Списки определений
- Другие виды списков

Концепция списков в HTML

Автоматическое создание списков — очень удобная функция текстовых процессоров и причина первых восторгов у многих новичков. Это понятно: вместо того чтобы следить за нумерацией и отступами, можно уделить больше внимания логическому построению перечня и другим более важным вещам. Программа сама расставит нужные цифры, проследит за их форматированием. Кроме того, вместо плюсов и черточек, которые используют в ненумерованных списках машинистки, можно подобрать приличествующие случаю кружки, "птички" и т.п.

Как создать список в HTML?

Конечно, можно поступить просто, "по-машинистски", т.е. как секретарь-машинистка: самому ввести цифры и маркеры, лично проследить за отступами (рис. 7.1). Получится почти как на пишущей машинке. Даже лучше.

Есть ли другой, более простой и изящный способ? Разумеется, есть.

Все списки в HTML — как нумерованные, так и ненумерованные (маркированные) — размечаются по единому принципу, с помощью двух видов парных дескрипторов. Внутри первого заключается весь список. Этот дескриптор определяет тип списка и его параметры в целом. Каждый элемент списка заключается внутри дескриптора второго типа, который определяет параметры данного элемента. Если обозначить эти дескрипторы как <Д1> и <Д2>, то общая структура списка будет выглядеть так:

```
<Д1 параметры всего списка>
    <Д2 параметры элемента 1>Элемент 1</Д2>
    <Д2 параметра элемента 2>Элемент 2</Д2>
```

```
...
<Д2 параметры элемента 3>Элемент 3</Д2>
</Д1>
```

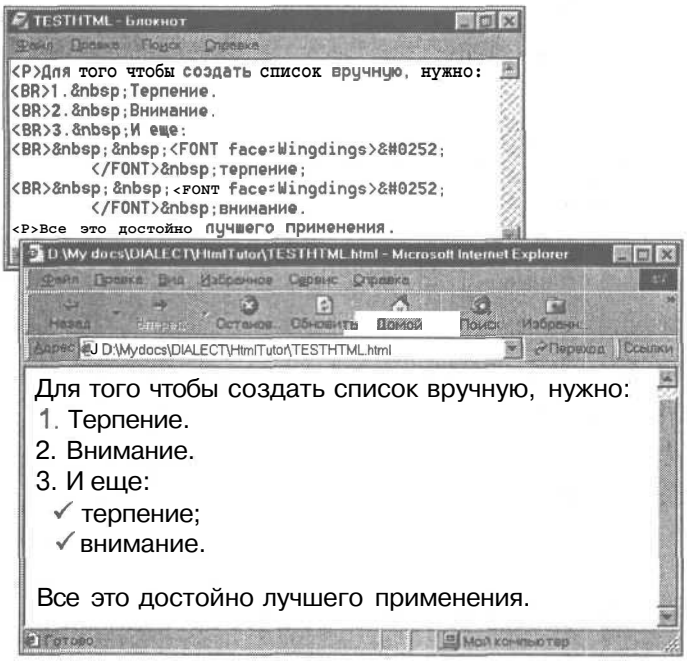


Рис. 7.1. Даже вручную можно создать достаточно сложный список

Нумерованные списки

Для создания нумерованных списков используется пара дескрипторов <OL> и <LI>. Название первого происходит от английского *ordered list* — "упорядоченный список", а название второго — от английского *list item* ("элемент списка").

Попробуем разметить список по уже знакомой нам схеме: поместить весь список внутрь конструкции <OL>... </OL>, а каждый элемент — в отдельную конструкцию <LI>... </LI> (рис. 7.2). Похоже, мы, действуя по привычке, ввели в код много лишнего...

Убрав нумерацию — она теперь выполняется автоматически, — а также ставшие ненужными пробелы и разрывы строк, получаем очень простой и изящный код (рис. 7.3).

Хотя дескриптор <LI> и парный, закрывать его не обязательно: его "закроет" следующий дескриптор <LI> или закрывающий дескриптор </OL> (а вот последний обязателен!). Дескриптор <P> в начале следующего за списком абзаца также можно опустить: текст после закрывающего дескриптора списка всегда начинается с новой строки.

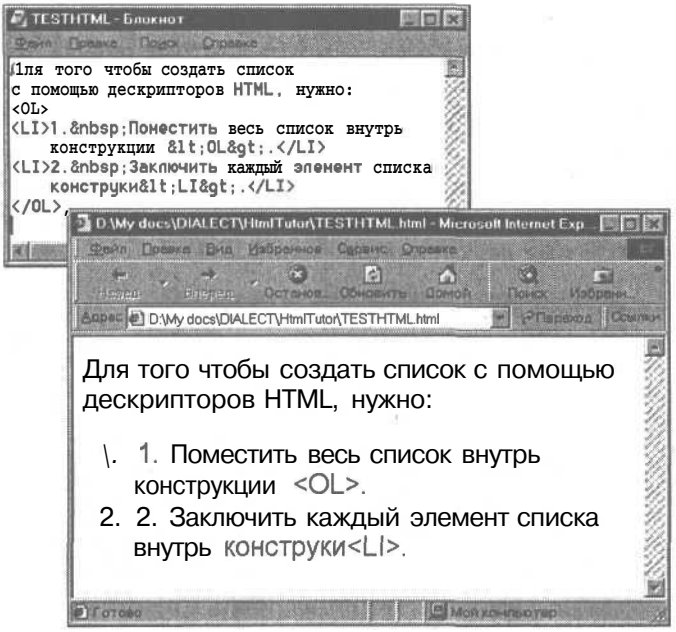


Рис. 7.2. Разметив список с помощью дескрипторов HTML, обнаруживаем, что многие вещи стали лишними

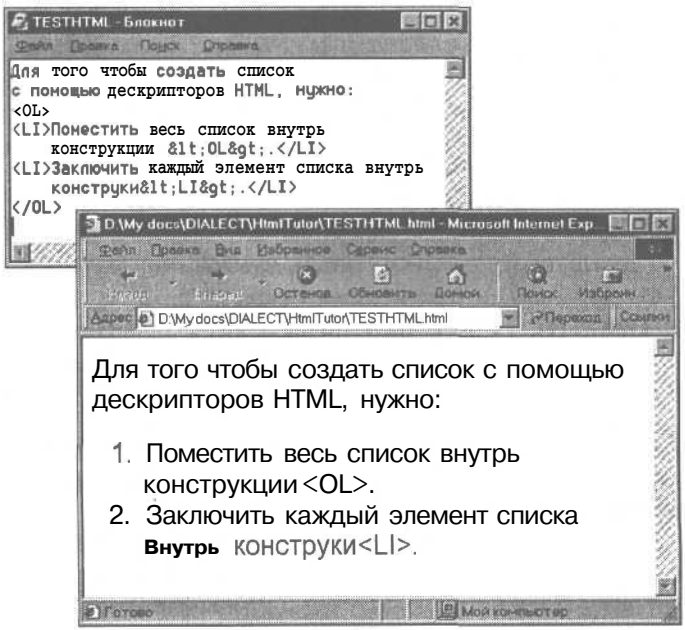


Рис. 7.3. При такой разметке нумерация выполняется автоматически

Отлично. Но как теперь изменить нумерацию, например, чтобы она начиналась не с единицы, а с нуля? Или например, заменить арабские цифры латинскими?

В текстовых процессорах для этого используются специальные функции, определяющие свойства всего списка. В HTML для этого есть параметры дескриптора <OL> — start и type.

По умолчанию нумерация списка всегда начинается с единицы (start=1). Для того чтобы она начиналась, например с нуля (как любят программисты), нужно указать в дескрипторе <OL> параметр start=0 (рис. 7.4).

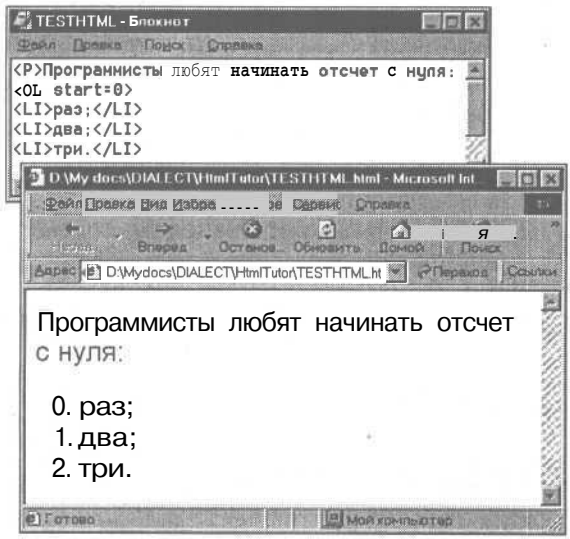


Рис. 7.4. Начальный номер списка определяется параметром start

Попробуем теперь заменить нумерацию с арабской на латинскую. Для этого нам понадобится параметр type. Он может принимать одно из пяти значений: A, a, I, i или 1, которые соответствуют типам нумерации: большими латинскими буквами (A, B, C, ...), малыми латинскими буквами (a, b, c, ...), большими римскими цифрами (I, II, III, IV, ...), малыми римскими цифрами (i, ii, Hi, iv, ...) и арабскими цифрами (1, 2, 3, ...). Таким образом, нам нужно присвоить параметру type значение I (рис. 7.5).

К сожалению, в HTML нет способа автоматической разметки списка, пронумерованного кириллицей: *а, б, в,* и т.д. Как нет и способа изменить символ, стоящий после буквы или цифры: это всегда точка. Поэтому для того чтобы создать список, подобный списку ответов в этой книге, придется использовать ручную разметку.

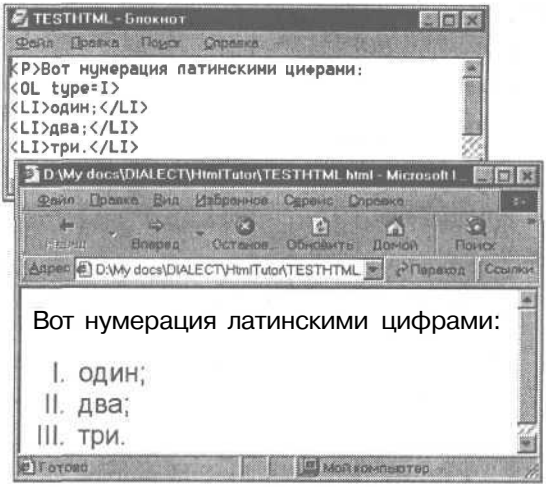


Рис. 7.5. Для того чтобы изменить тип нумерации, используется параметр type

Маркированные списки

Как известно, кроме нумерованных списков, т.е. таких, где важен порядок следования элементов, есть такие, где важен только их перечень. Вместо цифр или букв в них используют маркеры — точки, черточки, кружки. Такие списки называют *маркированными*.

Для разметки маркированных списков в HTML применяется тот же принцип, что и для нумерованных списков, только вместо дескриптора <OL> используется дескриптор <UL>. Как нетрудно догадаться (если, конечно, знать английский), его название происходит от английского *unordered list* — "неупорядоченный список".

Что получится, если в рассмотренном выше примере заменить дескриптор <OL> на <UL>? Нумерация исчезнет, а вместо цифр в начале каждой строки появятся характерные жирные точки (рис. 7.6).

А как заменить эти точки на что-то другое, например на "птички"?

Ответ на этот простой, казалось бы, вопрос, будет несколько длиннее, чем ему полагается быть.

Часть первая, короткая: у дескриптора <UL>, как и у <OL>, есть параметр type. Он-то и определяет вид маркера.

Часть вторая, из-за которой ответ становится длинным: параметр type принимает только три значения: circle, disc и square. По умолчанию он равен disc, что и соответствует жирным точкам. Значение circle означает маркеры в виде маленьких окружностей, а square — в виде квадратиков (рис. 7.7).

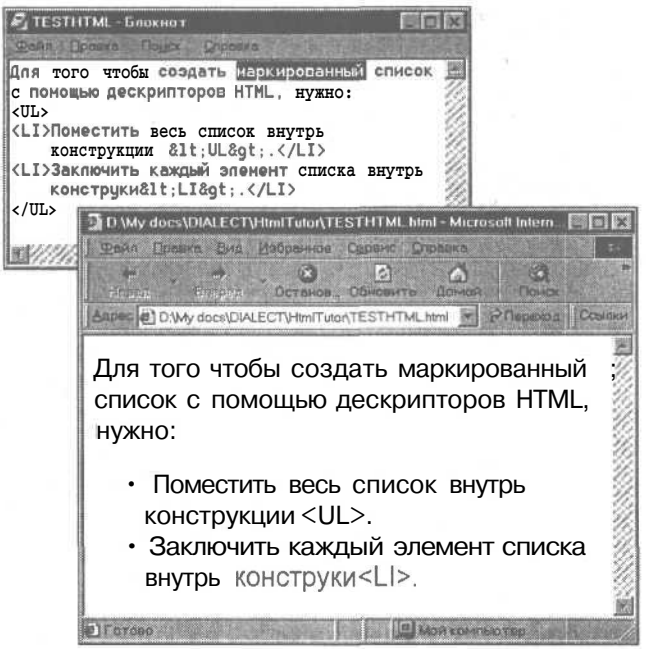


Рис. 7.6. Вот что получается, если заменить дескриптор `<OL>` на `<UL>`

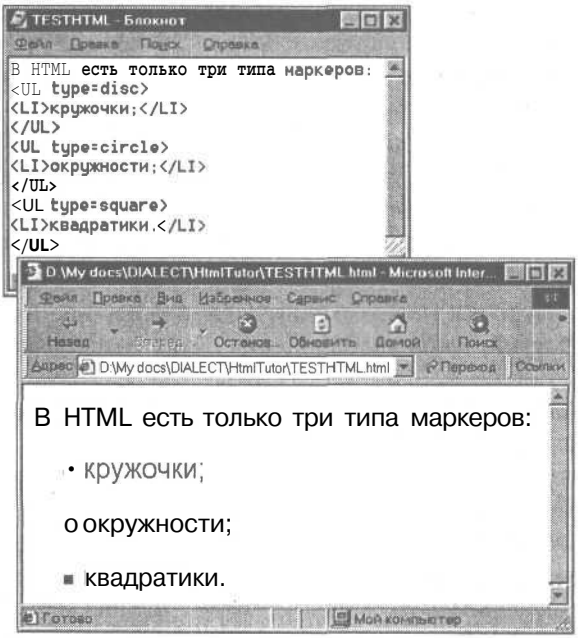


Рис. 7.7. В HTML предусмотрено только три вида маркеров: жирные точки, окружности и квадратики

Наконец, часть третья, длинная: что делать, если хочется использовать нестандартные маркеры?

Отказаться от разметки списка и сделать все самому. Лично проследить за маркерами и отступами. А, возможно, и собственноручно нарисовать эти самые маркеры в Photoshop (см. главу 8).

Возможно, однажды в новую версию стандарта HTML будут внесены необходимые изменения — то ли увеличится количество маркеров, то ли будет предусмотрена возможность вставки в качестве маркеров любых символов. А пока остается использовать то, что есть или верстать списки вручную. Только в последнем случае нужно помнить, что далеко не любой выбранный вами символ есть на компьютере пользователя. Возможно, именно поэтому дизайнеры или используют стандартные маркеры, или рисуют собственные и помещают их на Web-страницу в виде графических файлов.

Некоторые Web-дизайнеры используют такой прием: формируют список, пользуясь только дескрипторами <LI>. Попробуем так сделать и мы (рис. 7.8). Что получается? Маркированный список без горизонтальных отступов. И с полным пренебрежением к абзацам. Похоже, если навести порядок с последними, то такая конструкция вполне имеет право на существование.

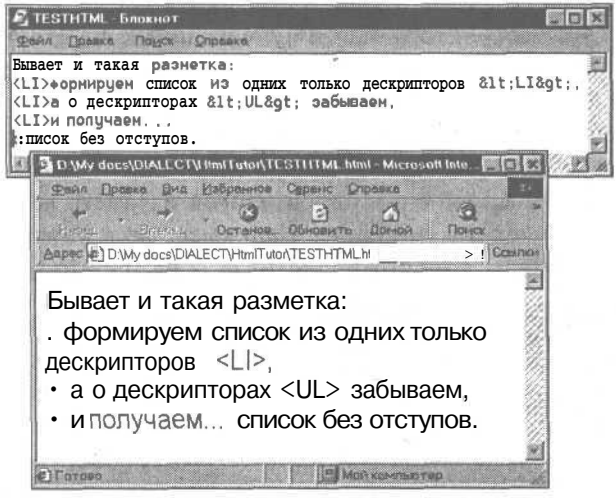


Рис. 7.8. Если убрать дескрипторы <UL>, получится маркированный список без отступов

Параметры элемента списка

А как быть, если параметры некоторых элементов одного списка отличаются от остальных? Неужели только из-за этого нужно дробить список на мелкие части, как это сделано на рис. 7.7?

К счастью, у нас есть возможность оптимизировать HTML-код. Для этого нужно воспользоваться параметрами дескриптора <LI>, а именно параметром type (рис. 7.9).

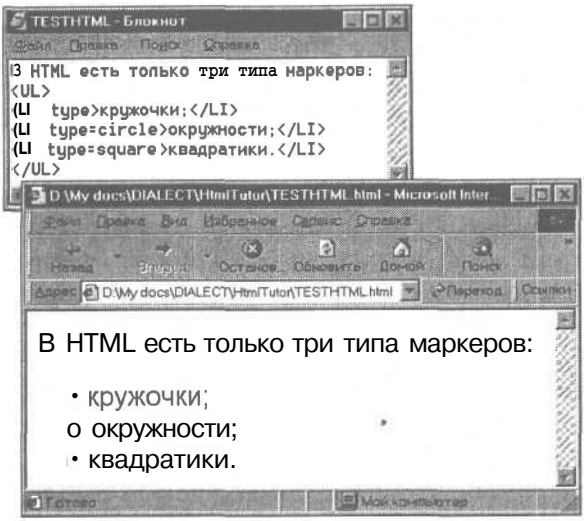


Рис. 7.9. Благодаря использованию параметра type в дескрипторе <LI> можно менять тип маркера внутри списка

А теперь вопрос на сообразительность: что определяет и какие значения принимает параметр type дескриптора <LI> в нумерованных списках? Предупреждаю, что вся информация, необходимая для того, чтобы дать правильный ответ, вам уже известна. На всякий случай — вдруг вы что-то подзабыли — перечитайте начало этой главы еще раз... Ну как, ответ готов? Тогда читайте дальше.

Действительно, в нумерованных списках параметр type дескрипторов <LI> принимает такие же значения, что и одноименный параметр всего списка в дескрипторе <OL>. И изменяет его значение. Например, весь список может нумероваться арабскими цифрами, а один из элементов — римскими (рис. 7.10).

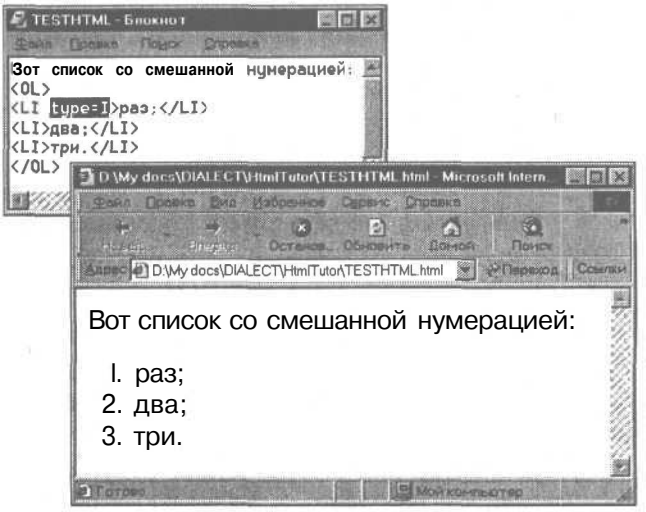


Рис. 7.10. Список со смешанной нумерацией

Правда, такая смешанная нумерация не является типичной. Гораздо чаще используют вложенные списки, которые рассматриваются ниже в этой главе.

Таким образом, параметр type в дескрипторе <LI> дополняет и расширяет возможности одноименного параметра дескрипторов <OL> и <UL>. Возможно, у вас уже вертится на языке вопрос: а есть ли у дескриптора <LI> параметр, дополняющий параметр start и позволяющий "нарушить" последовательность нумерации, не изменяя ее типа? Например так: 1, 2, 3, 5, 6?

Если такой вопрос у вас возник, то вы мыслите в правильном направлении. Действительно, у дескриптора <LI> есть такой параметр, и называется он value (от английского *value* — "значение"). Этот параметр определяет не только номер данного элемента списка, но и номера последующих элементов (рис. 7.11).

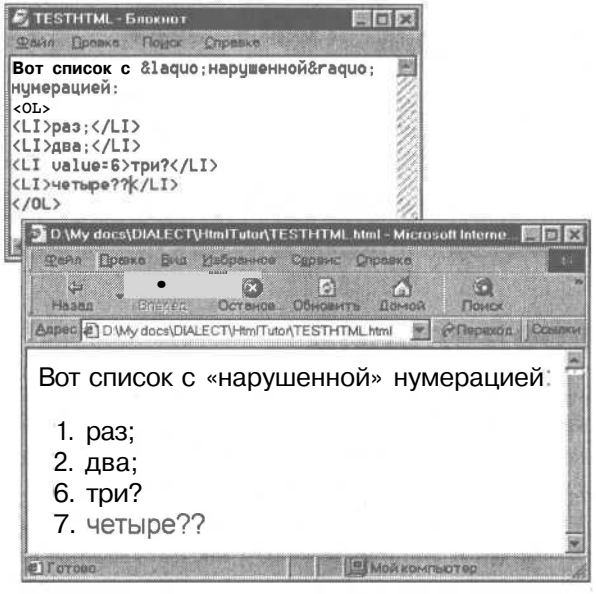


Рис. 7.11. Параметр value позволяет "нарушить" последовательность нумерации

Многоуровневые списки

И еще один вопрос, который у многих уже, наверное, давно вертится на языке: как быть с многоуровневыми списками? В Microsoft Word, например, для этого есть целая вкладка диалогового окна со множеством вариантов этих самых списков. А как в HTML?

А в HTML многоуровневых списков *нет*.

То есть, для них нет отдельной конструкции. Но она и не нужна. Для разметки многоуровневых списков достаточно того, что нам уже известно. В тех местах, где нужно вставить подпункты списка, используются вложенные списки, аналогично любым вложенным дескрипторам (рис. 7.12).

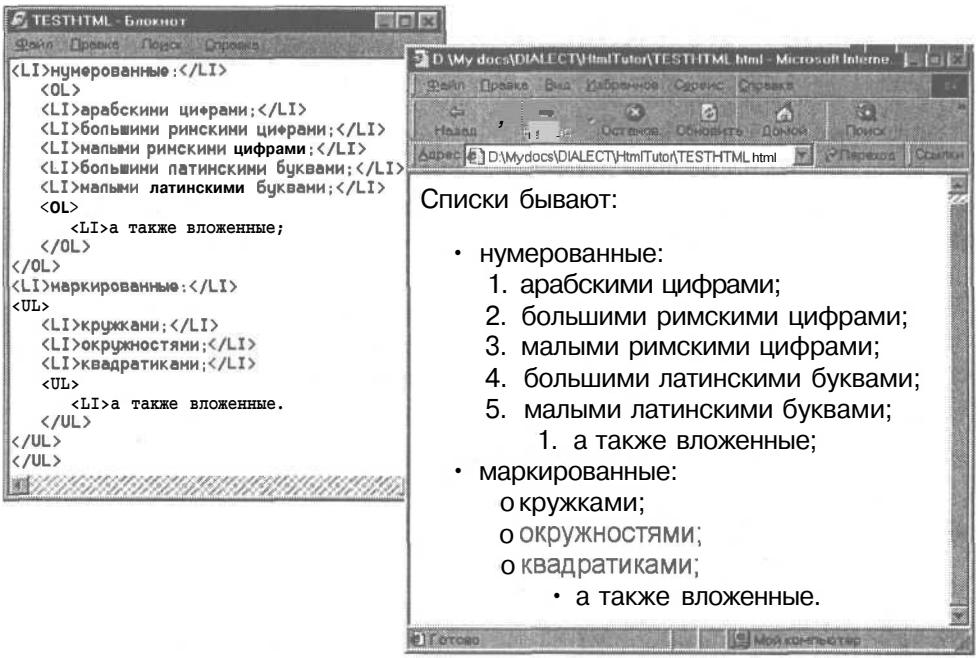


Рис. 7.12. Многоуровневые списки создаются из обычных списков, вложенных друг в друга

Списки определений

Об этом виде списков вспоминают достаточно редко. В большинстве случаев, когда хочется сделать красиво, воображение "зацикливается" на выборе того или иного маркера. Как будто единственное, от чего зависит красивый и аккуратный дизайн, — это пресловутые "птички" или кружочки!

На самом деле достаточно часто — гораздо чаще, чем кажется на первый взгляд — можно использовать еще один вид списков. Это так называемые *списки определений*.

Возможно, одной из причин такого консерватизма мышления является само название этого вида списков. Действительно, *определения* в чистом виде и в таком количестве, чтобы образовать список, встречаются нечасто, разве что вы решили сделать электронную версию толкового словаря.

Но очень часто списки строятся по принципу: сначала короткая фраза или слово (как правило, выделенное полужирным шрифтом или как-нибудь иначе), потом более подробное разъяснение. Такие фрагменты можно организовать по-разному, в зависимости от общего вида страницы: можно как обычные абзацы, а можно — как списки определений.

Из чего состоит такой список? Очевидно, из термина, который нужно описать, и самого описания. Следовательно, сколько дескрипторов для этого нужно? По первой прикидке — два: для термина и для описания. Да, пожалуй, понадобится еще один - для разметки самого списка, как <OL> для нумерованного и <UL> для маркированного.

Так и есть: весь список определений заключается внутрь парного дескриптора <DL> (от английского *definition list* — "список определений"), каждый термин — внутрь дескриптора <DT> (от английского *definition term* — "определяемый термин") и каждое описание — внутрь дескриптора <DD> (от английского *definition description* — "описание определения"). Общий вид списка определений показан на рис. 7.13.

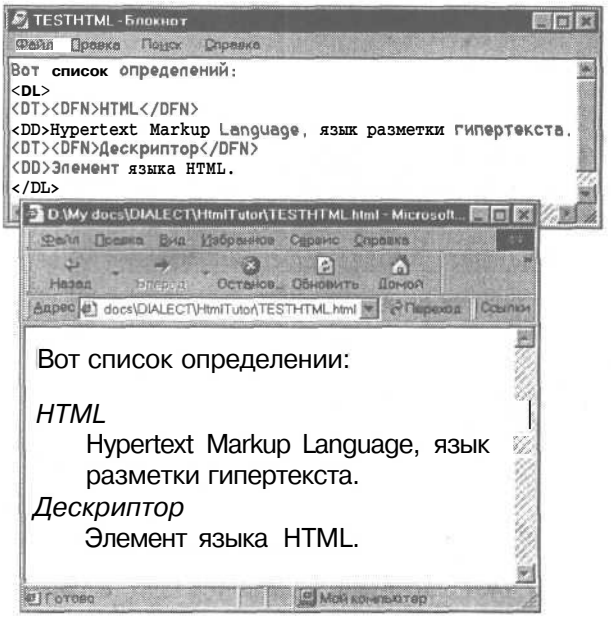


Рис. 7.13. Список определений: дескрипторы <DL>, <DT> и <DD> "отвечают" только за отступы. За остальным форматированием приходится следить самому

Как видим, конструкция, образованная дескрипторами <DL>, <DT> и <DD>, обеспечивает использование следующих элементов форматирования:

* отступ между предыдущим абзацем, списком и следующим абзацем;
* каждый термин находится в отдельной строке;
* каждое описание термина также находится в отдельном абзаце, причем сдвинуто вправо относительно термина.

И все. Если мы хотим выделить определения еще как-нибудь, например цветом или полужирным шрифтом, придется делать это вручную.

Здесь уместно вспомнить о различиях между физической и логической разметкой. Можно выделить определения с помощью, например дескриптора **<B>** или <I>, а можно воспользоваться уже известным нам дескриптором для разметки определений — <DFN>.

Посмотрим, какие дескрипторы "отвечают" за каждый из этих элементов. Для этого мы проделаем небольшой эксперимент.

Вначале уберем дескрипторы <DL> и </DL>. Что получилось? Пропали отступы между списком и окружающими его абзацами. Кроме того, исчез отступ между термином и определением. Сохранилась только "красная строка", а остальные строки абзаца начинаются вровень с другими абзацами (рис. 7.14).

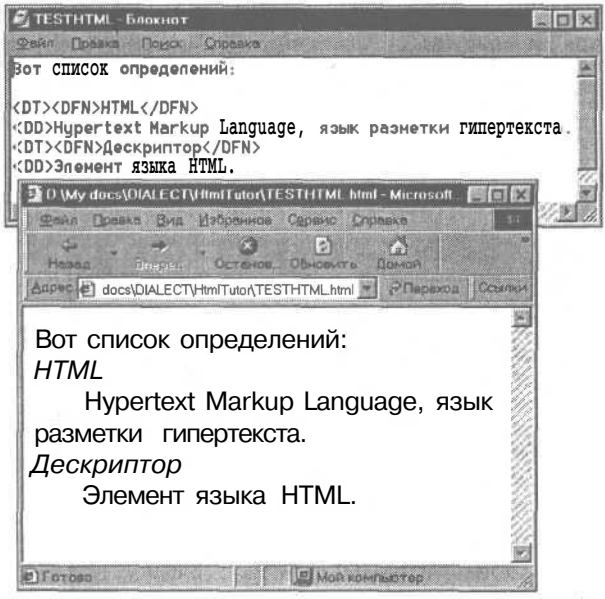


Рис. 7.14. Если убрать дескриптор <DL>, исчезают вертикальные отступы между списком и другими абзацами, а также горизонтальные отступы между терминами и их определениями

Что ж, такой эффект вполне имеет право на существование. Иногда отступы выглядят красиво, а иногда лучше без них.

Попробуем теперь убрать дескрипторы <DD>. По идее, они должны "отвечать" за остальное, а именно, за то, что описания начинаются с новой строки. Так и есть: убрали <DD> — и определения объединились с терминами в один абзац (рис.7.15).

Что ж, такой вариант разметки тоже вполне имеет право на существование. Пожалуй, для нас он даже привычнее, чем "классический" список определений. Только следует помнить о том, что по правилам русского языка в таких случаях после термина перед его описанием обязательно ставится какой-нибудь знак препинания — точка, если дальше следует отдельное предложение, а если нет, то двоеточие или тире.

Другие виды списков

Стандартом HTML предусмотрены еще два вида списков: каталоги и меню. Первый описывается дескриптором <DIR> *(directory list),* а второй — дескриптором <MENU>. Их способ применения и параметры ничем не отличаются от способа применения и параметров дескриптора <UL>.

Не отличаются и сами списки: текст, размеченный с помощью этих дескрипторов и открытый в окне броузера, неотличим от текста, размеченного с помощью дескриптора <UL>. Это еще два дескриптора, сделанных "на вырост": пока что они не востребованы, но, возможно, в будущем ситуация изменится.

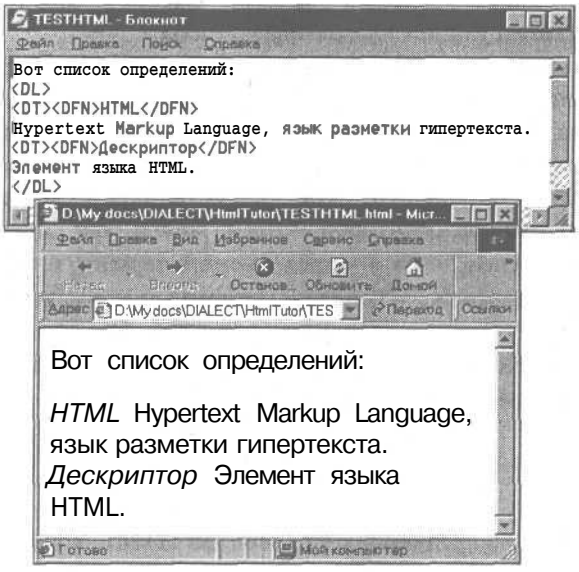


Рис. 7.15. Если в коде списка определений убрать дескрипторы <DD>, то каждый термин с его описанием образует один абзац

Резюме

Нумерованные и маркированные списки размечаются в HTML с помощью дескрипторов двух типов: первый определяет параметры всего списка, второй — параметры каждого из его элементов.

Маркированные списки описываются дескриптором <UL>. Его параметр type определяет вид маркера — квадратики (square), кружки (disc) и "пустые" окружности (circle). По умолчанию параметру type присваивается значение disc.

Нумерованные списки описываются дескриптором <OL>. Этот дескриптор имеет два параметра: type, определяющий способ нумерации, и start, определяющий, с какой буквы или цифры она будет начинаться. Параметр type дескриптора <OL> принимает значения 1, I, i, A или a, что соответствует нумерации арабскими, большими и малыми римскими цифрами, а также большими или малыми латинскими буквами. Других вариантов нумерации, в частности буквами кириллицы, к сожалению, не предусмотрено.

Элементы нумерованных и маркированных списков размечаются с помощью дескрипторов <LI>. Этот дескриптор имеет те же параметры* что и дескриптор всего списка: если список нумерованный, то это type и start, а если маркирован-

ный, то только type. Параметры дескриптора <LI> имеют более высокий приоритет, чем параметры всего списка и, таким образом, позволяют изменить порядок нумерации или вид маркера.

Для организации многоуровневых списков со смешанной нумерацией используются вложенные дескрипторы <UL> и <OL>: вместо очередного блока <LI>... </LI> ставится соответствующий вложенный список.

Кроме маркированных и нумерованных списков, в HTML предусмотрена конструкция, образующая *список определений.* Каждый элемент такого списка состоит из некоего термина и его определения. Термины и определения находятся в отдельных абзацах, причем последние выводятся с увеличенным горизонтальным отступом относительно остального текста. Разметка списка определений осуществляется с помощью трех дескрипторов — <DL>, <DT> и <DD>. Дескриптор <DL> описывает весь список в целом, <DT> — определяемый термин, а <DD> — определение.

Кроме того, в стандарте HTML предусмотрены еще два вида списков — каталоги (дескриптор <DIR>) и меню (дескриптор <MENU>). Однако эти дескрипторы не получили распространения, и размеченные ими списки в большинстве броузеров отображаются как обычные маркированные.

Тесты

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а)
   ```
   <UL start=1>
   <LI> Первый элемент списка
   <LI> Второй элемент списка
   </UL>
   ```

   б)
   ```
   <UL start=circle>
   <LI> Первый элемент списка
   <LI> Второй элемент списка
   </UL>
   ```

   в)
   ```
   <UL type=1>
   <LI> Первый элемент списка
   <LI> Второй элемент списка
   </UL>
   ```

   г)
   ```
   <UL type=circle>
   <LI> Первый элемент списка
   <LI> Второй элемент списка
   </UL>
   ```

2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а)
   ```
   <OL start=1>
   <LI> Первый элемент списка
   <LI> Второй элемент списка
   </OL>
   ```

б)
```
<OL start=circle>
<LI> Первый элемент списка
<LI> Второй элемент списка
</OL>
```

в)
```
<OL type=1>
<LI> Первый элемент списка
<LI> Второй элемент списка
</OL>
```

г)
```
<OL type=circle>
<LI> Первый элемент списка
<LI> Второй элемент списка
</OL>
```

3. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а)
```
<DL>
<DT> Термин 1</DT>
<DD> Определение 1</DD>
<DT> Термин 2</DT>
<DD> Определение 2</DD>
</DL>
```

б)
```
<DT> Термин 1</DT>
<DD> Определение 1</DD>
<DT> Термин 2</DT>
<DD> Определение 2</DD>.
```

в)
```
<DL type=circle>
<DT> Термин 1</DL>
<DD> Определение 1</DD>
<DT> Термин 2</DL>
<DD> Определение 2</DD>
</DL>
```

г)
```
<DL start=1>
<DT> Термин 1</DL>
<DD> Определение 1</DD>
<DT> Термин 2</DL>
<DD> Определение 2</DD>
</DL>
```

Глава 8

Графика на Web-странице

В этой главе...

- Вставка графики в текст
- C:\Мои документы на языке WWW
- Размеры изображения
- Вместо картинок
- Обтекание графики текстом
- Выравнивание по вертикали
- Картина в раме

4 Отступы

4 Форматы графических файлов

Вставка графики в текст

Есть несколько вопросов, по поводу которых среди Web-дизайнеров идут постоянные дискуссии. Некоторые из них нам еще встретятся. А вот первый: что есть картинки для Web-страницы — польза или вред?

"Конечно, польза! — кричат одни, — делать Web-страницы без графики — все равно что назло кондуктору купить билет и пойти пешком!"

"Вред! — надрываются другие, — попробуйте-ка скачать эту графику нашими слабосильными модемами! Заставляя посетителей это делать, мы только набиваем кошельки провайдерам! А если построить на графике весь дизайн страницы, то многие его просто не увидят, потому что благоразумно отключили в своем броузере загрузку картинок!"

Казалось бы, этот спор вот-вот должен кончиться: ведь на смену телефонным линиям активно внедряются более скоростные средства передачи данных. Но уймет ли это спорщиков? Мне думается, вряд ли. Просто по мере повышения пропускной способности линий счет станет вестись не на килобайты, как сейчас, а, скажем, на сотни мегабайт, как это произошло, например, с программным обеспечением, когда вместо 360-килобайтных дискет мы стали пользоваться гигабайтными винчестерами. Которых все равно не хватает...

Впрочем, время покажет. Но есть вещи не менее вечные, чем спор между профессионалами. Одной из них является здоровое желание новичка увидеть на

экране монитора свою фотографию с гордой надписью вроде: "Домашняя страничка Меня, Любимого".

Итак, мы сфотографировались. В парадной шляпе. Но как поместить фотографию на страницу?

Какие есть идеи? "Выделить и вставить" — не пройдет. "Перетащить и опустить" — тоже. Это нам не Word. Это голый текст, и никакие картинки в него не вставляются.

Но нам и не нужно вставлять в код *картинку*. Достаточно указать *ссылку* на нее, чтобы броузер в нужный момент знал, куда за ней обратиться. А уж как вывести картинку на экран, он разберется сам.

Итак, нам нужно указать в HTML-коде две вещи: во-первых, что мы вставляем графический элемент, а во-вторых, *откуда* мы его вставляем. Это делается с помощью дескриптора <IMG> (от английского *image* — "изображение") с параметром src (от английского *source* — "источник"). Попробуем (рис. 8.1)...

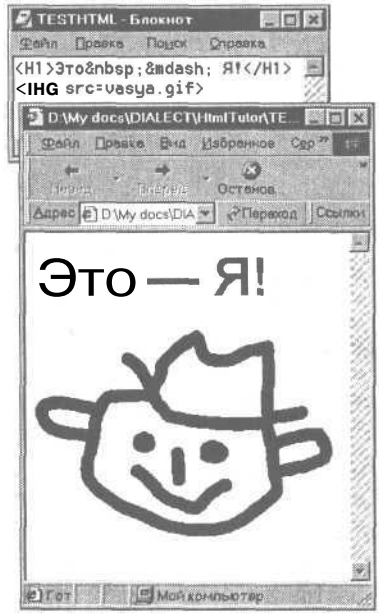


Рис. 8.1. Всего один дескриптор — и дело в шляпе!

Обратите внимание, что дескриптор <IMG> — *одиночный*. Действительно, зачем ему пара? Он только определяет ту точку, куда вставляется содержимое другого файла, и закрывающий дескриптор ему не нужен.

C:\Мои документы **на языке WWW**

Как броузер находит файл с изображением? Очевидно, если файл с картинкой находится в том же каталоге, что и файл с кодом Web-страницы, то — по имени. А если он расположен в другом каталоге? Ведь такое вполне возможно. Более

того, если на странице много разных объектов, именно так обычно и поступают: помещают код в один каталог, а графику, чтобы не путалась под руками, в другой, или даже в несколько.

Казалось бы, все очень просто: нужно только указать полный путь. Попробуем (рис. 8.2)... Ну как, нравится? Лично мне — не очень. Проще было сразу делать страничку без картинок.

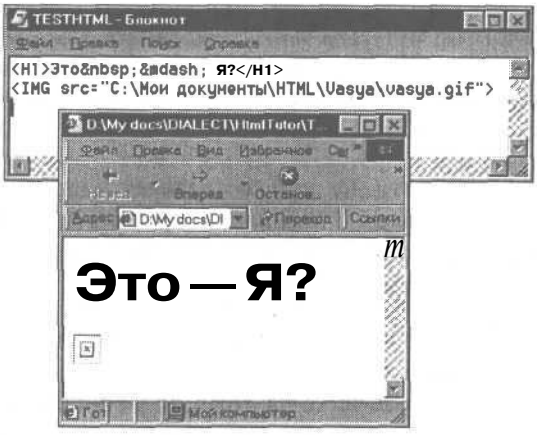


Рис. 8.2. Просто указать путь, как мы привыкли, еще недостаточно для того, чтобы броузер нашел файл

Такова расплата за многолетнюю привычку к операционным системам Microsoft. Ведь это только там путь к файлу обозначается через обратные косые. Internet — не Windows, и нам придется отказаться от многих стереотипов. Любой броузер, даже Internet Explorer, все равно работает по законам Internet. А там действуют правила обозначения пути к файлу, унаследованные из UNIX. Вкратце они таковы.

* Вложенные каталоги перечисляются через прямую, а не через обратную косую, как мы привыкли: `MyHTML/MyGraphics/vasya.gif`.

* Как и в DOS, указывать полный путь не обязательно. Например, если код страницы находится в каталоге MyHTML, а графика — во вложенном в него каталоге MyGraphics, то на картинку можно сослаться так: <IMG src=MyGraphics/vasya.gif>.

4 Именно так, указывая *относительный* (относительно того каталога, в котором находится основной код страницы), а не абсолютный путь (с именем диска и т.п.), и следует поступать, так как вам еще не раз придется копировать все это с одного компьютера на другой. Не менять же из-за этого каждый раз код страницы!

Если название одной из папок состоит из нескольких слов, разделенных пробелом, весь путь нужно заключить в кавычки. Вообще, здесь действует все то же универсальное правило HTML относительно кавычек: если вы не уверены, нужны ли кавычки, — ставьте их.

* Если страница находится на одном компьютере, а картинка — на другом, можно указать полный адрес, например: <IMG src="http://www.vasya.com/MyHTML/MyGraphic/vasya.gif">

* В отличие от дескрипторов HTML, в именах файлов и папок *имеет значение* регистр букв. Так, для многих броузеров и других программ, работающих в Internet, vasya.gif и vasya.GIF — *разные* файлы.

Для того чтобы не пугаться и не запоминать каждый раз, в каком регистре набрано то или иное имя, Web-дизайнеры пользуются неписаным соглашением: все имена папок и файлов состоят *только* из строчных (маленьких) букв.

За этим нужно следить, не только создавая новые папки, но и редактируя графические файлы. В частности, графический процессор Photoshop по умолчанию присваивает файлам расширения, набранные прописными (большими) буквами.

- И последнее. Если вам не нужны лишние проблемы — никакой кириллицы в именах папок и файлов! В частности, если вы пользуетесь русской версией Windows и храните коды страниц в папке Мои документы, рекомендуется разместить папку с изображениями там же, и не польститься на автоматически создаваемую приложениями Office папку Мои рисунки.

Точка единого отсчета

Если картинок много, то их удобнее поместить в отдельный каталог, чтобы они не путались с другими файлами. Но тогда каждый дескриптор, "отвечающий" за вставку изображения, станет длиннее ровно на путь к этому каталогу. Пожертвовать ли краткостью кода в пользу аккуратности или, наоборот, предпочесть краткость?

Как всегда, кроме двух "крайних" решений, есть компромиссное. Оно заключается в указании некоей точки отсчета, или базы, от которой будут "откладываться" все адреса. По умолчанию такой "базой" считается папка, в которой находится текущий HTML-файл. Для того чтобы указать другую базу, используется дескриптор <BASE>. В нем указывается "общая часть" пути всех объектов, на которые в дальнейшем мы будем ссылаться. Например, если все картинки находятся в каталоге MyGraphic, можно воспользоваться таким дескриптором:

```
<BASE href="http://www.vasya.com.ua/MyHTML/MyGraphic">
```

Как это ни удивительно, дескриптор <BASE> — непарный. Область его действия — от той точки, где он помещен, до конца файла или до следующего дескриптора <BASE>, где может быть указан другой исходный путь.

Поэтому дескрипторы <BASE> часто помещают в заголовок HTML-файла. Подробнее об этом читайте в главе 13.

Недостатком дескриптора <BASE> является то, что в качестве "отправной точки", определяемой параметром href, приходится указывать полный URL. В случае переноса Web-страницы, например на другой сервер, этот дескриптор придется редактировать.

Размеры изображения

Можно ли увеличить или уменьшить картинку в окне броузера? Когда картинка вставляется в текстовый документ, например MS Word, ее размеры изменяются с помощью специальных кнопок-манипуляторов, которые появляются, если выделить картинку.

Но ни в броузере, ни, тем более, в Notepad таких средств нет. Как быть?

Общее правило таково: за то, за что в других приложениях "отвечают" средства графического интерфейса, в HTML "отвечают"... Кто?

Правильно, дескрипторы и их параметры. В дескрипторе <IMG> есть два параметра, определяющих ширину и высоту изображения, width и height, соответственно. Задаются эти величины в пикселях. Проведем эксперимент. Не знаю, как у вас, а у меня размер картинки — 300×211 пикселей. Интересно, что получится, если задать параметр width равным, например 100? Каким будет размер картинки? 100×211? Как бы не так: картинка уменьшилась пропорционально так, что ее ширина стала равной 100 пикселям (рис. 8.3). Очевидно, аналогичным образом можно использовать параметр height. Не верите — попробуйте сами.

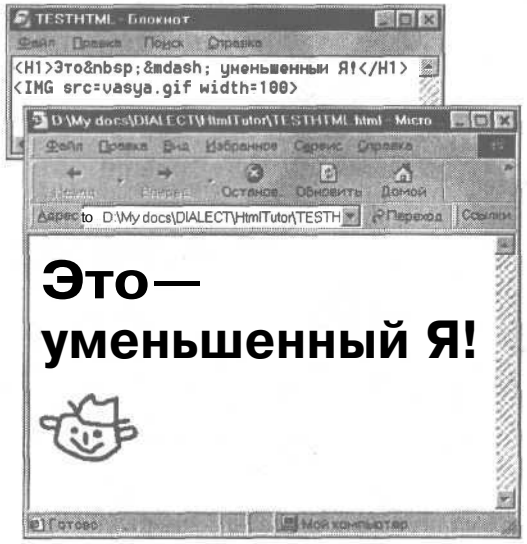


Рис. 8.3. Если задать параметр width или height, то размер картинки изменится пропорционально

Однако нужно отметить, что это, пожалуй, не лучшее применение параметров width и height. Почему? Ведь они, кажется, для того и созданы?

Дело в том, что при этом действительные размеры графического файла, который приходится загружать в сети, не изменяются: меняется только размер того, что выводится на экран, но не количество перекачиваемых килобайтов. Если же вы хотите, чтобы скорость загрузки соответствовала качеству получаемой картинки, нужно уменьшить картинку в графическом редакторе, например в Photoshop.

Но ведь зато, скажете вы, с помощью этих параметров можно сделать маленькую картинку, а на экране увеличить ее?

Можно, конечно. Но результат далеко не всегда оказывается удовлетворительным. Если в этом растровом изображении окажется хоть одна негоризонтальная и невертикальная линия, то при увеличении вам не избежать зубчатых краев, которые подчас выглядят весьма неопрятно (рис. 8.4).

Рис. 8.4. Если увеличить картинку, изменив параметр width или height, станут видны зубчатые края. Красиво ли это?

Зато есть другие, весьма полезные способы применения параметров height и width.

Первое, что приходит в голову, — то, что с помощью этих параметров можно менять пропорции изображения, растягивать его в длину и ширину. Этим можно добиться довольно любопытных эффектов (рис. 8.5). Обратите внимание: в коде, показанном на рисунке, дважды использовано одно и то же изображение.

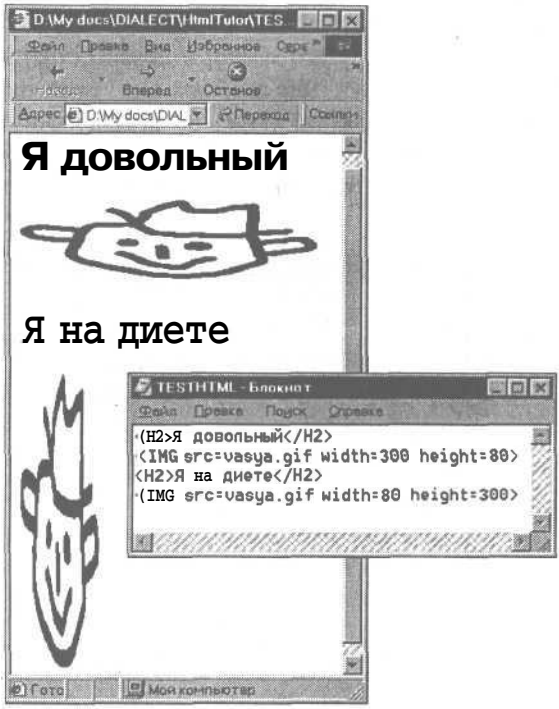


Рис. 8.5. Непропорциональное изменение параметров height и width позволяет добиться любопытных эффектов с одним и тем же изображением

Но главное — с помощью этих параметров можно задать точные габариты изображения. Зачем, ведь они и так соблюдаются? А вот посмотрим.

Сделаем так, чтобы броузер не нашел картинку. Для этого изменим имя файла в коде, например на no.gif. Что мы увидим вместо картинки? Маленький пустой квадратик, как на рис. 8.2. А теперь укажем в дескрипторе `<IMG>` точные параметры предполагаемого изображения. Что мы видим теперь? Прямоугольник, размеры которого точно соответствуют нашей картинке (рис. 8.6).

Теперь представьте себе, что происходит со страницей и картинкой, которые вы закачиваете из Internet на свой компьютер по медленному модему. Что загружается и появляется на экране первым? Очевидно, то, что важнее, что несет больше информации. Обычно это текст. Заодно он и загружается быстрее. Пока загрузится все остальное, посетитель страницы, вместо того чтобы сидеть без дела, сможет его почитать.

А как картинки? Пока броузер их не получил, он будет выводить вместо них те самые прямоугольники, которые мы видели в нашем эксперименте. Когда броузер получит картинки, он поставит их вместо этих прямоугольников.

Теперь представим себя на месте дизайнера такой страницы: мы разметили текст и графику в нужных местах так, чтобы они гармонично и пропорционально дополняли друг друга. И тут вместо картинок разного размера появляются одинаковые квадратики, а текст съезжает туда, куда заблагорассудилось броузеру...

Приятного мало. Если же указать габариты изображения в дескрипторе <IMG>, то броузер заранее "зарезервирует" под него место, и дизайн страницы не пострадает.

И еще один интересный момент. Что происходит, если окно броузера слишком мало, и текст в нем не помещается? Ответ на этот вопрос мы уже знаем: если текст не помещается по ширине, то он обычно переносится на другую строку, а если это не получается или если текст не помещается по высоте, — появляются полосы прокрутки.

Рис. 8.6. Если броузер не находит графический файл, но в коде указаны размеры изображения, он показывает прямоугольник соответствующих размеров

То же самое происходит и в случае, если в окне не помещается картинка. Но есть способ, позволяющий регулировать размеры картинки в зависимости от размеров окна броузера. Это — задание ширины и высоты изображения *в процентах*. Если, например, задать width=100%, то при изменении ширины окна размер изображения также будет меняться, и оно всегда будет "вписываться" в окно независимо от его размеров (рис. 8.7).

Обратите внимание, что если в процентах указана ширина изображения, а окно броузера изменяется по высоте, то размер картинки меняться не будет. И наоборот, если задать в процентах высоту изображения, а менять ширину окна броузера, размер изображения также не изменится.

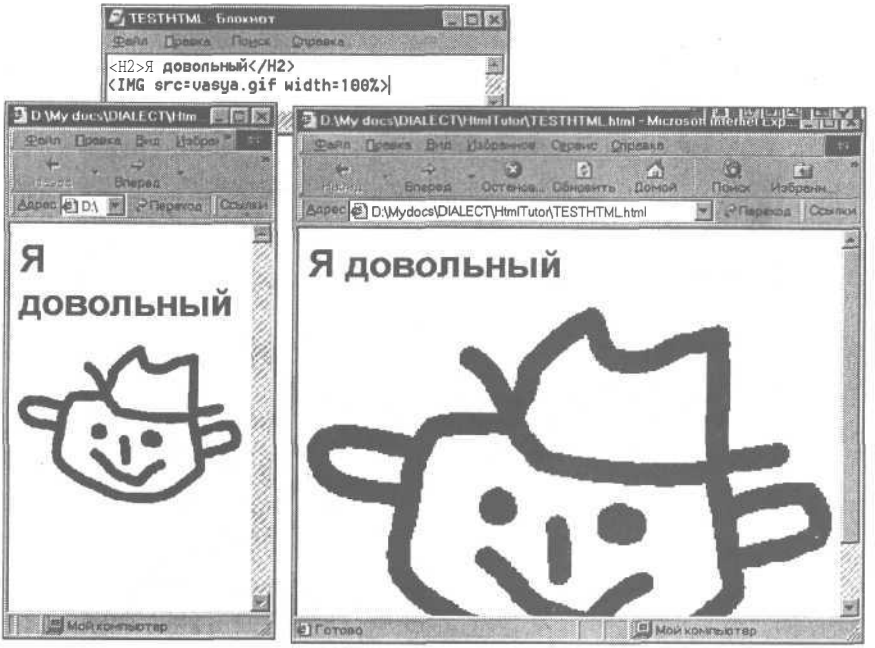


Рис. 8.7. Если задать ширину изображения в процентах, то размер изображения будет изменяться в зависимости от ширины окна

Вместо картинок

То есть как это вместо? Мы же вроде решили графикой заняться?

Вы можете считать это безобразием, но факт остается фактом: немало посетителей нахально отключают загрузку картинок, жалея потратить несколько жалких копеек на нашу замечательную графику. Они не ценят прекрасного... Но ведь нужно же как-то и до них донести то, что изображено на странице! Пускай они хотя бы пожалеют, что не видят такой красоты.

Как это сделать?

Конечно, можно снабдить все рисунки подписями на манер тех, что сопровождают иллюстрации в нашей книге. Но такое возможно далеко не всегда. Как быть?

На этот случай у дескриптора <IMG> есть атрибут, позволяющий вывести вместо картинки надпись — в качестве альтернативы. Он так и называется — alt. Если же пользователь не отключил отображение графики, эта надпись выводится в виде контекстной подсказки (рис. 8.8).

Но бывает так, что посетитель не отключал вывод картинок, а просто у него медленный модем или у нас слишком большая картинка. Если мы не хотим, чтобы этот посетитель все-таки отключил вывод графики, можно немного скрасить ему ожидание, предложив временно, пока загрузится большая и красивая картинка, полюбоваться другой, маленькой и не такой красивой. Для этого используется параметр lowsrc (от английского *low source* — здесь: "источник более низкого качества"):

```
<img src=vasya.gif alt="Это мой автопортрет" lowsrc=smallvasya.gif>
```

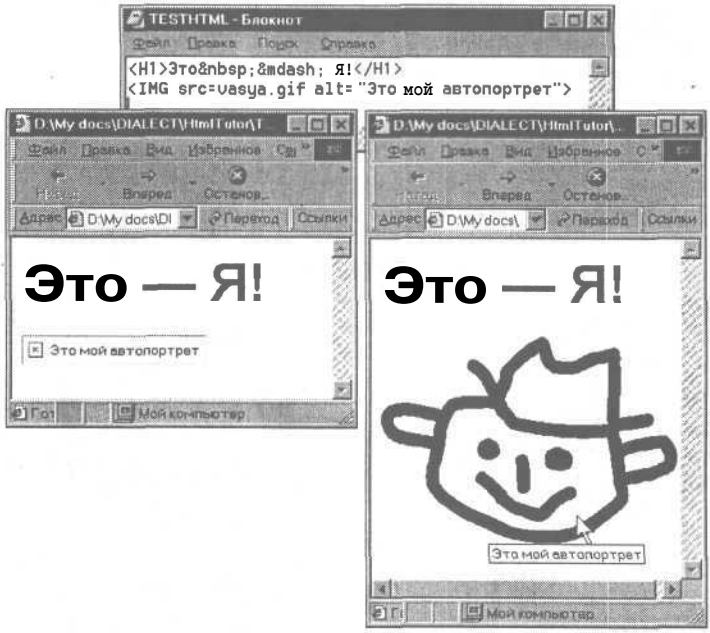


Рис. 8.8. Текст, указанный в качестве значения параметра alt, выводится вместо картинки или в качестве контекстной подсказки

Обтекание графики текстом

Справа от автопортрета самое место автобиографии. Попробуем ее туда вставить (рис. 8.9). Как видим, вышло не совсем то, что нам нужно. Это похоже на вставку картинки в MS Word: если в той же строке есть текст, то получается не совсем красиво.

Помнится, для выравнивания текста использовался атрибут align. Правда, там он применялся просто для выравнивания строчки по горизонтали относительно страницы. С графикой дело обстоит несколько иначе. Тем не менее, попробуем. Мы хотим, чтобы рисунок располагался слева. Так и напишем: align=left. Что получилось? Как раз то, что нужно (рис. 8.10).

Действительно, это удобно. Если текста мало — он размещается справа от картинки. А если его становится много, то он обтекает ее, и "не поместившиеся" строчки выводятся как обычно, начиная с левого края окна.

Не следует путать параметр align в дескрипторе <IMG> с одноименным параметром в дескрипторах <P> и <Hл>. В последних он управляет выравниванием *текста* по горизонтали, а в дескрипторе <IMG> — выравниванием *изображения,* причем как по горизонтали, так и по вертикали.

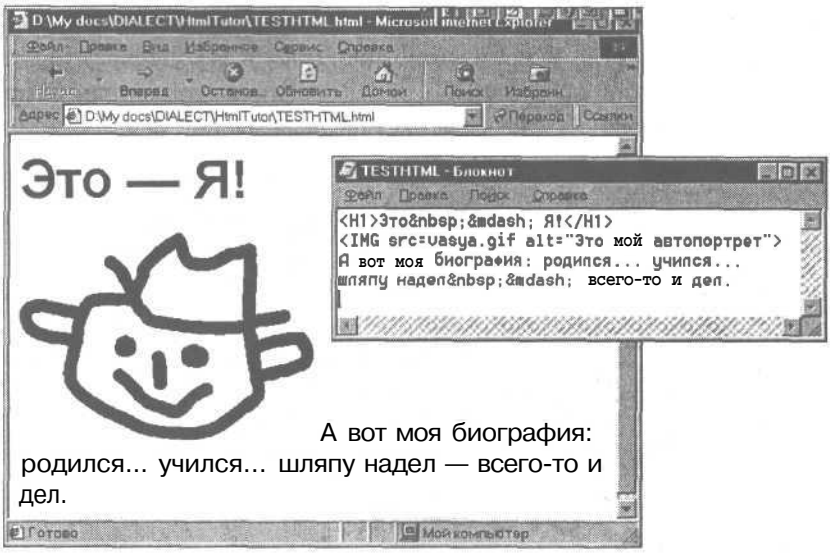


Рис. 8.9. Попытка разместить текст справа от картинки "в лоб" не вполне удалась

Рис. 8.10. Для обтекания картинки текстом используется атрибут **align** дескриптора `<IMG>` — тот же, что и для выравнивания текста

А как быть, если справа от картинки еще достаточно места, но текст нужно вывести не там, а под ней?

Идея номер один: поместить текст в отдельный абзац. В MS Word подобный прием работает. А в HTML? К сожалению, нет (рис. 8.11). Дескриптор **<P>** отменяет выравнивание текста относительно окна, заданное параметром align в предыдущем дескрипторе **<P>,** но не отменяет выравнивание *изображения* относительно текста, заданное в дескрипторе <IMG>.

Рис. 8.11. Для того чтобы изменить тип обтекания изображения текстом, дескриптор <P> не подходит

Для того чтобы изменить способ обтекания изображения текстом, используют уже знакомый нам дескриптор <BR>. Только одного этого дескриптора теперь недостаточно. Нам потребуется его параметр clear. Этот параметр позволяет отменить предыдущий режим выравнивания. Так, если присвоить параметру clear значение all, следующая строка начнется под картинкой, даже если сбоку от последней еще остается место (рис. 8.12). На случай если рисунков много, причем одни прижаты к правому краю окна, а другие — к левому, предусмотрены еще два значения этого параметра: если clear=right, следующая строка начнется в том месте, где правый край окна броузера свободен от рисунков, а если clear=left — то там, где от рисунков свободен левый край окна.

Рис. 8.12. Для того чтобы текст начинался с новой строки, нужно воспользоваться дескриптором
 с параметром clear

Выравнивание по вертикали

Есть еще одно любопытное применение картинок — внутри текста, где они иногда заменяют отсутствующие в распространенных шрифтах символы. Так, благодаря распространению Internet стали популярными значки, изображающие эмоции, — так называемые "смайлики". Первоначально смайлики рисовали с помощью уже стандартных символов, которые есть на любой клавиатуре: :-). Переверните страницу на 90° — и увидите улыбающуюся рожицу.

Со временем улыбающиеся человечки стали появляться в разных шрифтах, а на Web-сайтах, особенно на Internet-форумах, стали широко применяться не только улыбающиеся и грустные рожицы, но и раскрашенные во всевозможные цвета. Эти рожицы, а также другие мелкие графические элементы, иногда бывает полезно выровнять *по вертикали* относительно строки. Для выравнивания изображений по вертикали используется уже знакомый нам атрибут align, которому присваиваются значения absbottom, absmiddle, baseline, bottom, middle, texttop и top.

Не многовато ли?

Действительно, много. И не потому, что строка узкая. А потому, что эти семь параметров описывают *пять* положений изображения относительно строки (рис. 8.13). Значение absbottom соответствует выравниванию по нижним высту-

пающим элементам букв в строке, baseline и bottom — по нижнему краю строки, absmiddle и middle — по середине, texttop — по верхнему краю и top — по верхним выступающим элементам.

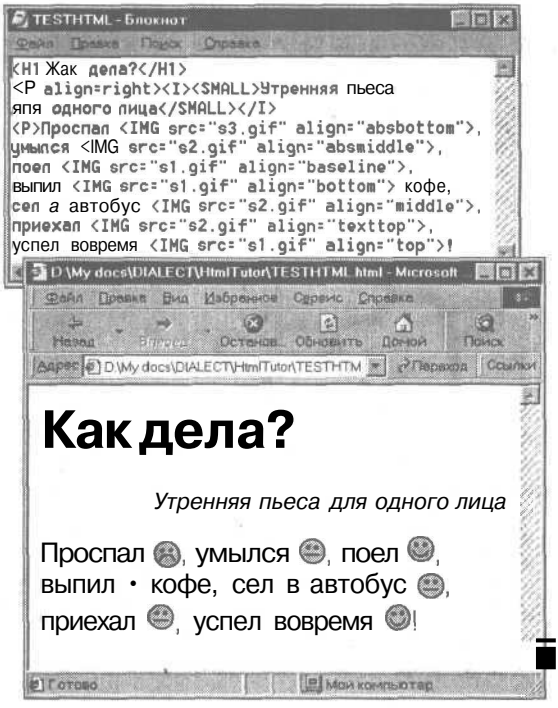


Рис. 8.13. С помощью параметра align можно выравнивать изображение по вертикали относительно строки текста

Картина в раме

Как отделить картинку от текста? Правильно — заключить в рамку.

Для этого вовсе не обязательно рисовать рамку в графическом редакторе. Если речь идет о простой черной рамке (только, надеемся, не очень толстой!), то гораздо проще и быстрее воспользоваться параметром border дескриптора <IMG>. Значение этого параметра определяет толщину рамки в пикселях (рис. 8.14).

Если фон изображения отличается от фона всей страницы, тонкий разделитель между ними смотрится особенно уместно.

Рис. 8.14. Параметр border определяет толщину рамки в пикселях

Изображения, которые "по совместительству" являются гиперссылками, по умолчанию заключаются в рамку толщиной 2 пикселя. Это часто выглядит неаккуратно и не к месту. Поэтому в таких случаях параметру border в дескрипторе <IMG> специально присваивают значение 0:

```
<img src=button.gif border=0 alt="Кнопка без рамки">
```

Подробнее о гиперссылках читайте в главе 9.

Отступы

Вы наверняка обратили внимание на то, что текст на рис. 8.14 разместился чересчур близко от края изображения. Это, пожалуй, не совсем красиво. Было бы лучше отступить на пару миллиметров.

Как это сделать?

Конечно, можно, поплевав на руки, открыть Photoshop и внести соответствующие изменения в графический файл: собственноручно нарисовать рамку, добавить отступы... Не знаю, как вам, а мне из-за таких мелочей редактировать графику лень. А при мысли, что это придется делать каждый раз, как захочется увеличить или уменьшить отступы, просто мурашки бегут по затекшей спине. И наконец, главное: всегда, когда что-то переносится из HTML-кода в графический файл, увеличивается размер последнего, а значит, и время загрузки всей страницы.

Гораздо проще задать отступы в HTML-коде. Для этого в дескрипторе <IMG> предусмотрены параметры vspace и hspace, определяющие в пикселях расстояние между изображением и текстом по вертикали и горизонтали, соответственно (рис. 8.15).

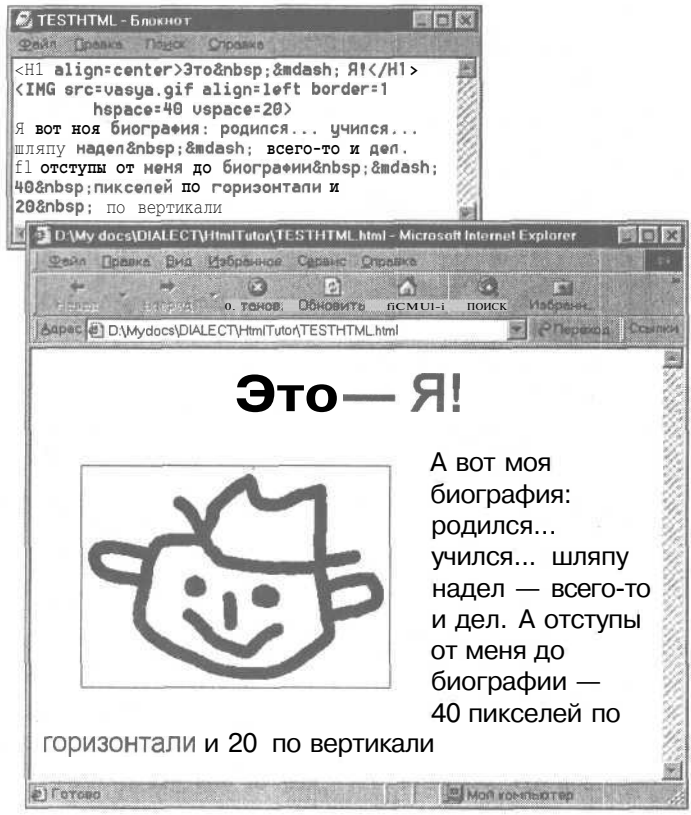


Рис. 8.15. Параметры vspace и hspace задают отступы до текста по вертикали и горизонтали, соответственно

Форматы графических файлов

Графических форматов существует множество. Каждая уважающая себя компания — разработчик пакета по обработке графики — считает своим долгом создать собственный формат. Многие из этих форматов оказались настолько удобными, что превратились в стандарты, негласно принятые в тех или иных областях, где применяется графика.

Здесь идет речь о форматах *растровой* графики. При этом изображение рассматривается как прямоугольная матрица, состоящая из точек разного цвета — что-то наподобие узора для вышивания крестиком.

Есть и другой способ сохранения изображений — *векторный*. При этом изображение рассматривается как набор геометрических фигур, обладающих определенными свойствами, — координатами, формой контура, цветом контура и внутренней области и т.п.

У каждого из этих способов хранения изображения — векторного и растрового — есть свои преимущества и недостатки. Так, растровые форматы обеспечивают более естественное, — а значит, и более компактное — сохранение таких изображений, где сложно выделить определенные геометрические объекты, где все решает переход цвета. Это относится, например к фотографиям. Векторные форматы, наоборот, более эффективны при сохранении рисунков и чертежей, т.е. таких изображений, которые сформированы набором линий, цветовых областей с четко очерченными краями и других объектов, которые можно легко описать математически.

Так сложилось, что в электронной полиграфии прижились главным образом растровые форматы. Все попытки создать универсальный, общепринятый векторный формат для Web пока привели к весьма ограниченным результатам, таким как Flash.

См. также Лит Г., Финкельштейн Э. *Macromedia Flash MX для "чайников"*. К.: Диалектика, 2002.

В Internet главным образом используются два растровых графических формата — GIF (Graphic Interchange Format), первоначально разработанный компанией CompuServe, и JPEG, созданный Объединенной группой экспертов по фотографии (Joint Photographic Experts Group). Им соответствуют файлы с расширениями *.gif и *.jpg. У каждого из этих форматов есть своя область применения, в соответствии с его особенностями. Рассмотрим эти особенности подробнее.

Сжатие. В обоих форматах — и в GIF, и в JPEG — используется внутреннее сжатие изображений. Для того чтобы в этом убедиться, достаточно сжать файл любого из этих форматов с помощью программы-архиватора. Размеры архива и исходного файла практически не отличаются. Нетрудно догадаться, что именно поэтому данные форматы были выбраны для представления графики в Internet: зачем нагружать линии связи лишними байтами?

Но алгоритмы сжатия в форматах GIF и JPEG отличаются. Изображения в файлах GIF сжимаются без потерь информации, в то время как в JPEG реализовано сжатие с потерями. В результате файл JPEG может оказаться втрое меньше файла GIF с тем же изображением.

Количество цветов. В формате GIF количество цветов не может превышать 256. При этом, если в рисунке используется меньшее количество цветов, например 2 — черный и белый, — то информация об остальных 254 цветах не сохраняется. В JPEG допускаются полноцветные изображения с 16,7 млн цветов. Но если в рисунке цветов меньше, то он все равно преобразуется в полноцветное изображение, и для каждого пикселя сохраняется информация о наличии/отсутствии каждого цвета. Поэтому, как правило, изображения, состоящие из небольшого количества цветов, — рисунки, чертежи — сохраняются в формате GIF, а изображения, где важно сохранить всю палитру цветов, например фотографии, в формате JPEG.

Кроме того, в отличие от палитры JPEG, одному из 256 цветов палитры GIF присвоено значение "прозрачного". Это очень полезное свойство широко используется Web-дизайнерами.

Первое, очевидное его применение — когда нужно разместить на странице непрямоугольное изображение, фон которого отличается от фона страницы. Такое изображение помещается в файл GIF с прозрачным фоном — и задача решена.

Кроме того, прозрачный "цвет" нашел в Web-дизайне еще одно применение: в виде GIF-файла, в котором хранится 1 "прозрачный" пиксель.

Какая польза от такой, с позволения сказать, картинки?

Внешность, — а в данном случае, отсутствие всякой внешности — обманчива. Однопиксельные прозрачные GIF-изображения применяются в Web-дизайне сплошь и рядом — везде, где дизайнер не хочет полагаться на порой ненадежные средства стандарта HTML. Мы еще не раз с ними встретимся, а пока приведем только два примера такого применения.

1. **Точный отступ между строчками.** Как известно, в HTML существует два варианта отступа: большой (между абзацами) и поменьше (между строчками). Если мы хотим сами регулировать это расстояние, можно воспользоваться прозрачным GIF, "растянув" его в высоту, насколько нужно (рис. 8.16).

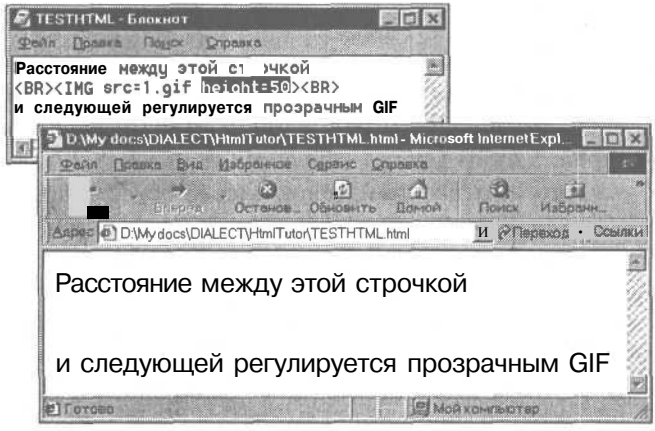


Рис. 8.16. С помощью прозрачного GIF можно регулировать расстояние между строками

2. **Красная строка.** Как известно, дескриптор <P> не предусматривает принятой в отечественной полиграфии "красной строки" в начале абзаца (см. главу 2). Однако реализовать соответствующий отступ можно с помощью прозрачного GIF, "растянув" его в ширину на соответствующее количество пикселей (рис. 8.17).

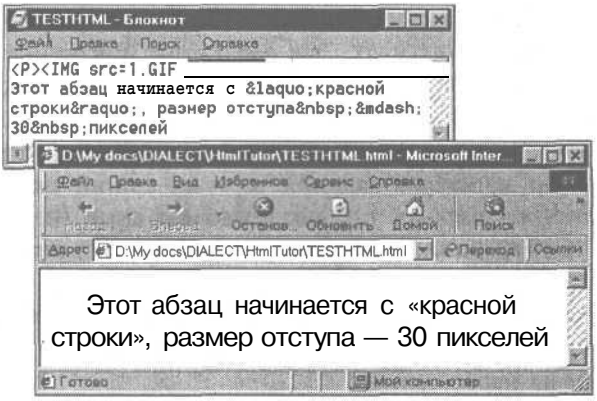


Рис. 8.17. Абзац с "красной строкой", выполненный с помощью прозрачного GIF

Анимация. В отличие от JPEG, формат GIF позволяет сохранить в одном файле несколько изображений, снабженных управляющими блоками. Это широко используется для создания небольших анимированных картинок.

Анимированные GIF-изображения широко используются в баннерах и прочей Internet-рекламе. Кроме того, на Web-страницах, посвященных развлечениям, много анимированных кнопок, маркеров и других художественных элементов. В Internet есть целые хранилища таких готовых аксессуаров. Однако пользоваться анимацией следует осмотрительно: мелкие мельтешащие объекты отвлекают внимание посетителя от того основного дела, ради которого он пришел на вашу страницу, и довольно часто вызывают раздражение. Лучше пользоваться таким правилом: анимацию, как и любой другой броский элемент, следует использовать только тогда, когда на то есть серьезные причины.

Тем, кто решит серьезно заняться графикой для Internet, рекомендую изучить книгу Мак-Клелланд Д., Обермайер Б. *Photoshop 7 для "чайников"*. К.: Диалектика, 2002.

Резюме

Для размещения графических элементов на Web-странице используется следующая технология: изображение сохраняется в отдельном файле, а в HTML-код вставляется ссылающийся на него дескриптор <IMG>. Свойства изображения, размещаемого на странице, определяются параметрами дескриптора <IMG>.

Имя файла, в котором находится изображение, и путь к нему определяются параметром src. Задавая путь, нужно руководствоваться правилами, принятыми для записи URL. Эти правила отличаются от правил записи пути к файлу в ОС Windows. В частности, имена папок и файлов разделяются не обратными, а прямыми косыми, а строчные и прописные буквы различаются. Кроме того, в именах файлов, передаваемых по Internet, не рекомендуется использовать пробелы и кириллицу.

Ширина и высота изображения определяются параметрами width и height, соответственно, и задаются в пикселях. Эти параметры можно использовать как для изменения параметров изображения, так и для того, чтобы определить их заранее. Первый способ применения параметров width и height используется достаточно редко, так как при увеличении маленьких изображений ухудшается их качество, а уменьшение приводит к нерациональной загрузке линий связи. Второй способ применяется достаточно широко, так как позволяет сразу, не дожидаясь окончательной загрузки изображений, разместить элементы Web-страницы в соответствии с замыслом дизайнера.

Часто изображения снабжаются комментирующими надписями, указанными в качестве значения параметра alt. Эти надписи появляются рядом с указателем мыши, наведенным на изображение.

Для определения способа обтекания графики текстом используется параметр align. Этот параметр описывает выравнивание не только по горизонтали — по левому краю (значение left) или по правому краю (значение right), — но и по вертикали относительно той строки, в которой размещено изображение. Значение absbottom соответствует выравниванию по нижним выступающим элементам букв в строке, baseline и bottom — по нижнему краю строки, absmiddle и middle — по середине, texttop — по верхнему краю и top — по верхним выступающим элементам. Для отмены обтекания изображения текстом используется дескриптор
 с параметром clear, который принимает значения right (отмена обтекания справа), left (отмена обтекания слева) и all (отмена всех типов обтекания)

Для отделения изображения от текста используются два средства — рамки и отступы. Параметр border определяет толщину рамки в пикселях. Цвет такой рамки всегда черный. Для создания более сложных рамок следует использовать графический редактор. Горизонтальный и вертикальный отступы между изображением и текстом задаются в пикселях с помощью параметров hspace и vspace.

Изображения для Internet обычно хранятся в файлах форматов GIF и JPEG. Выбор этих форматов для Internet определяется компактностью хранения информации: она хранится там в сжатом виде и, следовательно, при ее передаче нагрузка на линии связи меньше, чем при передаче несжатых изображений. В формате JPEG сохраняются полноцветные изображения, в частности качественные фотографии. В формате GIF хранятся изображения с ограниченным количеством цветов, а также анимированные изображения и изображения с прозрачным фоном. Кроме того, в Web-дизайне широко применяется GIF-изображение, представляющее собой 1 пиксель прозрачного "цвета". С его помощью можно, в частности, регулировать расстояние между строками, а также создавать отступы заданной величины.

Тесты

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а) <IMG src="Мои flOKyneHTbt\vasya.gif">

   б) <IMG src="Mydoc\vasya.gif">

в) <IMG src="Mydoc/vasya.gif">

г) <IMG src=Mydoc/vasya.gif>

д) <IMG src="Mydoc/vasya.tif">

е) <IMG src="Mydoc/vasya.GIF">

2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <IMG src=vasya.gif border=l bordercolor=red>

б) <IMG src=vasya.gif border=l>

в) <IMG src=vasya.gif border=l,5>

г) <IMG src=vasya.gif border=1.5>

д) <IMG src=vasya.gif border=red>

е) <IMG src=vasya.gif border=0>

3. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <IMG src=vasya.gif alt=vasya.jpg>

б) <IMG src=vasya.gif alt=vasya>

в) <IMG src=vasya.gif alt="Это Вася">

г) <IMG src=vasya.gif alt=Это Вася>

д) <IMG src=vasya.gif alt=Eto Vasya>

4. Для того чтобы текст обтекал изображение справа, используется следующий код.

а) <IMG src=pict.gif align=right> текст

б) <IMG src=pict.gif align=left> текст

в) <IMG src=pict.gif> текст

г) <IMG src=pict.gif align=left><BR clear=right> текст

д) <IMG src=pict.gif align=left><BR clear=left> текст

5. Для того чтобы текст обтекал изображение слева, используется следующий код.

а) <IMG src=pict.gif align=right> текст

б) <IMG src=pict.gif align=left> текст

в) <IMG src=pict.gif> текст

г) <IMG src=pict.gif align=left><BR clear=right> текст

д) <IMG src=pict.gif align=left><BR clear=left> текст

6. Для того чтобы изображение разместилось по середине строки, нужно написать следующий код.

а) <IMG src=pict.gif align=center>

б) <BR clear=center><IMG src=pict.gif>

в) <IMG src=pict.gif><BR clear=center>

г) <P align=center><IMG src=pict.gif>

Глава 9

Гипертекстовые ссылки

В этой главе...

- Точки входа в гипертекст
- "Якоря" в море Internet
- 4 Закладки
- Ссылки, которые не являются ссылками
- Ссылки-картинки
- Виртуальная навигация

Точки входа в гипертекст

Кто не знает, что такое гиперссылка? На страницах Internet ее видели все. И не только видели, но и не раз "щупали" мышью. Так не дадите ли определение сего явления? Непросто, правда?

Действительно, понятное и исчерпывающее "энциклопедическое" определение гиперссылки встречается нечасто. Вот, например, одно такое определение.

Гиперссылка (hyperlink). Указатель внутри гипертекстового документа, указывающий на другой документ (связывает с другим документом), который также может быть гипертекстовым.

А теперь давайте разбирать, что все это значит. Что такое указатель? Любой программист поймет... А не программист? Возможно, поймет интуитивно: раз указатель — значит, на что-то указывает. Но это все равно что сказать: раз ссылка — значит, на что-то ссылается. Раз "гипер" — значит, на гипертекстовый документ. Вот и все.

Но как раскрыть в сухом определении всю прелесть этого маленького элемента Web-страниц? Мгновенный (в идеале) переход из некторой точки одного электронного документа в соответствующую ему точку другого такого же документа чем-то напоминает гиперпространственные порталы, по которым герои фантастических романов переходят из одной главы в другую...

Как выглядит гиперссылка? Это знает любой, кто провел в Internet хотя бы пятнадцать минут: указатель мыши, поменявший форму, — была стрелка, стала рука с вытянутым указательным пальцем. Если щелкнуть на том месте, на которое показывает палец, то обязательно что-нибудь произойдет: откроется новая страница, появится окно, в котором можно написать электронное письмо, начнет закачиваться файл...

Естественно предположить, что код, "отвечающий" за гиперссылку, должен состоять из двух частей: одна определяет "точку входа" (место, откуда происходит переход), а вторая — "точку выхода", или то, что происходит, если щелкнуть в "точке входа".

Так и есть. "Точкой входа" является фрагмент исходной страницы — текст, графика или то и другое вместе, размеченное дескриптором <A>, а "точку выхода" определяет параметр дескриптора <A>, внутри которого заключен этот фрагмент.

"Якоря" в море Internet

Да-да, настоящие якоря. Дело в том, что название дескриптора <A>, с помощью которого создаются гиперссылки, происходит от английского слова *anchor* — "якорь". В самом деле, если присмотреться повнимательнее, что-то в этом есть: этот дескриптор как бы цепляется за объект, на который указывает ссылка. И вот уже страница надежно связана с этим объектом, даже если последний находится от нас за сотни серверов и тысячи километров кабелей...

Дескриптор <A> — парный. Все, что находится между <A> и </A>, является "якорем", гиперссылкой. За что же цепляется якорь? Что служит надежным признаком "точки выхода"?

В первую очередь, конечно, это имя файла. Его указывают в параметре href дескриптора <A>. Вообще, дескриптор <A> без параметров, очевидно, вещь бессмысленная — как якорь без лап...

Рис. 9.1. Гиперссылка всегда состоит из двух частей — "точки входа" и "точки выхода"

Итак, для того чтобы одна Web-страница ссылалась на другую, в коде первой должен содержаться примерно такой фрагмент:

```
Хотите увидеть, как работает гиперссылка?
Щелкните <A href="page2.html"> здесь!<A>
```

И тогда мы увидим то же, что и на рис. 8.1, разумеется, при наличии файла page2.htm.

Если файл, на который указывает гиперссылка, находится в другом каталоге или на другом компьютере, нужно указывать путь к нему по правилам, описанным в главе 8.

Закладки

Действительно, гиперссылка — могучее средство. В сущности, это готовый инструмент, решающий одну из самых важных задач в любой книге, будь она бумажная или электронная, — задачу перехода в нужную точку из оглавления или предметного указателя. В бумажной литературе, кстати, последняя задача так и не решена удовлетворительно: найти по ссылке в предметном указателе страницу, где объясняется нужное понятие, еще можно, но отыскать на странице сам термин бывает непросто. И, в любом случае, это довольно скучная и долгая операция.

То ли дело гипертекстовая ссылка! Переход по ней происходит мгновенно и сразу в нужную точку.

Стоп.

В нужную точку? До сих пор мы выяснили только, как перейти на нужную *страницу*. Конечно, можно разбить электронную книгу так, чтобы каждая глава размещалась в отдельном файле. (Кстати, именно так очень часто и поступают, чтобы уменьшить размер загружаемого файла.) Но как поступить, если мы ищем некий термин, описываемый в середине главы? Понадеяться на встроенную в броузер функцию контекстного поиска?

Конечно, можно поступить и так. Но это значит — заставить посетителя страницы каждый раз вручную вводить искомый термин в окно поиска. Верьте мне на слово: ему это надоест, и очень быстро. И тогда он помянет вас словом — не обязательно тихим и вряд ли незлым...

Как организовать переход не только на нужную страницу, но в нужную *точку* страницы? В текстовых редакторах для этого предусмотрен специальный инструмент — *закладки*. То есть некие именованные метки, по которым можно отыскать соответствующую им точку документа.

В HTML такие закладки тоже есть. По реализации они очень похожи на метки, используемые программистами для переходов внутри программы. Для перехода по такой закладке в HTML-коде требуется создать два объекта.

В "точке выхода" создается закладка, которой присваивается какое-нибудь имя. Это имя может содержать любые латинские буквы, а также цифры. Например, метку для перехода в начало страницы можно назвать begin, а для перехода в конец — end. В соответствующих точках кода страницы ставятся дескрипторы <A> с параметром name, которому присвоены эти значения:

```
<!-- начало страницы>
<A name=begin>
...
<A name=end>
<!-- конец страницы>
```

Лучше выбирать меткам осмысленные имена. Броузеру все равно, но вам потом будет проще читать и редактировать код.

Теперь, если мы хотим сослаться не просто на файл, а на конкретное место этого файла, нужно, кроме имени файла, указать имя закладки. По правилам HTML разделителем между этими именами служит знак #:

```
<A href="page2.html#begin">Перейти в начало страницы page2.html</A>
<A href="page2.html#end">Перейти в конец страницы page2.html</A>
```

Обратим внимание: в наших руках теперь средство, позволяющее "перепрыгивать" не только с одной страницы на другую, но между разными точками одной и той же страницы. В последнем случае указание имени файла в ссылке становится излишним, и его можно опустить. Если переход по ссылке происходит в пределах одного файла, достаточно ограничиться указанием только имени метки:

```
<A href="#begin">Перейти в начало текущей страницы</A>
```

Проведя достаточно времени в Internet, вы заметите, что ссылки ведут себя по-разному: в одних случаях открываются в том же окне, в других — в новом. Такое поведение гиперссылок определяется параметром target. Подробнее мы поговорим о нем, когда займемся фреймами (см. главу И). А пока что обратим внимание на два его предопределенных значения - _blank и _self (обязательно со знаками подчеркивания впереди!). Если в дескрипторе <A> указан параметр target=_blank, то объект, на который указывает эта гиперссылка, всегда открывается в новом окне. Если же target=_self, то объект открывается в том же окне, что и документ, содержащий гиперссылку. Последний режим используется по умолчанию, так что указывать его специально не обязательно.

У значения _blank параметра target есть свои преимущества и недостатки. Как показывает практика, в большинстве случаев посетители страниц предпочитают сами решать, в каком окне открывать новую страницу, — в новом или в уже существующем. И это несмотря на то, что, как правило, они все же предпочитают новое окно! Такой вот парадокс... Но иногда использование значения _blank удобно и оправдано, например, когда на странице много ссылок. Такие ситуации возникают, в частности, на страницах поисковых серверов.

Ссылки, которые не являются ссылками

Да, такой вот парадокс: гипертекстовая ссылка далеко не всегда ссылается на гипертекстовый документ. На самом деле в качестве значения параметра href можно указать ссылку на любой объект. Если броузер распознает в нем документ, который он может открыть для просмотра, то он так и поступит. В противном случае он просто предложит вам загрузить этот файл. Броузеры последних версий позволяют просматривать (разумеется, если на компьютере установлено соответствующее приложение) документы MS Word, MS Excel, Adobe Acrobat и др.

Мы рассмотрим несколько наиболее интересных и широко распространенных способов применения гиперссылок не совсем по прямому назначению.

Загрузка файлов. Если значением параметра href является имя файла, который броузер не может открыть (например, zip-архив), то броузер предлагает загрузить этот файл из Internet и сохранить на диске, чтобы пользователь потом сам решил, что с ним делать. Если на компьютере установлена специальная программа для загрузки файлов, такая как Go!zilla или GetRight, броузер передает управление ей. В противном случае он загружает файл самостоятельно. Выглядят подобные ссылки примерно так:

```
<A href=price.zip> Скачать прайс-лист </A>
```

Этим свойством гиперссылок широко пользуются для передачи через Internet самой разной информации. Обычно в целях ускорения загрузки она выкладывается на сайты в виде архивов. Хотите попробовать сами? Тогда загляните на любой сайт бесплатного ПО. Кстати, можете поискать там подходящую программу для загрузки файлов.

Советую загрузить из Internet и установить одну из таких программ. Большинство из них — недорогие условно-бесплатные продукты. Средства загрузки, встроенные в броузеры, обычно представляют собой весьма примитивные функции. По качеству они отличаются от специализированного ПО примерно настолько же, насколько Notepad отличается от специализированных HTML-редакторов. Загрузчик броузера, как правило, не справляется с такими проблемами, как нестабильная связь или разрыв соединения (что на наших линиях случается частенько). Специализированные программы позволяют оптимизировать скорость загрузки, продолжить загрузку в случае разрыва соединения, запланировать загрузку на определенное время, а также имеют множество других полезных функций.

Ускоренная отправка писем. Редкий сайт обходится без ссылки вроде "Написать Web-мастеру". По ней Web-мастер желает получать письма от благодарных посетителей и будущих заказчиков, а получает вирусы и горы спама. И тем не менее упрямо ставит в углу главной страницы ссылку на свой почтовый ящик: как ни крути, а без обратной связи не обойтись. Чтобы ее организовать, используется гиперссылка, где значением параметра href является адрес электронной почты, перед которым стоит ключевое слово mailto: (рис. 9.2).

Если щелкнуть на такой ссылке, запускается почтовая программа (надо полагать, на компьютере посетителя она есть!) и открывается бланк письма, где в строке адреса уже стоит `e_polons@ukr.net`. Дальнейшее управление передается почтовой программе.

Но это еще не все. Можно составить письмо полностью, с темой и содержимым! Для этого параметру href присваивается длинное значение, составленное по следующим правилам.

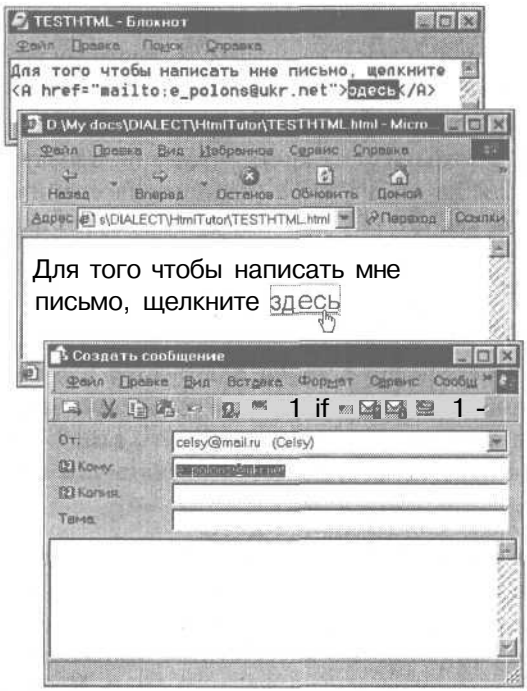


Рис. 9.2. С помощью дескриптора <A> можно создать ссылку, по которой будет открываться заготовка для электронного письма

1. После адреса вводится код `?subject=`, а после него — текст темы письма.
2. Если нужно написать что-то в теле письма, то этот текст вводится после текста темы. Разделителем служит код ;body=.
3. Все пробелы, точки и прочие знаки, которые броузер может посчитать служебными символами, в теме и теле письма заменяются их ASCII-кодом, перед которым стоит символ `%`.

Посмотрим, как это выглядит на примере (рис. 9.3). Диковато... Зато потом получается красиво.

Как-то раз меня спросили: можно ли с помощью такой ссылки отправить письмо сразу нескольким адресатам? Ох, уж эти любители спама... Впрочем, ладно, отвечу: можно. Формируя письмо для отправки, броузер не берет на себя функций почтовой программы, а только заполняет соответствующие поля письма. Что вы в них напишете, то он туда и поместит, не заботясь о дальнейшем. Что пишут в строке адреса, если письмо направляется в несколько адресов? Разумеется, список этих адресов через точку с запятой. Именно такой список нужно написать после `mailto:`, и письмо будет разослано.

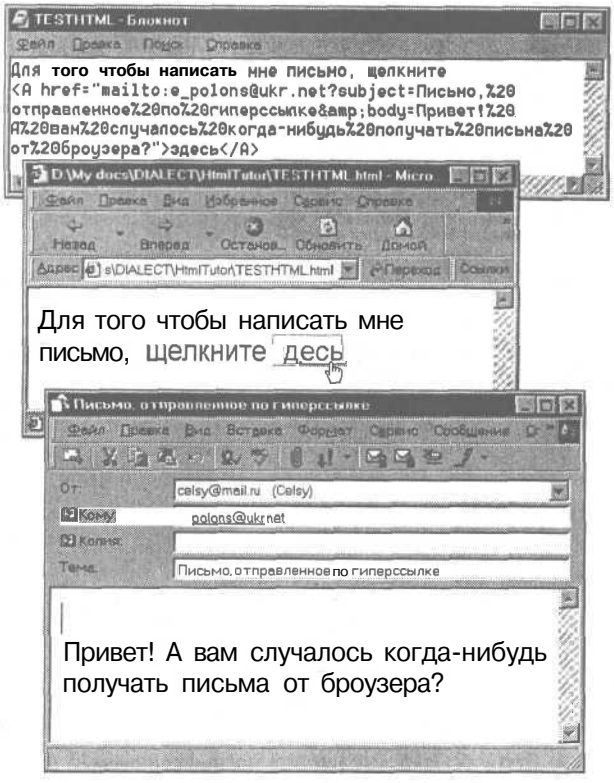


Рис. 9.3. Код для отправки писем стандартного содержания выглядит жутковато. Но результат того стоит

Запуск программ. Наконец, следует обратить пристальное внимание еще на один способ использования гипертекстовых ссылок — ссылки на выполняемые файлы.

Такие ссылки встречаются в Internet достаточно часто. Большинство выполняемых файлов, которые загружаются таким образом, — самораспаковывающиеся архивы. Но встречаются и другие программы.

Итак, напишем код, с помощью которого с Web-страницы, по идее, должна запускаться произвольная программа. В сущности, ничего сложного или незнакомого здесь нет — обычная ссылка на файл. Единственная особенность заключается в том, что при обращении по ссылке броузер предлагает на выбор два действия: запустить программу немедленно или сохранить ее на диске (рис. 9.4).

Такое предупреждение выводится не зря. Трудно придумать более "питательную" среду для компьютерных вирусов, чем Internet. Поэтому, если вы загружаете файл из Сети, лучше сохранить его на диске, проверить антивирусной программой и только потом запускать на выполнение.

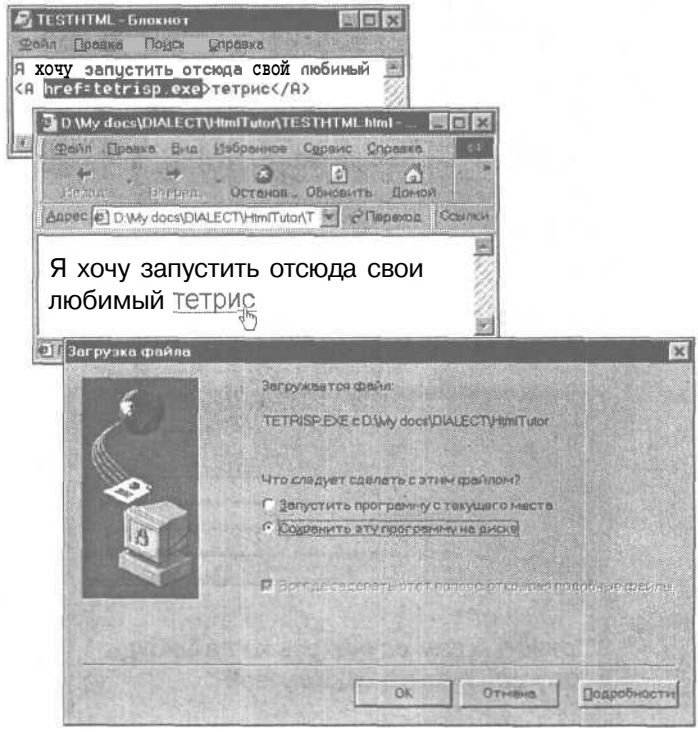


Рис. 9.4. Если ссылка указывает на выполняемый файл, броузер предлагает на выбор два действия: запустить программу немедленно или сохранить ее на диске. Как правило, лучше сохранить...

Ссылки-картинки

Едва ли не чаще текста в качестве гиперссылок используются изображения — всевозможные пиктограммы, логотипы, миниатюрные фотографии и даже целые графические "поля" ссылок, когда разные фрагменты одной картинки служат ссылками на различные объекты.

Используя в качестве гиперссылки картинку, следует помнить об одной особенности, уже упоминавшейся в главе 8: если у обычных изображений по умолчанию рамка отсутствует, то изображения, "по совместительству" служащие гиперссылками, по умолчанию заключаются в черную рамку толщиной 2 пикселя. Это далеко не всегда выглядит так эстетично, как, возможно, замышляли разработчики стандарта. Поэтому дескрипторы <IMG> таких изображений обычно снабжаются параметром border=0.

Последний вариант, — когда разные фрагменты одного изображения служат ссылками на различные объекты, — встречается сплошь и рядом. Действительно, это представляется естественным, особенно если мы хотим, чтобы у зрителя

складывалось целостное представление о странице. Такие объекты — одно изображение на несколько ссылок — описываются *картами изображений (image map)*.

Что такое карта изображения? Представим себе, что у нас есть картинка (рис. 9.5), разные объекты которой должны служить ссылками на разные Web-страницы. (На самом деле это изображение взято из мультимедийной версии Британники, но мы же можем помечтать, вообразив себя разработчиками электронной энциклопедии на базе HTML, верно?) Что нам нужно, чтобы сделать задуманное, не разрезая изображение на части?

Рис. 9.5. Каждый предмет на столе может служить ссылкой на свой раздел мультимедийной энциклопедии. Но как разбить это изображение на зоны гиперссылок, не разрезая его?

Очевидно, необходима некая конструкция, позволяющая разметить картинку, разделить ее на "зоны влияния" разных гиперссылок. Причем, в идеале у нас должна быть возможность создавать зоны разной формы: например, не только прямоугольные, но и круглые.

Именно для этого — для разделения изображения на зоны, каждой из которых может быть поставлена в соответствие гиперссылка, — и применяются карты изображений. Формируются они так.

Представим, что нам нужно разбить на зоны прямоугольное поле, размеры которого соответствуют размерам нашего изображения. Нам понадобится как-то отмечать координаты этих зон. Примем за точку отсчета горизонтальных и вертикальных координат верхний левый угол изображения.

Для создания карты изображения используется конструкция, состоящая из дескриптора `<MAP>` и "вложенных" в него дескрипторов `<AREA>`. Дескриптор `<MAP>` (от английского *map* — "карта") определяет собственно карту изображения целиком, а дескрипторы `<AREA>` (от английского *area* — "область", "зона") описывают отдельные области этой карты. Следовательно, дескриптор `<MAP>` является парным, а `<AREA>` — непарным. Вся же конструкция выглядит примерно так:

```
<MAP параметры карты>
      <AREA параметры области>
      <AREA параметры другой области>
      ...
</MAP>
```

Какие параметры есть у всей области? Размеры? Положение? Но все эти свойства определяется самой картинкой, так что описывать их заново нет смысла. Достаточно связать карту изображения с самим изображением. Именно за это "отвечает" единственный параметр дескриптора <MAP> — name. Благодаря ему карте присваивается уникальное имя, которое потом указывается в дескрипторе <IMG> с помощью специального параметра usemap. Таким образом карта изображения как бы "накладывается" на само изображение:

```
<MAP name="mymap">
      ...
</MAP>
...
<IMG src=bigimage.jpg usemap=#mymap border=0>
```

Имя изображения составляется по тем же правилам, что и другие имена в HTML. Но обратите внимание: при ссылке на него в значении параметра usemap перед именем карты ставится символ I.

Какие параметры у зон, на которые разбивается изображение? О, здесь наши возможности гораздо шире. Прежде всего, с помощью параметра shape мы можем описать форму области: прямоугольник, круг или произвольный многоугольник. Этим формам соответствуют значения rectangle (или сокращенно rect, используется по умолчанию), circle (сокращенно circ) и polygon (сокращенно poly). Маловато? Хочется разнообразия? Не стоит беспокоиться: даже если мы определим зоны действия гиперссылок приблизительно, посетитель страницы, используя такой неточный манипулятор, как мышь, этого не заметит.

Но знать одну лишь форму областей явно мало. Нужно еще описать их размеры и положение. Это делается с помощью параметра coords. Значением этого параметра является заключенный в кавычки список целых чисел. Эти цифры — координаты и размеры области, измеряемые в пикселях. То, как броузер их трактует, зависит от ее формы. Как вы думаете, сколько чисел нужно, чтобы однозначно описать круглую область? Правильно, три: координаты по горизонтали и вертикали (отсчитываются от верхнего левого угла изображения, помните?) и радиус. Именно в таком порядке они перечисляются в списке coords. Например, если нужно описать круг радиусом 40 пикселей, отстоящий от верхнего левого угла изображения на 10 пикселей по горизонтали и 20 пикселей по вертикали, нужно написать такой код:

```
<AREA shape=circ coords="10, 20, 40" другие параметры >
```

Вероятно, теперь вы догадываетесь, как описать прямоугольник: просто двумя парами чисел, определяющими координаты его границ. Перечисляются они по часовой стрелке: левая, верхняя, правая и нижняя. Например, прямоугольная область

размером 40x50, отстоящая от верхнего левого угла изображения на 10 пикселей по горизонтали и 20 пикселей по вертикали, описывается таким кодом:

<AREA shape=rect coords="10, 20, 50, 70" *другие параметры* >

Как задать координаты произвольного многоугольника? Вероятно, вы уже догадываетесь: нужно просто перечислить по очереди координаты всех его вершин. При этом нужно придерживаться только двух правил. Первое: горизонтальная координата задается раньше вертикальной. Второе: для того чтобы фигура получилась замкнутой, последняя пара координат должна дублировать первую. Например, прямоугольный треугольник с вершиной, отстоящей от верхнего левого угла изображения на 10 пикселей по горизонтали и 20 пикселей по вертикали и катетами, равными 40 и 50 пикселей, соответственно, описывается таким кодом:

<AREA shape=poly coords="10, 20, 60, 60, 10, 60, 10, 20" *другие параметры* >

И еще одно полезное значение параметра shape: default. Что оно означает, если, как мы уже знаем, по умолчанию все области считаются прямоугольными?

Область с параметром shape=default соответствует области, "покрывающей" все изображение. Это очень удобно: сначала задаем ссылку, которая должна работать в тех местах изображения, для которых не созданы ссылки.

Вроде, все. Мы знаем, как задать границы области. Не забыли ли мы чего-то важного?

Ну, конечно: если это область гиперссылки, то где же сама ссылка? Куда переходить после щелчка на этой области? За это в дескрипторе <AREA> "отвечает" тот же параметр, что и в дескрипторе <A> — href. Кроме того, для описания "неактивных" областей, не имеющих своих ссылок, используется параметр nohref. У этого параметра нет значений. Для того чтобы сообщить броузеру, что данная область не имеет гиперссылки, достаточно написать:

<AREA nohref *координаты области* >

Как и в других дескрипторах HTML, порядок перечисления параметров дескриптора <AREA> не имеет значения. Но порядок описания зон имеет значение: в случае, если эти зоны перекрываются, приоритет имеет та из них, которая была описана первой. В частности, это следует учитывать при описании неактивных зон (с параметром nohref): такие зоны описывают первыми.

Теперь, разобравшись в теории, давайте вернемся к нашей картинке из Британники. Разбить ее на зоны можно по-разному. Один из вариантов предлагается на рис. 9.6. Как видим, некоторые зоны перекрываются. И хорошо, что у нас есть правило перекрытия зон, гласящее, что гиперссылка будет работать для зоны, описанной первой. Иначе некоторые зоны пришлось бы описывать более сложными фигурами. В частности, вместо простого круга, внутри которого находится лупа, мы вынуждены были бы использовать многоугольник с большим количеством вершин.

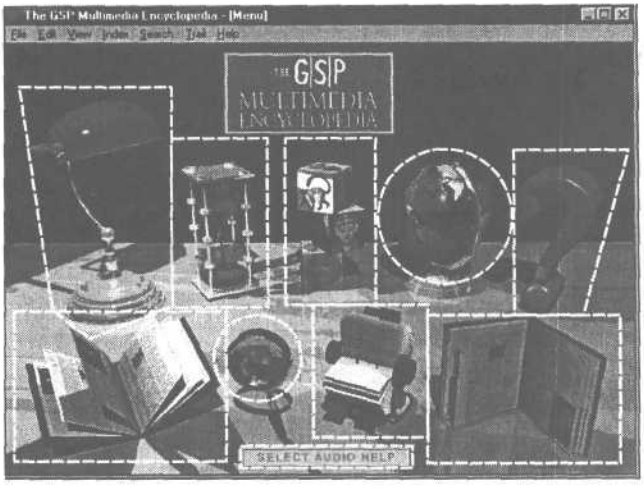


Рис. 9.6. Изображение, разбитое на зоны гиперссылок. Разумеется, в окне броузера эти линии не видны

Описывается такая карта изображения следующим кодом:

```
<MAP name=britan>
<AREA shape=poly coords="26, 125, 247, 125, 244, 445, 129, 493, 75, 450, 26, 125"
href="home.html">
<AREA shape=rect coords="248, 200, 386, 445" href="timeline.html">
<AREA shape=rect coords="411, 200, 543, 442" href="topics.html">
<AREA shape=circ coords="653, 321, 106" href="atlas.html">
<AREA shape=poly coords="745, 218, 911, 218, 845, 457, 745, 457, 745, 218"
href="quiz.html">
<AREA shape=rect coords="18, 452, 325, 672" href="browse.html">
<AREA shape=circ coords="370, 521, 63" href="search.html">
<AREA shape=rect coords="452, 442, 619, 637" href="index.html">
<AREA shape=rect coords="619, 458, 898, 674" href="display.html">
<AREA shape=rect coords="345, 650, 596, 680" href="audio.html">
</MAP>
```

Осталось "привязать" эту карту к изображению britan.jpg. Для этого поместим в соответствующее место HTML-кода следующий дескриптор:

```
<IMG src="britan.jpg" usemap=lbritan border=0>
```

Как рассчитать координаты и размеры областей? Для этого достаточно воспользоваться любым графическим редактором. В таких приложениях обычно имеется горизонтальная и вертикальная шкалы или хотя бы раздел строки состояния, где указываются координаты текущей точки.

А как быть тем посетителям страницы, которые скорости ради отключили отображение картинок в броузере? Смогут ли они пользоваться картами ссылок?

Конечно, не видя картинки, это сделать сложно. Но у нас есть возможность облегчить им жизнь. Для этого нужно воспользоваться тем же средством, что и при отображении обычных картинок — параметром alt. Но указывается он не в дескрипторе <IMG> (точнее, по крайней мере, не только в нем), а в дескрипторах <AREA> (рис. 9.7).

```
<MAP name=britan>
<AREA shape=poly coords="20,70,170,70,170,305,110,335,55,305,20,70" href="home.html"
alt="Заставка">
<AREA shape=rect coords="175,170,255,305" href="timeline.html" alt="Шкала времени">
<AREA shape=rect coords="290,170,365,305" href="topics.html" alt="Разделы">
<AREA shape=circ coords="445,217,80" href="atlas.html" alt="Географический атлас">
<AREA shape=poly coords="510,160,620,160,575,320,505,320,510,160" href="quiz.html"
alt="Викторина">
<AREA shape=rect coords="10,325,220,470" href="browse.html" alt="Обзор">
<AREA shape=circ coords="255,365,63" href="search.html" alt="Поиск">
<AREA shape=rect coords="315,305,425,435" href="index.html" alt="Алфавитный указатель">
<AREA shape=rect coords="425,320,625,470" href="display.html" alt="Показать">
<AREA shape=rect coords="240,450,410,470" href="audio.html" alt="Звуковое сопровождение">
</MAP>
<IMG src="britan.jpg" usemap=#britan alt="Заставка Британники" border=0>
```

Рис. 9.7. Отдельные области карты изображения можно снабжать комментариями

В дескрипторе <AREA>, как и в обычной гиперссылке, описанной дескриптором **<A>**, действует параметр target. Подробнее о нем читайте в главе 11.

Другой распространенный способ разбить изображение на несколько областей, каждая из которых ссылается на отдельный объект, заключается в физическом разделении этого изображения на несколько графических файлов, в каждом из которых содержится своя часть изображения: Затем эти части помещаются в таблицу (см. главу 10). Как легко догадаться, главный недостаток такого способа — то, что полученные области могут быть только прямоугольными. Однако изображение, разбитое на более мелкие файлы, быстрее загружается.

Виртуальная навигация

... с помощью якорей! Воистину, у программистов мозги набекрень. По каким еще морям, кроме виртуальных, плавают не на парусах, не на веслах, а на якорях-гиперссылках?

Так или иначе, но почему-то перемещение по Web-страницам назвали именно навигацией. И одна из главных задач Web-дизайнера — сделать так, чтобы эта навигация была как можно удобнее для посетителей его страниц.

Какие особенности дизайна больше всего раздражают посетителей вашего сайта? Не знаете? Ладно, меняем вопрос: что больше всего раздражает *вас* в чужих сайтах? Лично меня — необходимость пробираться через целый лес ссылок, чтобы докопаться до нужной информации. И каждый раз, когда загружается очередная страница, ждать, ждать, ждать...

Представьте себе такую ситуацию: вам нужен прайс-лист некоторой фирмы. Вы заходите на ее сайт и видите... великолепную заставку с логотипом. Щелкаете на этой заставке, попадаете на главную страницу, а там — пространная история компании. Вот черт! Ладно, в углу имеется ссылочка: "Наша продукция". Смотрим, что там... Большие фотографии и рекламные проспекты этой самой продукции. Но где же цены?! Ну наконец-то, вот оно: "Загрузить прайс-лист". Уф-ф... Щелкнем на последней кнопке и ждем загрузки.

Это только на первый взгляд кажется, что понятие "удобство" нельзя описать математической формулой. На самом деле оно — правда, очень приблизительно — обратно пропорционально количеству переходов по ссылкам, необходимых, чтобы добраться до нужной информации. Поэтому дизайнеры, планируя структуру сайта, стараются следовать таким правилам.

* Чем выше спрос на информацию, тем ближе к главной странице сайта она должна находиться. Как определить спрос на информацию? Ну, хотя бы по посещаемости страницы, на которой она расположена.
* Ссылки-стрелки с надписями наподобие "вернуться в начало" и "перейти в конец" на длинных, требующих многократной прокрутки страницах обычно не вызывают у посетителей ничего, кроме глубокой

признательности. То же касается ссылок на главную страницу, а также не следующий и предыдущий связанные документы.

- Очень часто на Web-страницах создают специальный блок — *панель ссылок*. Обычно она располагается сбоку, сверху или снизу от основного содержимого страницы, так, чтобы до нее было удобно добраться и чтобы она не мешала читать основной текст. Очень многие размещенные в Сети книги и крупные статьи построены по такому принципу: слева помещается список глав, а справа — та глава, которую открыл посетитель.
- Почти на любом крупном сайте есть специальная страница, посвященная исключительно ссылкам — *карта сайта*. На ней обычно в текстовой, а иногда в красивой графической форме отражена структура сайта: какие страницы на нем есть, что там находится и, разумеется, ссылки на эти страницы. Карта сайта — это та точка, куда с последней надеждой приходят посетители, заблудившиеся в Паутине. Не обманите их ожидания!

Несмотря на сходство названий, следует помнить о том, что карта сайта и карта изображения — совершенно разные вещи. Первая является Web-страницей, играющей роль справочника-путеводителя, а вторая — конструкцией HTML. Впрочем, очень часто можно встретить карту сайта, реализованную в виде карты изображения.

Резюме

Гиперссылки — мощнейшее средство языка HTML, благодаря которому возможны переходы между различными документами и объектами, размещенными в Internet. Реализуются гиперссылки с помощью дескриптора <A>. Это парный дескриптор, внутри которого заключен объект, служащий гиперссылкой, — текст, графика или то и другое вместе. Основным параметром дескриптора **<A>** является href, определяющий ссылку на объект, — другую HTML-страницу, электронное письмо или файл. Если файл, на который указывает ссылка, может быть открыт в броузере или в поддерживаемом броузером приложении, он открывается. В противном случае выполняется загрузка этого файла с сохранением его на диске.

Отдельного внимания заслуживает применение гиперссылок для отправки писем по электронной почте. Дескриптор **<A>** позволяет составить шаблон электронного письма, которое затем посетитель страницы может отправить по адресу, указанному в параметре `href`.

Для того чтобы сослаться не просто на страницу, а на ее определенное место, используются закладки. Закладки — это дескрипторы <A> с параметром name, которому присвоено уникальное имя. Для того чтобы сослаться на такую закладку, в значении параметра `href`, кроме URL файла, указывают имя закладки. Разделителем между URL и именем закладки служит символ `#`.

Кроме описанных свойств, дескриптор **<A>** имеет еще одно: с помощью параметра target он позволяет выбрать окно, в котором будет открыт новый объект. Подробнее этот параметр будет описан в главе И, а пока что мы можем пользо-

ваться двумя его предопределенными значениями — _self и _blank. Первое из них предполагает открытие объекта, на который указывает ссылка, в том же окне, что и документ, где эта ссылка находится, а второе — в новом окне.

Гиперссылкой может служить не только целое изображение, но и его фрагменты. Для этого создаются так называемые карты изображений. Карта изображения — это конструкция языка HTML, образованная с помощью дескрипторов <MAP> и <AREA>. Дескриптор <MAP> определяет имя карты, по которому она впоследствии связывается с самим изображением. В параметре usemap дескриптора <IMG> указывается имя, присвоенное карте с помощью параметра name дескриптора <MAP>, предваряемое символом #.

Внутри конструкции <MAP>... </MAP> помещаются дескрипторы <AREA>. Каждый такой дескриптор описывает параметры определенной области внутри изображения, которой поставлена в соответствие та или иная гиперссылка. Этими параметрами являются форма, координаты, адрес объекта, на который указывает ссылка. Кроме того, здесь, как в дескрипторе <A>, действует параметр target и, как в дескрипторе <IMG>, — параметр alt (см. главу 8).

Форма области определяется параметром shape, который принимает одно из четырех значений: rect (по умолчанию), circ, poly и default. Эти значения соответствуют прямоугольнику, кругу, многоугольнику и всей области изображения. Для каждой из первых трех форм предусмотрена своя система задания координат. Координаты области определяются параметром coords. Значением этого параметра служит текстовая строка, где через запятую перечислены значения координат (в пикселях), однозначно определяющие размер и положение данной области. Для круга это координаты центра и радиус, для прямоугольника — координаты левой, верхней, правой и нижней сторон, а для многоугольника — координаты вершин.

Система гиперссылок, связывающих отдельные страницы Web-сайта, должна быть тщательно продумана. От удобства этой системы зависит то, насколько комфортно посетителям будет пользоваться сайтом и, в конечном счете, то, насколько этот сайт будет полезен и им, и вам.

Тесты

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?
   а) <A> Гиперссылка </A>
   б) <HREF> page2.htm </HREF>
   в) <A href=page2.htm> Гиперссылка </A>
   г) <A href="page2.htm"> Гиперссылка </A>
   д) <A target=page2.htm> Гиперссылка </A>
2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?
   а) <A href=page2.htm target=blank> Гиперссылка </A>
   б) <A href=page2.htm target=_blank> Гиперссылка </A>
   в) <A href=page2.htm target=blank_> Гиперссылка </A>

3. Какой или какие из следующих фрагментов HTML-кода, содержащие ссылку на метку label, которая находится на странице page2.htm, содержат ошибки?

   а) <A href=page2.htm name=label> Гиперссылка </A>

   б) <A href="page2.htm\label"> Гиперссыпка </A>

   в) <A href="page2.htm/label"> Гиперссыпка </A>

   г) <A href="page2.htm#label"> Гиперссылка </A>

   д) <A href="page2.htm, label"> Гиперссылка </A>

4. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а) <A href="mailto:tome@mail.dom"> Написать мне письмо </A>

   б) <A href="mailto:tome@mail.dom; subject=Привет;body=Как дела?"> Написать мне письмо </A>

   в) <A href="mailto:tome@mail.dom? subject=Привет;body=Как дела?"> Написать мне письмо </A>

   г) <A href="mailto:tome@mail.dom? subject=Привет;body=Как%20дела?"> Написать мне письмо </A>

   д) <A href="mailto:tome@mail.dom?subject=Привет"> Как дела? </A>

   е) <A href="mailto:tome@mail.dom?subject=Привет;"> Как дела? </A>

5. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а)
```
<MAP name=image>
<AREA shape=triangle coords="10,20,30,40" href=pag2.htm>
</MAP>
```

   б)
```
<MAP name=image>
<AREA shape=rect coords="10,20,30,40" href=pag2.htm>
</MAP>
```

   в)
```
<MAP name=image>
<AREA shape=poly coords="10,20,30,40" href=pag2.htm>
</MAP>
```

6. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а)
```
<MAP name=image>
...
</MAP>
...
<IMG usemap=image>
```

   б)
```
<MAP name=image>
...
</MAP>
...
<IMG usemap=#image>
```

в)
```
<MAP name=image>
...
</MAP>
...
<IMG usemap=#image src=image.jpg>
```

г)
```
<MAP name=image>
...
</MAP>
...
<IMG src=image.jpg#image>
```

д)
```
<MAP name=image>
...
</MAP>
...
<A href=image> <IMG src=image.jpg> </A>
```

Глава 10

Табличный дизайн

В этой главе...

- Для чего нужны таблицы
- 4 Из чего состоят таблицы
- Пример табличного дизайна
- Горизонтальное выравнивание
- Вертикальное выравнивание
- Размеры таблицы
- Размеры ячеек
- Внутренние отступы
- Рамки
- Частичное отображение рамок
- Фон таблицы и ячеек
- Слияние ячеек
- Заголовок таблицы
- Заголовки строк и столбцов
- Группировка ячеек

Для чего нужны таблицы

Один из ответов на этот вопрос очевиден: для представления информации в табличном виде. Но — только один.

Если вам кажется, что большинство Web-страниц обходится без таблиц, проведите несложный эксперимент. Скачайте штук пять наиболее симпатичных вам страниц и откройте их, например в MS Word. Примерно в трех случаях из пяти у страницы обнаружится невидимый "скелет" - - сложная сетка из крупных и мелких ячеек (рис. 10.1). Уберите его — и содержимое страницы собьется в бесформенную кучу.

Если вы заглянете в HTML-код этих страниц, то, очень может быть, придете в состояние священного трепета перед сверхчеловеками, которые могут такое написать и, главное, потом не только они могут разобраться в написанном, но *это еще и работает*!

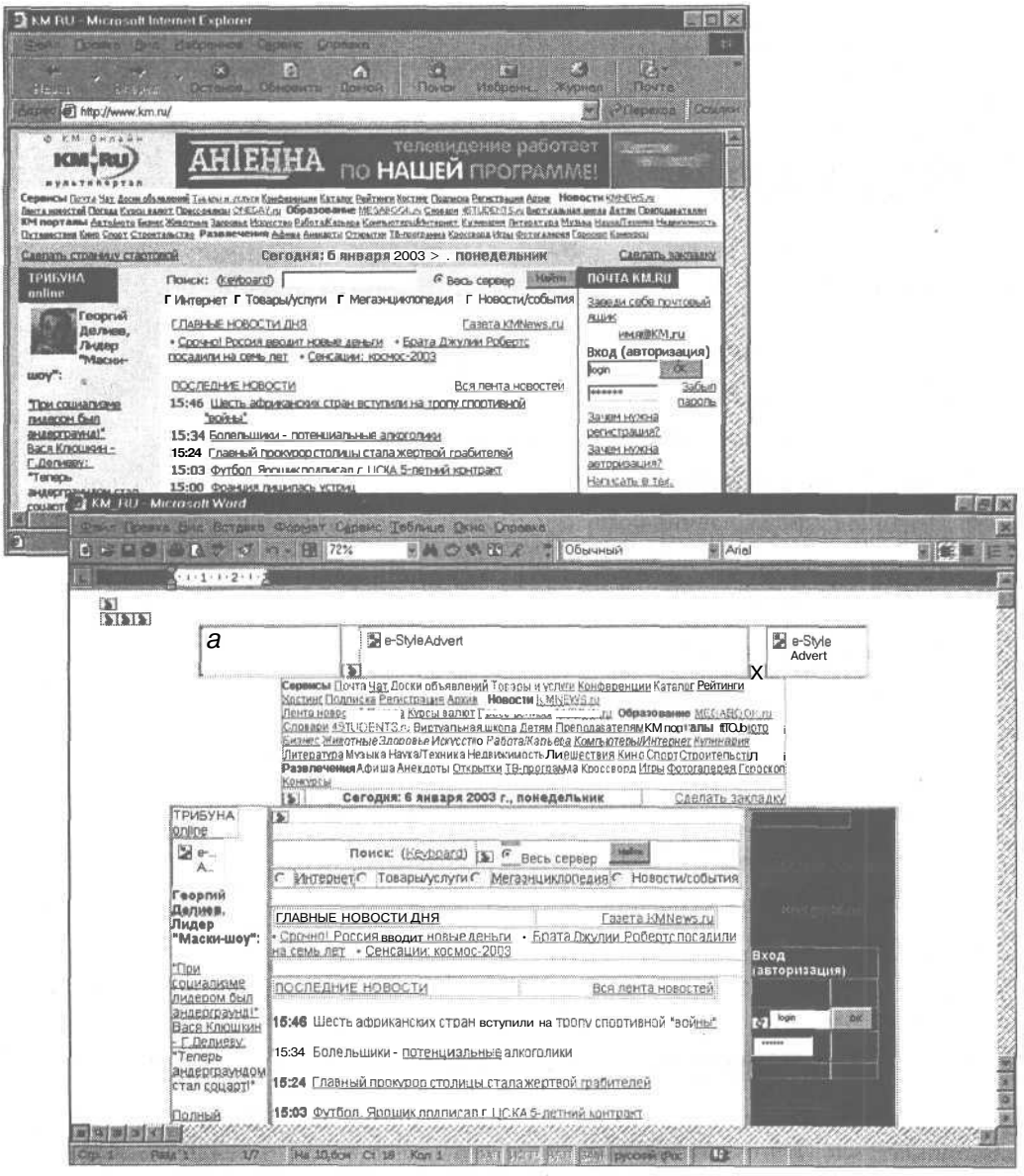


Рис. 10.1. У многих Web-страниц есть невидимый "скелет" — табличная структура

Если вас бросило в жар — выпейте стакан чистой холодной воды, если в холод — крепкого чая или кофе. Но настоящее лекарство от священного трепета делается из здравого смысла, логики и практического опыта. Первые два ингредиента у вас наверняка есть, а последний, как известно, приходит со временем. Начиная прямо с этого момента.

Из чего состоят таблицы

Ответ на этот вопрос смехотворно прост: из ячеек. Ну да, из прямоугольных ячеек, в которые что-нибудь вписано или врисовано. Тем, кто имел дело с электронными таблицами или таблицами в электронных текстовых документах, такие ячейки отлично знакомы. Поскольку ячейки прямоугольные, то они образуют горизонтальные ряды, именуемые строками, и вертикальные ряды, именуемые столбцами. Частенько красоты ради две, три и больше ячеек "сливают" в одну, а внутрь ячеек, кроме обычного текста и фафики, вставляют новые таблицы. Вот после этого и начинается тот кошмар, который вы видели на Web-страницах, открытых в Word.

Следующий вопрос немного посложнее. Нарисовать плоскую таблицу на плоской "странице" Word — пара пустяков. Нужно только задать количество строк и столбцов и подрегулировать их ширину. Но как быть с HTML-кодом, где объекты, выводимые на экран броузера, описываются последовательно? Не можем же мы, в самом деле, рисовать в Notepad двумерную таблицу! Как же ее закодировать?

Вот именно так — *последовательно*. По очереди, ячейку за ячейкой, строку за строкой, как если бы мы "разобрали" таблицу и вытянули ее в одну длинную цепочку. На рис. 10.2 приведен код простейшей таблицы из двух строк и двух столбцов.

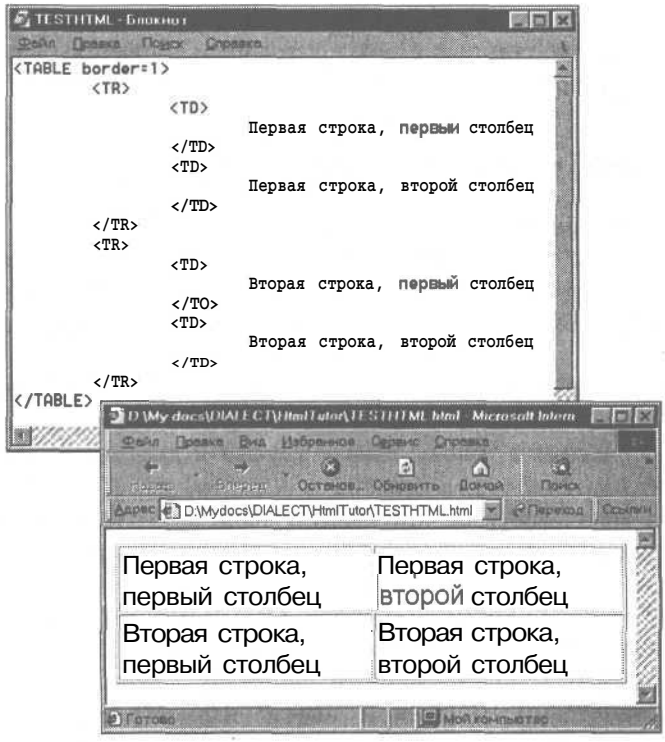


Рис. 10.2. Код простейшей таблицы, состоящей из двух строк и двух столбцов

Как видим, HTML-структура, описывающая таблицу, состоит из трех основных элементов.

* Дескриптора <TABLE>, внутри которого заключена вся таблица. Этот дескриптор описывает параметры таблицы в целом.
* Дескрипторов <TR> (от английского *Table Row* — строка таблицы), внутри которых заключены строки. Каждый такой дескриптор описывает параметры соответствующей строки.
* Дескрипторов <TD> (от английского *Table Data* — данные, или содержимое таблицы), внутри которых заключено содержимое ячеек. Каждый такой дескриптор описывает параметры отдельной ячейки.

Как вы уже, вероятно, заметили, текст в ячейках разбивается на строки в зависимости от ширины окна броузера. Это не всегда желательно, так как может сильно изменить вид всей страницы. В таких случаях используют параметр nowrap, запрещающий автоматический переход на новую строку в данной ячейке:

```
<TD nowrap> Не переходить на новую строку! </TD>
```

Однако последствия применения этого параметра не всегда приятны: для просмотра длинных строк посетителям придется пользоваться горизонтальной прокруткой, а это очень нежелательно.

Пример табличного дизайна

Рассмотрим подробнее эти дескрипторы на примере.

Предположим, нам нужно решить простейшую дизайнерскую задачку: построить начальную Web-страницу сайта, где сверху написано название фирмы, справа на большом поле — последние новости, а слева — узкий столбик ссылок на другие страницы сайта.

Как с помощью таблицы разбить поле Web-страницы на две колонки, в правой из которой находится список новостей, в левой — список ссылок?

Первое, что приходит в голову: сверху крупно и красиво пишем заголовок, а дальше идет таблица *2xN* — два столбца и *N* строк, где *N* — количество ссылок или новостей, в зависимости от того, чего больше. Для этого нам вполне подойдет код, представленный на рис. 10.2, в котором изменено только количество строк и содержимое ячеек (рис. 10.3).

Да, примерно так... Но только без рамок. Откуда они взялись?

Из "образцовой" таблицы, которую мы скопировали, не особо задумываясь над содержанием ее отдельных частей. А напрасно: внешний вид таблицы определяется ее параметрами. В данном случае параметр border в дескрипторе <TABLE> — явно лишний. Если его удалить, исчезнет и рамка (рис. 10.4).

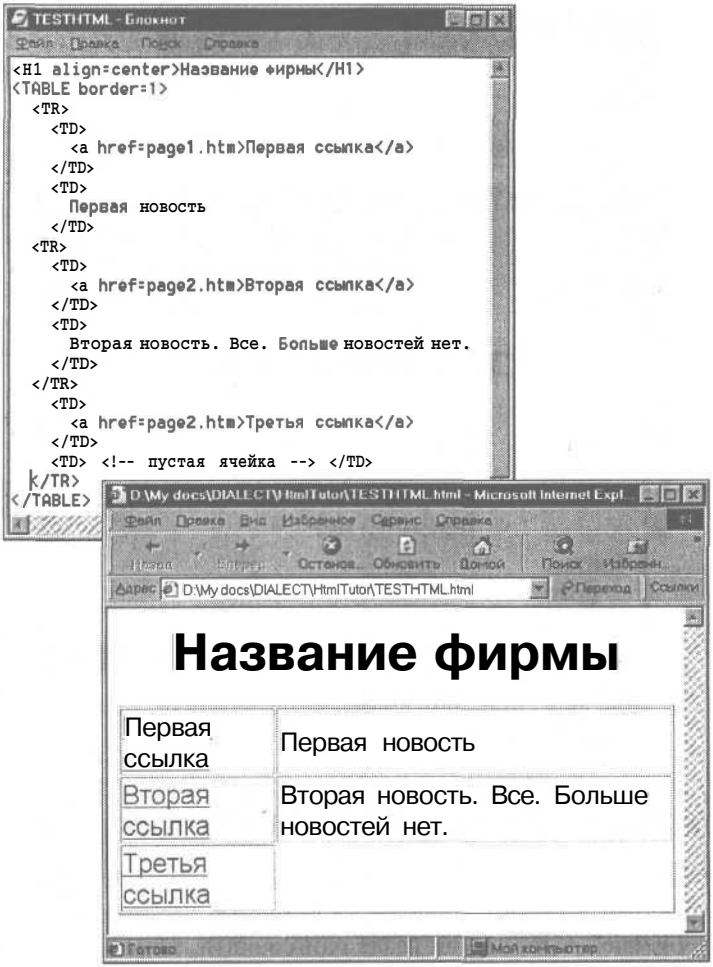


Рис. 10.3. Простейшая табличная верстка Web-страницы: все бы хорошо, но рамок не должно быть видно

Кажется, самое время рассмотреть параметры таблиц, благодаря которым стало возможным такое многообразие дизайнерских Internet-решений.

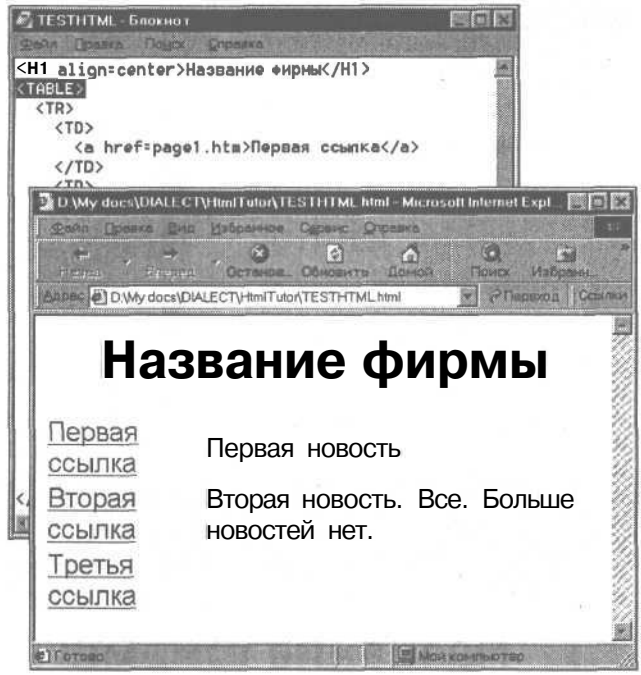


Рис. 10.4. Для того чтобы не было рамки, удалите параметр border=l

Горизонтальное выравнивание

Ну как же без него! Что бы мы ни делали со страницей, атрибут align неотвязно нас преследует. Впрочем, лично я на него не в претензии: очень помогает в самых разных случаях. А то, что он везде одинаковый, помогает еще больше.

Если вы подзабыли, где еще используется параметр align, просмотрите главы 2 и 8.

В дескрипторе <TABLE> атрибут align "отвечает" за выравнивание таблицы относительно краев окна броузера (или ячейки другой, большей таблицы, внутрь которой она вставлена). Нетрудно догадаться, что происходит, когда параметр align принимает значения left, right и center: если только таблица не занимает всей ширины отведенного ей пространства, то она соответственно прижимается к левому или правому краю либо выравнивается по центру.

В Куда при этом девается свободное место справа и слева таблицы?

При выравнивании таблицы по правому краю оставшееся слева место можно заполнить текстом или рисунком. Обратите внимание, что в HTML-коде "заполнитель" располагается *после* таблицы:

```
<TABLE align=right>

</TABLE>
<I>Мы всегда рады видеть Вас у нас!</I>
```

К сожалению, такой прием ненадежен: стоит пользователю изменить размер окна броузера, и весь дизайн "съезжает", как талый снег с крыши, вплоть до полного безобразия (рис. 10.5).

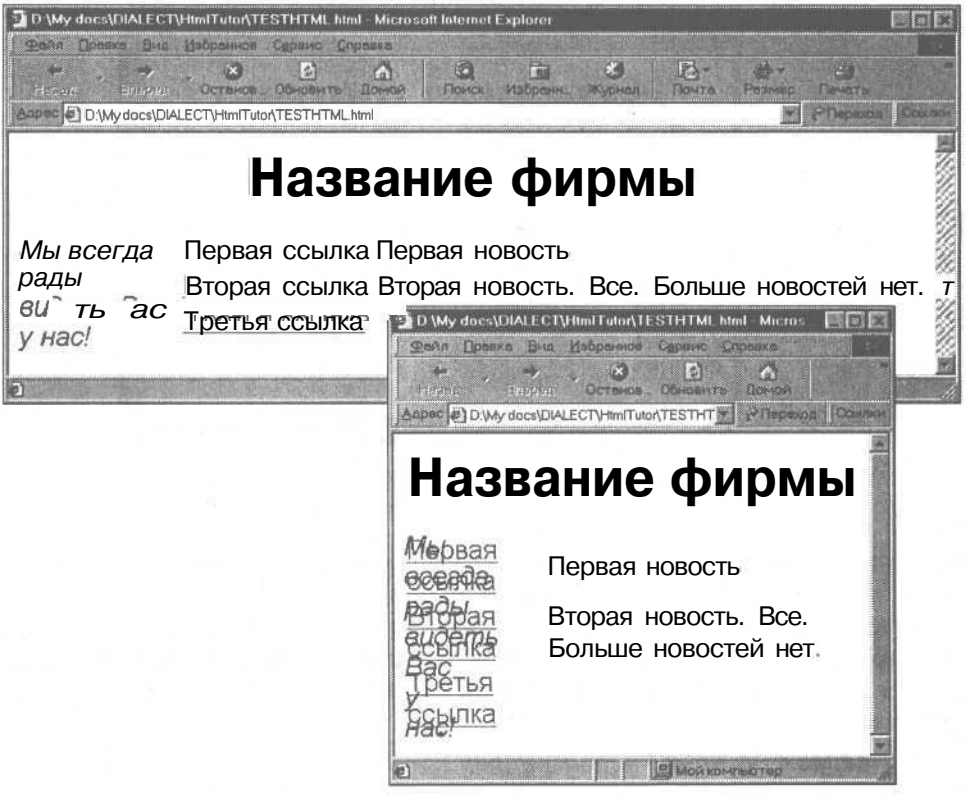


Рис. 10.5. Сбоку от таблицы можно поместить текст. Но осторожно: стоит изменить размер окна, и весь дизайн "поплывет"

А как задать выравнивание содержимого ячеек? Параметр align дескриптора <TABLE>, мы уже знаем, здесь не поможет. Но ведь этот параметр универсальный: кроме <TABLE>, он используется в дескрипторах <P>, <IMG>... Почему бы не воспользоваться им в качестве параметра ячейки? И, действительно, такой подход срабатывает: параметр align внутри дескриптора <TD> определяет выравнивание текста относительно ячейки (рис. 10.6).

Таким образом, мы можем использовать уже знакомые нам значения параметра align — left, right и justify, — для того чтобы выравнивать содержимое ячейки таблицы по левому или правому краю либо по центру.

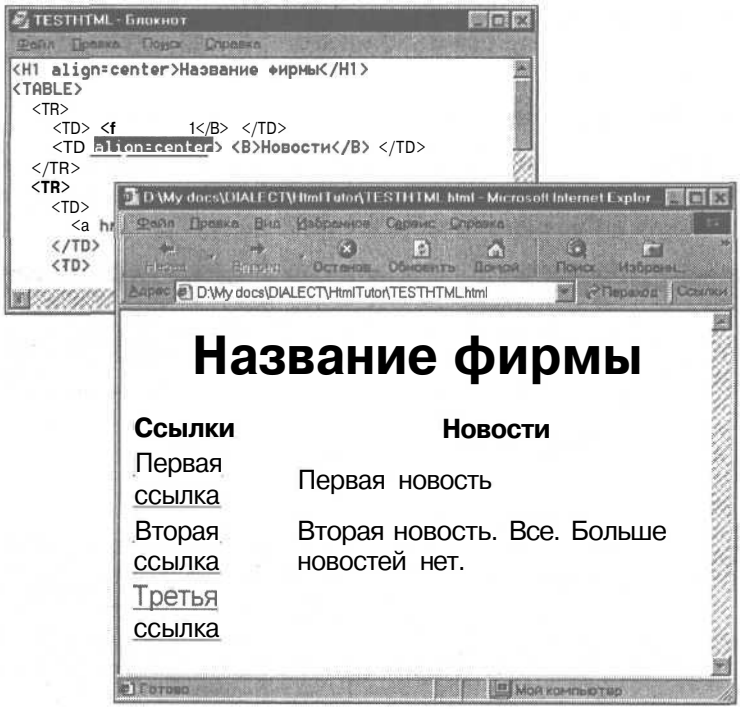


Рис. 10.6. Параметр align внутри дескриптора определяет выравнивание текста относительно ячейки

Но ячеек много. Как быть, если в выравнивании нуждается целая строка или столбец? Конечно, можно, поплевав на ладони, аккуратно вставить соответствующие параметры в каждую ячейку. Но до чего же это скучно!

Что касается строк, то здесь имеется простое и логичное решение: вместо описания способа выравнивания в каждой отдельной ячейке задать его для всей строки в дескрипторе <TR>:

```
<TR align=center> ... </TR>
```

Что же касается столбцов, то здесь задача решается несколько сложнее: сначала нужно сгруппировать ячейки, а потом описать общие для них параметры (см. раздел "Группировка ячеек").

Что произойдет, если для всей таблицы задан один тип выравнивания, а для ячейки — другой? Здесь действует общее правило для одноименных параметров: если значение параметра, определенного для всей таблицы в дескрипторе <TABLE>, не совпадает со значением такого же параметра в дескрипторе <TR>, то для данной строки используется параметр, определенный в дескрипторе <TR>. Аналогичным образом, если значение параметра для некоторой ячейки отличается от значения такого же параметра для содержащей ее строки или для всей таблицы, то приоритет имеет значение, заданное в дескрипторе ячейки. Например,

в коде на рис. 10.7 содержимое всей таблицы выравнивается по левому краю, в первой строке — по центру и в последней ячейке первой строки — по правому краю.

Рис. 10.7. При определении одноименных параметров приоритет имеют параметры ячейки, затем — и строки всей таблицы

Вертикальное выравнивание

А нельзя ли немножко укоротить код в рассмотренном выше примере, но так, чтобы вид страницы при этом не изменился? Ведь, в сущности, нам нужна таблица, состоящая всего из одной строки с двумя ячейками: в левой ячейке — колонка ссылок, в правой — колонка новостей (рис. 10.8).

И все бы неплохо, но между колонкой новостей и заголовком появился странный зазор. Откуда он взялся?

Разумеется, это не "глюки" броузера или недосмотр разработчиков. Просто по умолчанию содержимое табличной ячейки выравнивается вертикально по центру так, чтобы расстояние от текста до верхней и нижней границ было одинаковым. Пока ячейки были маленькие, это не было заметно. Но стоило появиться крупной ячейке с небольшим количеством текста внутри — и все изменилось.

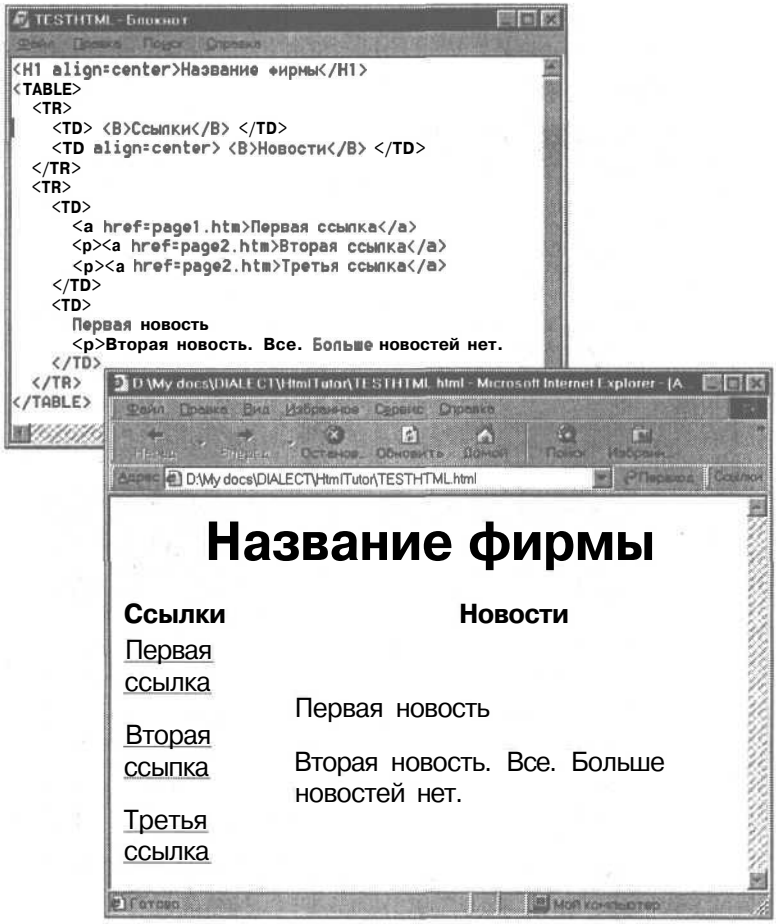


Рис. 10.8. Количество ячеек можно сократить, но тогда колонка новостей странно смещается вниз

Как поправить дело? Нужно изменить режим вертикального выравнивания, определив явно значение параметра valign. В частности, для выравнивания содержимого ячейки по верхнему краю нужно присвоить ему значение top (рис. 10.9).

Вертикальное выравнивание... Несмотря на то, что это вещь вполне логичная и естественная, мы с ним сталкиваемся впервые. До сих пор речь шла только о выравнивании объекта — будь то текст или рисунок — только по горизонтали. И это понятно: необходимость вертикального выравнивания относительно окна броузера возникает сравнительно редко. Но относительно ячейки таблицы это приходится делать сплошь и рядом.

Какие еще есть варианты вертикального выравнивания? Конечно, по нижнему краю и по середине ячейки. Им соответствуют значения bottom и middle параметра valign. Впрочем, последнее значение не обязательно указывать явно: оно используется по умолчанию. Кроме того, есть еще один способ вертикального вы-

равнивания, которому соответствует параметр baseline. Это выравнивание первых строк текста по базовой линии шрифта — воображаемой линии, на которой "стоят" буквы, так что вниз "свешиваются" только подстрочные элементы.

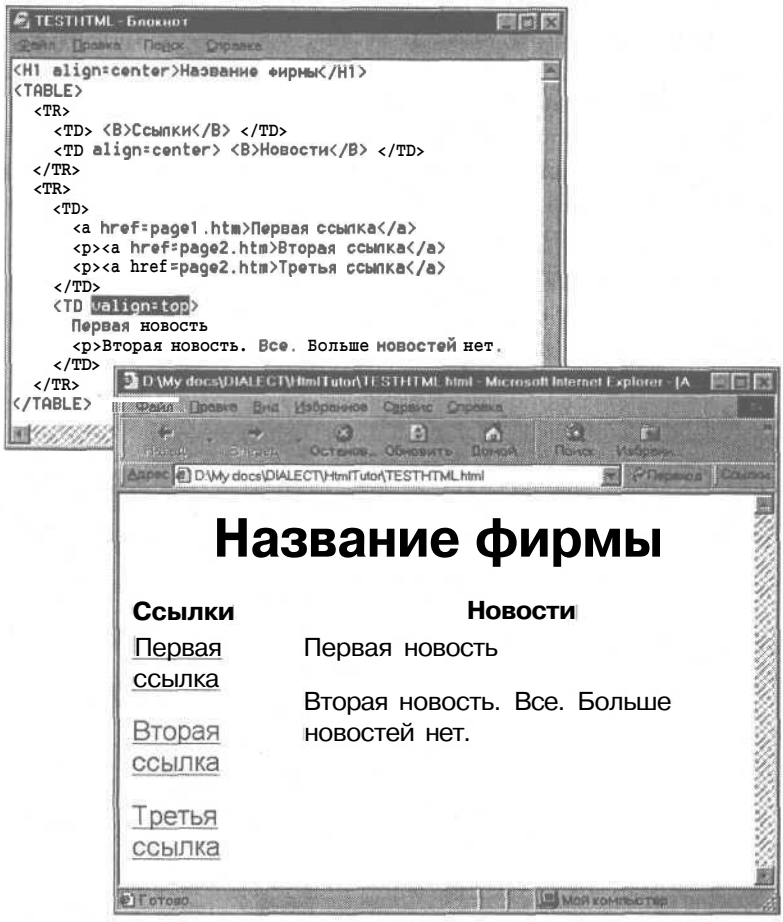


Рис. 10.9. Для выравнивания содержимого ячейки по верхнему краю нужно присвоить параметру valign значение top

Часто тот или иной способ вертикального выравнивания применяется для всех ячеек в строке таблицы. Для того чтобы упростить себе работу и сократить код, можно определить параметр valign для всей строки, поместив его в дескриптор <TR>:

```
<TR  valign=top>
      <TD>
      ...
      </TD>
</TR>
```

Размеры таблицы

Для того чтобы вид страницы не так сильно зависел от размеров окна броузера, в дескрипторе <TABLE> задают минимальную ширину и высоту таблицы. Для этого используются параметры width и height, соответственно. У вас возникло ощущение "дежа вю"? Отлично! Значит, вы помните, что точно такие параметры и в аналогичных целях использовались... где?

Правильно, в дескрипторе <IMG> (см. главу 8). Только в таблицах возможности применения этих параметров гораздо богаче. Как и в случае с изображениями, габариты задаются как в абсолютных значениях (в пикселях), так и в относительных (в процентах от ширины окна).

Способ верстки Web-страниц, при котором размеры таблиц задаются в процентах, иногда называют "резиновым" дизайном за то, что при изменении размеров окна такие страницы как бы растягиваются или сжимаются, сохраняя основные пропорции.

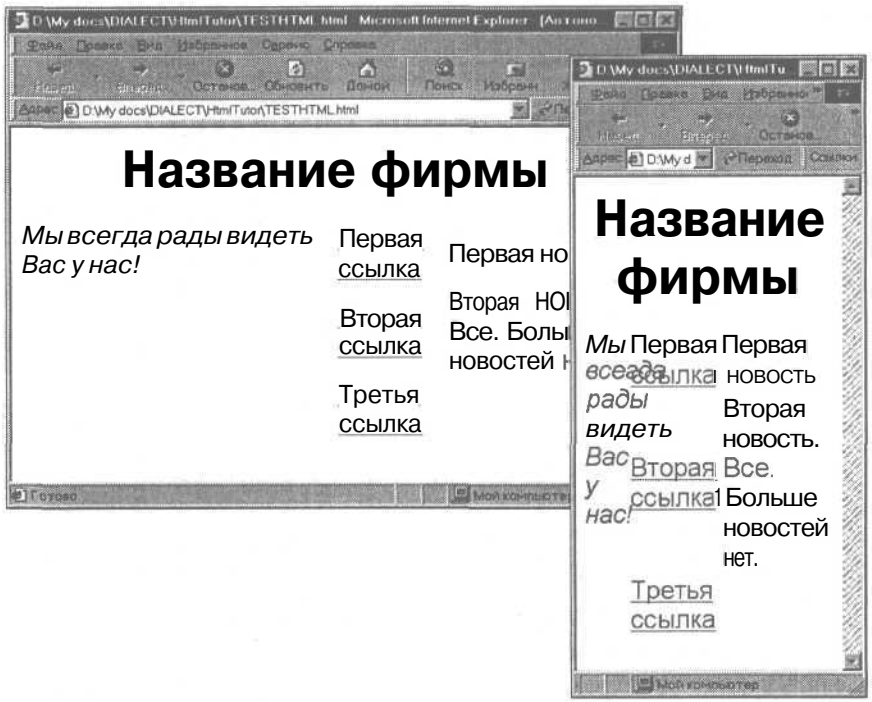


Рис. 10.10. Если офаничить таблицу половиной окна, для текста слева остается больше места. Но такое решение по-прежнему далеко от радикального

Например, чтобы избежать неприятной ситуации, показанной на рис. 10.5, можно ограничить ширину таблицы, скажем половиной окна, заменив первую строчку кода на такую:

```
<TABLE align=right width=50%>
```

Впрочем, эта мера действенна только до определенной степени. Можно сузить окно так, чтобы в левой части осталось по одному слову в каждой строке, и слова все равно будут "налезать" на таблицу (рис. 10.10). Что тогда делают Web-дизайнеры? Тяжело вздыхают и... строят новую таблицу, побольше, чтобы вписать строптивый кусок текста в ее левый столбец.

Размеры ячеек

Как и многие другие параметры HTML, width и height являются универсальными: определяют аналогичные свойства похожих объектов. Нетрудно догадаться, что размеры табличных ячеек задаются тоже с их помощью. Причем эти размеры, так же как и другие, могут быть абсолютными (в пикселях) или относительными (в процентах от ширины таблицы).

Обычно, если это важно, размеры ячеек указывают в процентах. Тогда при изменении размеров окна броузера таблица будет пропорционально "растягиваться" или "сжиматься". Если же задать размеры в пикселях, то в слишком большом окне рядом с такой таблицей останется много пустого места, что выглядит неаккуратно, а в слишком маленьком окне таблица не поместится, что приведет к появлению горизонтальной прокрутки. А, как мы уже знаем, текст, не помещающийся в окне по горизонтали, очень неудобно читать.

Например, что нужно сделать, чтобы в нашем примере правый столбец был втрое шире левого? Правильно, присвоить ячейкам правого столбца значение параметра width, равное 75%. Или ячейкам левого столбца — 25% (рис. 10.11). Теперь, как бы мы ни растягивали окно броузера, левый столбец всегда будет занимать только четверть его ширины.

Однако в некоторых случаях приходится определять размер ячейки именно в пикселях. В частности, такой подход широко используется, когда таблица представляет собой рисунок, разбитый на части так, чтобы каждая часть помещалась в отдельной ячейке. Тогда, для того чтобы размеры ячейки точно соответствовали размерам помещенного внутри изображения, их задают в пикселях.

Можно ли задать, например, высоту всей строки в дескрипторе <TR>, "сэкономив" на этом немного сил и сократив код? Нет, нельзя. Ширина и высота ячеек определяются только в дескрипторах, описывающих сами ячейки либо их группы (см. раздел "Группировка ячеек").

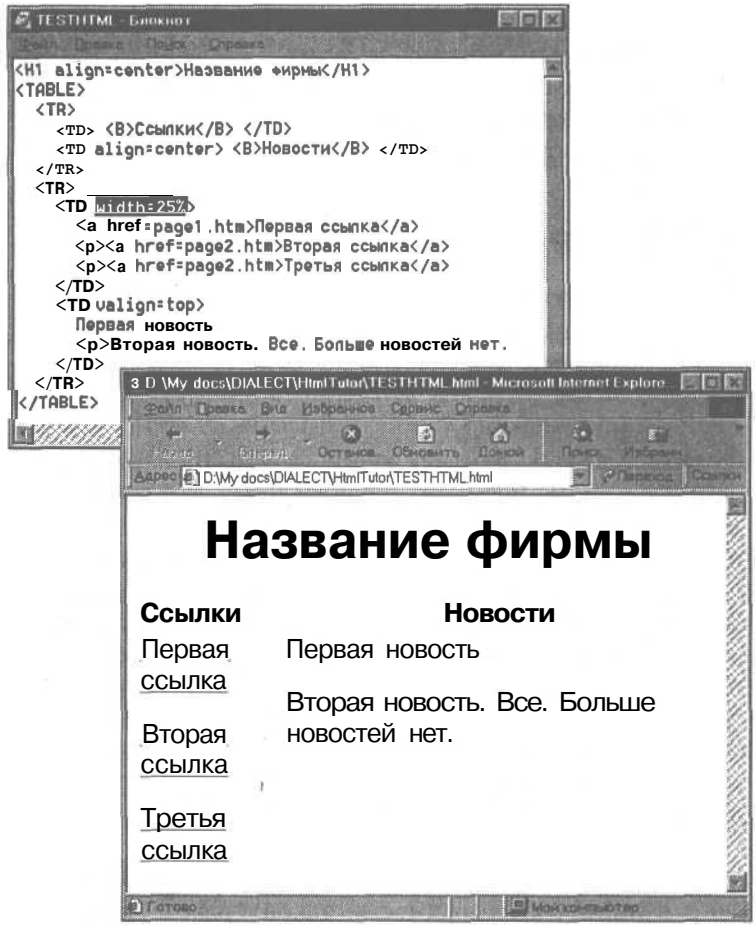


Рис. 10.11. Теперь, когда параметру width присвоено значение 25%, ячейка ссылок всегда будет занимать по ширине четверть таблицы, независимо от ширины окна броузера

Внутренние отступы

А это что такое? Первый вид отступов вам, вероятно, знаком, так как используется в таблицах, которые встречаются в электронных текстовых документах. Это расстояние между границей ячейки и границей текста. Оно измеряется в пикселях и определяется параметром cellpadding. Не начинать же текст прямо от рамки! Это неаккуратно. Здесь нужен некоторый отступ. Только в том случае, если рамка невидима, можно себе позволить сделать это расстояние нулевым. И то не всегда.

Второй вид отступов вам, скорее всего, встретится впервые. Для того чтобы понять, что это такое, нужно внимательно присмотреться к Web-таблицам. Обратите внимание: на самом деле, в отличие от других, знакомых вам, таблиц, их рамки *двойные* — каждая ячейка как бы заключена в собственное "окошко". Ме-

жду соседними "окошками" обычно имеется некий зазор. Его величина регулируется параметром cellspacing — опять же, в пикселях. Например, если присвоить таблице параметры cellspacing=40 и cellpadding=25, то отступ между ячейками составит 40, а между границей ячеек и текстом — 25 пикселей (рис. 10.12).

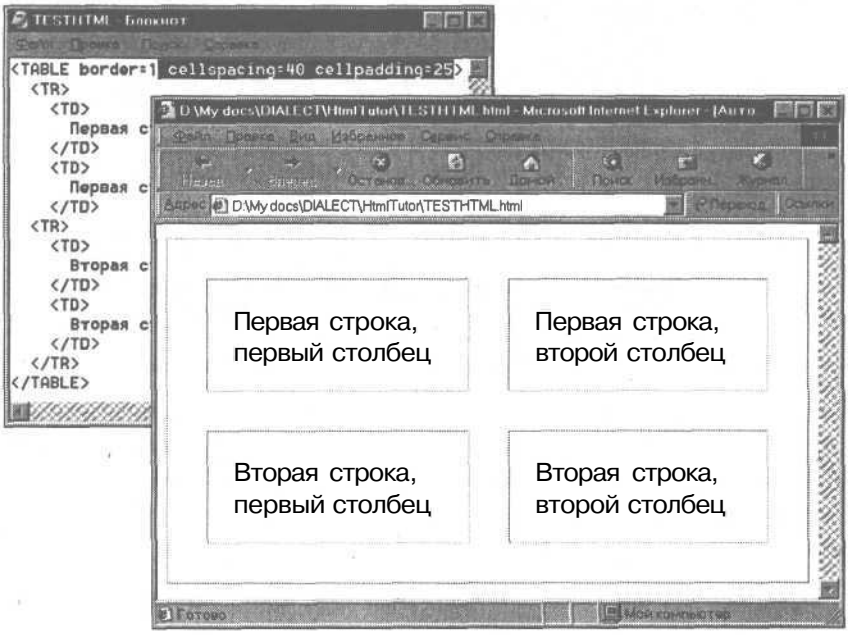


Рис. 10.12. Отступы между границами ячеек регулируются параметром cellspacing, а от границы ячейки до ее содержимого — параметром cellpadding

Рамки

Какие могут быть свойства у рамки таблицы? Здесь все очень просто: толщина и цвет. С параметром, "ответственным" за первое свойство, мы уже знакомы: это параметр border, и он определяет толщину рамки в пикселях. Все ясно? Вроде бы... Но давайте проведем эксперимент: что получится, если присвоить ему значение, скажем 15? По идее, не должно получиться ничего особо интересного: что мы, таблицу с толстой рамкой не видели, что ли? Вот и нет: результат выходит весьма любопытный (рис. 10.13). Внешняя рамка действительно толстая, а вот нижняя так и осталась тонкой. Почему? Потому что параметр border определяет толщину только *внешней* рамки.

Как мы уже убедились, если параметра border не указывать, то и рамки у таблицы не будет. Вообще никакой. Этим последним обстоятельством мы воспользовались в примере на рис. 10.4. И любой, кто провел в Internet больше получаса, подтвердит, что мы здесь далеко не первопроходцы: Web-дизайнеры пользуются этой особенностью очень широко.

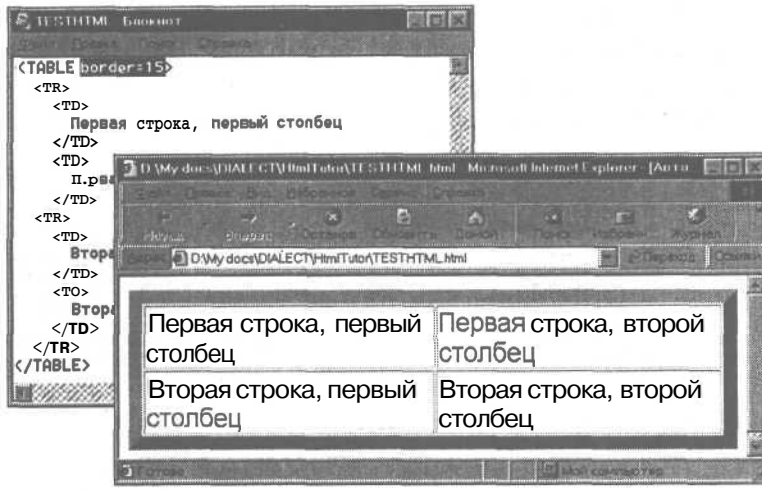


Рис. 10.13. Параметр border определяет толщину внешней рамки

Цвет рамки тоже приподнесет нам сюрприз. Как вы думаете, сколько параметров нужно, чтобы его описать? Один? Предположим. Действительно, такой параметр есть: bordercolor. По умолчанию рамка черно-серая. Попробуем заменить ее цвет, например, на зеленый:

```
<TABLE  border=15  bordercolor=green>
```

Если вы забыли, как в HTML задается цвет, загляните в главу 4, где описывается параметр color.

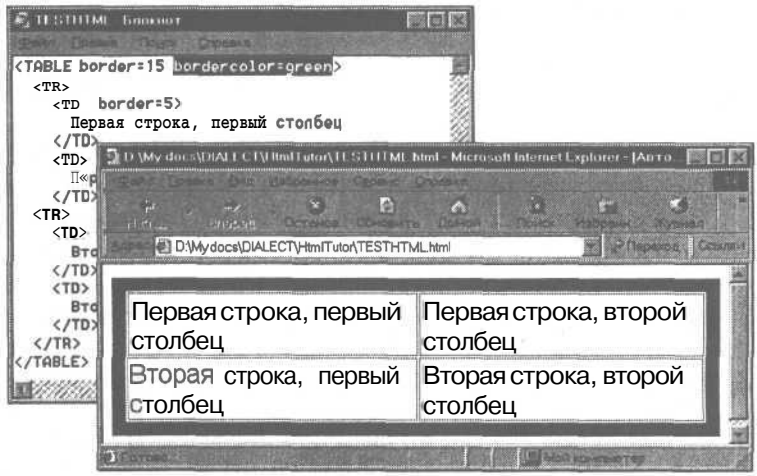


Рис. 10.14. Если определить цвет рамки, используя единственный параметр bordercolor, исчезнет эффект "выпуклости" (см. рис. 10.13)

Что получилось? Рамка и впрямь зазеленела, как молодая травка. Но — обратите внимание — исчезла свойственная ей выпуклость (рис. 10.14). Это естественно: ведь

эффект "выпуклости" обеспечивается разницей цветов верхне-левой и нижне-правой частей рамки. Для того чтобы изменить один из этих цветов, "не трогая" другой, используются параметры bordercolordark и bordercolorlight. Несмотря на свои названия, в переводе с английского означающие "темный цвет рамки" и "светлый цвет рамки", эти параметры никак не связаны с насыщенностью цвета. Параметр bordercolordark определяет цвет нижнего правого, а bordercolorlight – верхнего левого углов рамки. Например, если написать <TABLE border=15 bordercolorlight=red bordercolordark=blue>, получится некое подобие боксерского ринга.

Упоминания о светлом и темном цветах в названиях параметров bordercolordark и bordercolorlight появились благодаря тому, что по умолчанию верхний левый угол таблицы светлее нижнего правого: таблица как будто освещена, причем источник света находится в верхнем левом углу окна.

А как же с рамками отдельных ячеек? Можно ли отменить их отображение, задать цвет или толщину?

К сожалению, здесь наши возможности гораздо скуднее. Толщина внутренних рамок является фиксированной. Зато цвет можно задавать для каждой строки и ячейки в отдельности, как с помощью параметра bordercolor, так и по отдельным "углам" с помощью параметров bordercolorlight и bordercolordark.

Если в ячейке ничего нет, то рамки вокруг нее тоже не будет, как ни старайтесь. Для того чтобы ячейка *выглядела* пустой, но имела рамку, нужно "положить" в нее нечто невидимое. Как правило, такими "невидимыми" объектами являются неразрывный пробел   или прозрачный GIF-файл размером 1x1 пиксель (см. главу 8).

Частичное отображение рамок

Итак, если мы хотим получить рамку, нужно использовать параметр border, если не хотим — просто пропускаем его. А если хотим, но не везде? Например, как быть, если мы хотим оставить видимыми только вертикальные границы ячеек, как между газетными столбцами?

В дескрипторе <TABLE> есть два параметра, позволяющие "поиграться" с отображением разных частей рамок. К сожалению, они работают не во всех броузерах: это стандарт Internet Explorer. За отображение рамок отдельных ячеек (при этом внешняя рамка отображается всегда) "отвечает" параметр rules, а за отображение разных частей внешней рамки — параметр frame. Эти параметры принимают следующие предопределенные значения.

Если параметр rules принимает значение none, то отображается только внешняя рамка таблицы. Границы между ячейками становятся невидимыми. Если нужно, чтобы, кроме того, были видны еще и границы между строками, используем параметр rows, а если — между столбцами, то — cols (рис. 10.15). Обратите внимание, что все это имеет смысл только при ненулевом значении параметра border. Если ширина рамки равна нулю, то границ видно не будет. Если же присвоить параметру

frame значение all, то, независимо от значения border, границы всех ячеек будут видимы. Наконец, с помощью значения groups можно заключить в рамку группы ячеек. О том, как создать такую группу, читайте в разделе "Группировка ячеек".

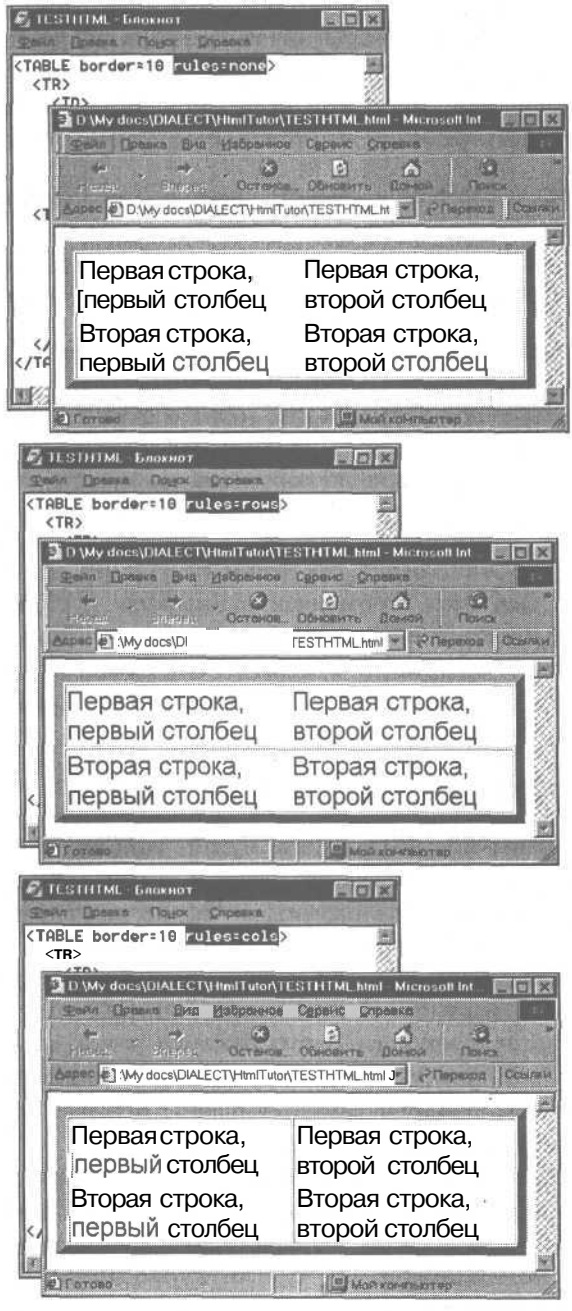
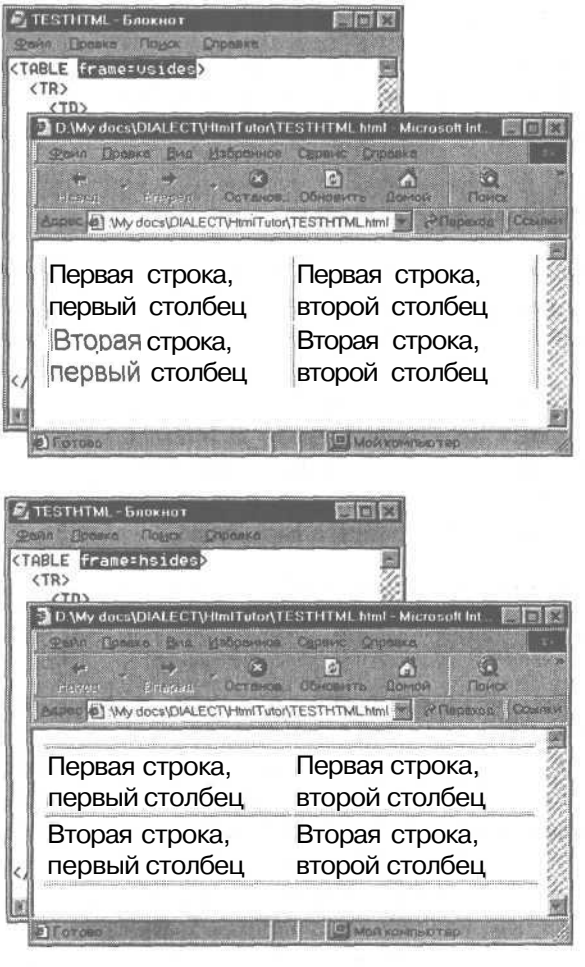


Рис. 10.15. Различные варианты использования параметра rules

Учитывая тот факт, что параметр border "понимают" все броузеры, чего нельзя сказать о параметре rules, вряд ли стоит использовать значение rules=all вместо `border=1`.

При использовании параметра frame подразумевается, что рамка таблицы существует. Поэтому указывать параметр border не обязательно. Для того чтобы оставить рамку только слева от ячеек, нужно присвоить параметру frame значение `lhs`, справа — rhs, по обеим сторонам — vsides. Если же нам требуются горизонтальные разделители, то нужно воспользоваться значением above (над ячейками), below (под ячейками) или hsides (сверху и снизу ячеек). Наконец, значение void позволяет добиться оригинального эффекта: если значение параметра border указано явно и не равно нулю, становятся видны только границы между ячейками, но не рамка вокруг всей таблицы (рис. 10.16). По умолчанию же используются значения border или box. Результат их использования одинаков и ничем не отличается от простого `border=1`.



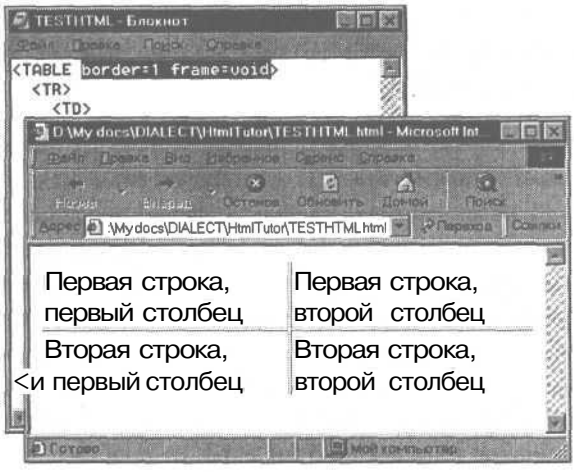


Рис. 10.16. Различные варианты использования параметра `frame`

Обратите внимание: поскольку эти параметры относятся ко всей таблице, то они определяются только в дескрипторе `<TABLE>` и не могут быть переопределены во внутренних дескрипторах таблицы для отдельных строк и ячеек.

Фон таблицы и ячеек

Наконец, нужно как-то описать фон ячеек. Если фоном служит обычная однотонная заливка, то ее цвет определяется параметром bgcolor. Значением этого параметра является код цвета — такой же, как при определении цвета шрифта (см. главу 4).

А как быть, если хочется "подложить" под текст ячейки более сложный фон например, с переходами цвета или узором? В Internet такие страницы встречаются сплошь и рядом. Для любых видов фоновой заливки, кроме однородного цвета, используются готовые графические файлы, которые "подключаются" к HTML-странице с помощью параметра background.

Если вы хотите освежить свои знания о графических файлах, используемых в Web-дизайне, перечитайте главу 8.

Выбирая фон для таблицы, нужно учитывать следующую особенность. Что произойдет, если изображение окажется больше, чем нужно? Скорее всего, вы сами можете ответить на этот вопрос: ничего хорошего — просто часть картинки окажется "обрезанной". А что будет, если картинка меньше области, занимаемой таблицей? Здесь мнения тех, кто хотел бы надеяться на лучшее, могут разделиться. Те, кто имеет в виду использование небольших повторяющихся изображений в виде узора, скажут, что картинка размножится на манер того, как это делается в "обоях" Windows. Те, кто предпочитает использовать одно большое фоновое изображение для всей таблицы, предположат, что оно растянется. Увы, таких опти

мистов ждет разочарование: если изображение, используемое в качестве фона для таблицы HTML, оказывается меньше той области, для которой оно предназначено, броузер *всегда* пытается "размножить" его на манер мозаики.

Последнее правило касается любых фоновых изображений — не только для таблиц, но и для всей HTML-страницы. Подробнее об этом читайте в главе 13.

И цвет фона, и фоновое изображение могут быть переопределены как для отдельных строк таблицы, так и для ячеек. Однако здесь следует учесть одну особенность: если фон, заданный для всей таблицы, заполняет также и промежутки между ячейками, то фон, определенный для строк и ячеек, заполняет только внутреннюю часть самих ячеек. При этом промежутки между ячейками (разумеется, если таковые существуют, т.е. когда значение параметра cellspacing не равно нулю) остаются заполненными тем фоном, который был определен для всей таблицы (рис. 10.17).

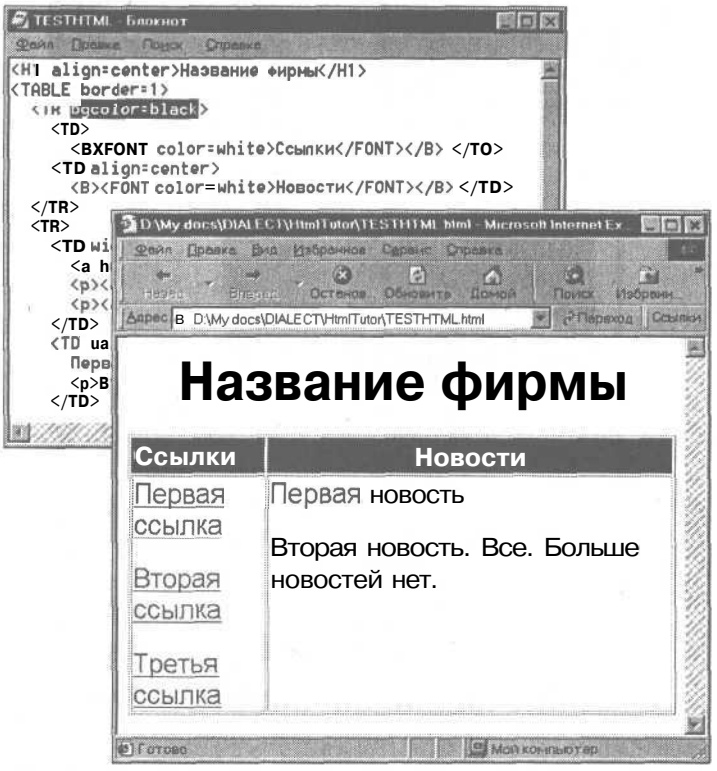


Рис. 10.17. Фон, заданный для ячеек строки, не распространяется на промежутки между ячейками

Слияние ячеек

Вот, наконец, мы и пришли к одному из самых интересных моментов, связанных с Web-таблицами: как создаются ячейки нестандартного размера, занимающие собой несколько столбцов или несколько строк? Как, например, получаются такие эффекты, как показаны на рис. 10.18?

Рис. 10.18. Эти варианты дизайна получены с помощью таблиц

Для слияния двух соседних ячеек в одной строке используется параметр rowspan, а для слияния смежных ячеек одного столбца — параметр rowspan. Оба эти параметра указываются в дескрипторе <TD> объединенной ячейки. Их значениями являются целые числа, обозначающие количество объединяемых ячеек (рис. 10.19).

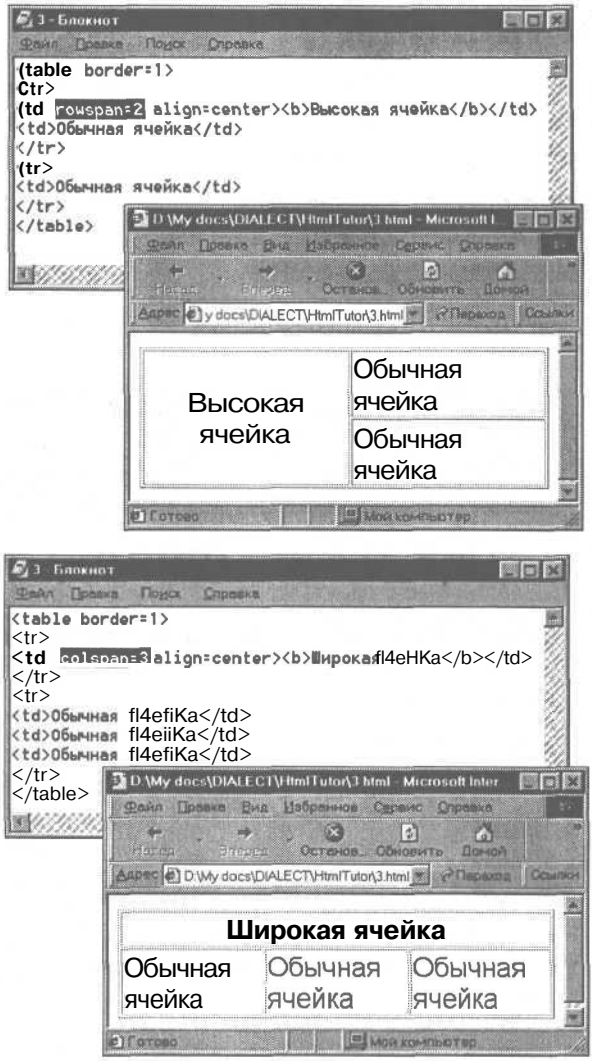


Рис. 10.19. Использование параметра `colspan` со значением 3 позволяет объединить три ячейки по горизонтали, а параметра `rowspan` со значением 2 — две ячейки по вертикали

Для того чтобы представить "в уме", как будет выглядеть в броузере то, что вы закодировали в HTML, требуется хорошее пространственное воображение. Поэтому часто дизайнеры создают таблицы в визуальных редакторах наподобие MS FrontPage, а потом долго "чистят" код. Или, как минимум, прежде чем писать код, сначала делают предварительный эскиз таблицы на бумаге.

Заголовок таблицы

В нашем примере для заголовка страницы использован логический дескриптор <H1>. Но при желании его можно включить в структуру таблицы — в качестве заголовка — с помощью дескриптора <CAPTION> (рис. 10.20).

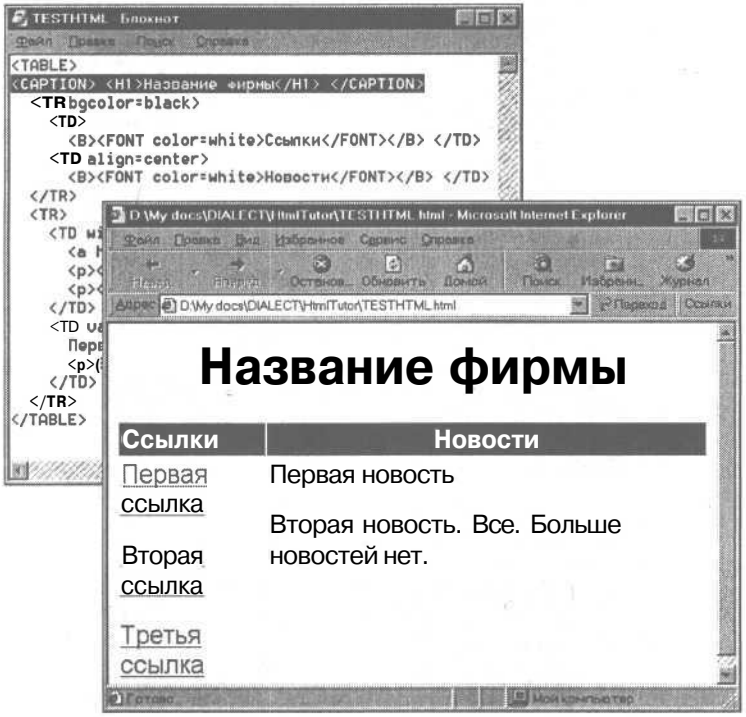


Рис. 10.20. Если в основе дизайна HTML-страницы лежит таблица, ее заголовок может играть роль названия страницы

Зачем это может понадобиться?

Во-первых, всегда приятно, когда некий виртуальный объект обладает всеми атрибутами своего реального прототипа. Ведь у обычных "бумажных" таблиц почти всегда есть если не надпись, так подпись. Во-вторых, если некий заголовок является неизменным спутником именно этой таблицы, то гораздо лучше жестко "связать" их, чтобы случайно не потерять при очередном копировании. Наконец, в-третьих, дескриптор <CAPTION> позволяет описать — к сожалению, не полностью — положение заголовка относительно таблицы. Надеюсь, вы обратили внимание на то, что в HTML-коде на рис. 10.20 отсутствует параметр выравнивания заголовка по центру. Тем не менее, заголовок расположен именно по центру. Это происходит потому, что такой режим выравнивания используется в дескрипторе <CAPTION> по умолчанию. Если же мы захотим изменить режим выравнивания, то нам следует задать значение параметра align явно: left (по левому краю) или right (по правому краю). Кроме того, если присвоить параметру align значение bottom, надпись переместится вниз, под таблицу.

А как же в этом случае задать горизонтальное выравнивание? Действительно, если параметру align уже присвоено значение top или bottom, мы не можем присвоить ему второе — left, right или center.

Поэтому для задания вертикального выравнивания лучше использовать второй параметр дескриптора <CAPTION> — valign. Этот параметр имеет всего два значения — top и bottom. Они определяют положение заголовка соответственно над или под таблицей.

Заголовки строк и столбцов

У многих таблиц есть "шапки" — заголовки строк и столбцов, оформление которых чем-то отличается от остальной таблицы. Например, в нашем примере роль такой шапки играют надписи "Ссылки" и "Новости". Несмотря на то, что они имеют одинаковое форматирование, мы кодировали это форматирование в каждой ячейке отдельно:

```
<TABLE>
      <TR>
            <TD align=center><B>Ссылки</B></TD>
            <TD align=center><B>Новости</B></TD>
      </TR>
...
</TABLE>
```

Но если "заголовочных" ячеек слишком много, то такое занятие может и надоесть. Конечно, выравнивание по центру можно вынести в дескриптор <TR>:

```
<TABLE>
      <TR align=center>
            <TD><B>Ссылки</B></TD>
            <TD><B>Новости</B></TD>
      </TR>
...
</TABLE>
```

Так уже лучше. Но, во-первых, к полужирному шрифту это не относится. А, во-вторых, как быть, если "шапка" затрагивает не только первую строку таблицы, но и, например, левый столбец?

Для того чтобы разметка таблиц не была слишком утомительна, в HTML предусмотрены специальные дескрипторы для "заголовочных" ячеек — <TH> (от *table header* — "заголовок таблицы"). Дескриптор <TH> полностью подобен дескриптору <TD>, за исключением того, что его содержимое обычно выводится полужирным шрифтом и выравнивается по центру (рис. 10.21). Поэтому дескрипторы <TH> можно использовать в любом месте таблицы вместо дескрипторов <TD>, а вовсе не обязательно только для верхней строки или левого столбца.

Конечно, возможности форматирования дескриптора <TD> небогаты. Например, он ничем не поможет нам, если мы захотим выделить "шапку" так, как показано на рис. 10.20. Зато благодаря использованию дескриптора <TD> можно значительно сократить код, упростить работу и, главное, обеспечить единообразие всех подобных элементов дизайна страницы.

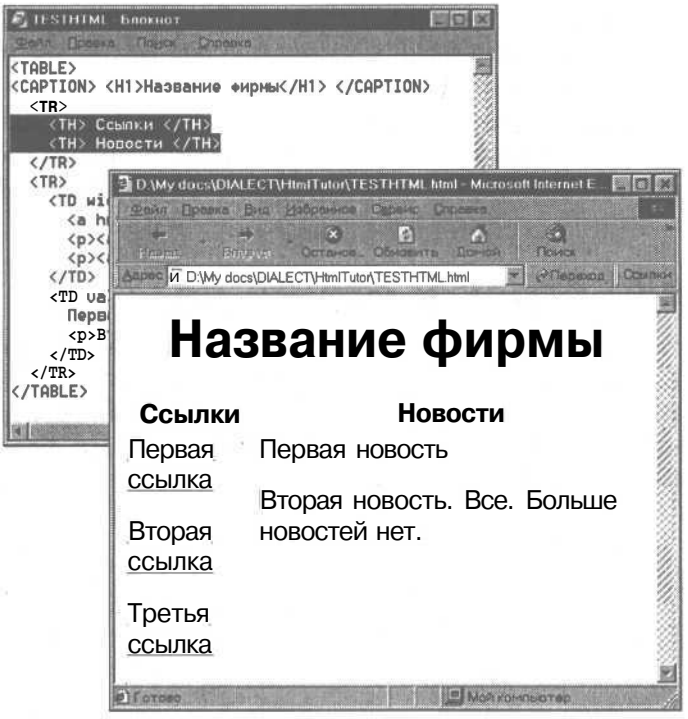


Рис. 10.21. Дескрипторы заголовков <TH> отличаются от дескрипторов ячеек <TD> только тем, что их содержимое отображается полужирным шрифтом и выравнивается по центру ячейки

Группировка ячеек

Эта идея с ячейками-заголовками выглядит какой-то незрелой... Ведь "шапки" таблиц — это зачастую целые массивы ячеек — строки, столбцы или еще большие группы. Соответственно, было бы логичнее описывать свойства всей группы, а не каждой ячейки в отдельности.

Некоторые свойства строки можно описать с помощью параметров дескриптора <TR>. Но как быть со столбцами? Предположим, что мы хотим отформатировать одни столбцы так, а другие — иначе. Пример? Пожалуйста. Предположим, что мы хотим на странице, изображенной на рис. 10.21, ввести третий столбец с датами новостей, причем ссылки должны выравниваться по центру, а новости с датами — располагаться на фоне другого цвета.

Конечно, все это можно сделать, введя соответствующие параметры каждой ячейки. Если не полениться. И нигде не ошибиться. Но гораздо удобнее и красивее описать свойства всего столбца. Для этого в HTML существуют следующие конструкции.

Свойства группы из нескольких смежных столбцов описываются дескриптором <COLGROUP>. Например, для того чтобы первый столбец таблицы выравнивался по центру, в то время как остальные выравниваются по левому краю, нужно написать такой код:

```
<TABLE>
<COLGROUP align=center>
<!-- содержимое таблицы -->
...
</TABLE>
```

Дескрипторы, описывающие свойства группы столбцов, обычно размещаются в начале таблицы.

А как быть, если требуется распространить форматирование на несколько столбцов? Конечно, можно выделить каждому из них по персональному дескриптору <COLGROUP>:

```
<TABLE>
<COLGROUP align=center> <!-- форматирование первого столбца -->
<COLGROUP align=center> <!-- форматирование второго столбца -->
<!-- содержимое таблицы -->
...
</TABLE>
```

Но до чего же это неизящно! Лучше воспользоваться специальным параметром дескриптора <COLGROUP> для объединения столбцов — span. Например, код <COLGROUP span=2 align=center> описывает группу из двух столбцов с выравниванием по центру. Как вы уже, вероятно, поняли, мы, строго говоря, могли бы в рассмотренном выше примере указать span=1, но не стали этого делать: зачем лишний раз усложнять код, когда это значение все равно подразумевается по умолчанию?

Похоже, с форматированием первых нескольких столбцов мы разобрались. Но как сообщить броузеру, что как раз начальные столбцы нужно пропустить и изменить форматирование, например, второго и третьего столбов таблицы? Ведь в HTML, в отличие от электронных таблиц, не существует нумерации строк и столбцов. Действительно, это, пожалуй, недоработка. Приходится пойти на хитрость: поставить дескриптор <COLGROUP> без параметров форматирования, указав лишь, сколько столбцов нужно "пропустить":

```
<COLGROUP span=n> <!-- пропустить n столбцов -->
<COLGROUP параметра следующей группы столбцов >
```

Какие еще параметры, кроме горизонтального выравнивания, можно присвоить группе ячеек с помощью дескриптора <COLGROUP>? Оказывается, их не так уж много. Это вертикальное выравнивание (параметр valign), ширина и высота ячеек (width и height) и цвет фона (bgcolor). Значения этих параметров и правила их присваивания такие же, как для дескрипторов ячеек <TD>.

Теперь, вооружившись всеми этими знаниями, мы можем наконец сгруппировать ячейки нашего примера так, как собирались: выровнять ссылки по центру, добавить столбец с датами новостей и "подложить" под новости вместе с датами фон другого цвета. Предлагаю вам решить эту задачу самостоятельно, а потом сверить результат с рис. 10.22.

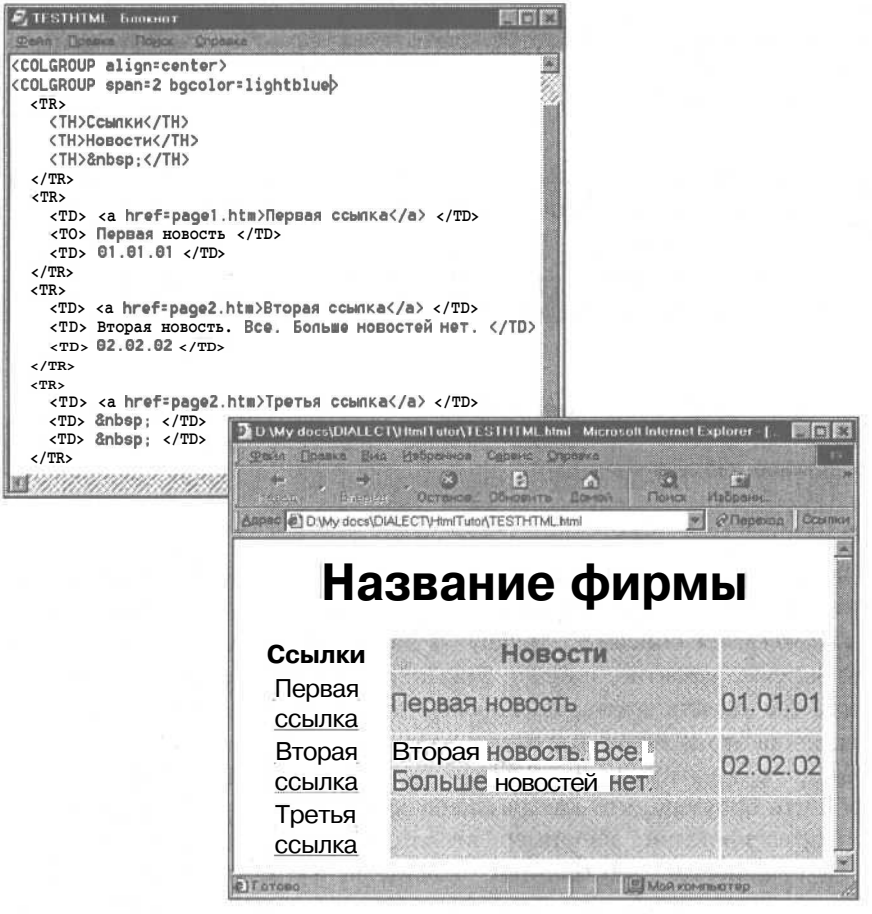


Рис. 10.22. Так можно использовать дескриптор <COLGROUP> для форматирования группы ячеек

Обратите внимание на пустые ячейки в первой и последней строках. Зачем они? Если их убрать, то исчезнут и наполняющие их цветные "квадраты", отчего нарушится дизайн страницы.

Возможно, у вас уже вертится в голове вопрос: дескриптор <COLGROUP> парный или непарный? И если парный, то что заключено внутри него?

Давайте подойдем к этому вопросу с другой стороны. Предположим, нам требуется конкретизировать свойства определенного столбца внутри группы. Например, выровнять новости по верхней границе ячеек, а даты — по нижней. Для этого используется дескриптор <COL>, расположенный внутри дескриптора <COLGROUP>. Этот дескриптор описывает параметры отдельного столбца, принадлежащего группе (рис. 10.23).

Какие параметры форматирования можно описывать с помощью дескриптора <COL>? Те же самые, что и с помощью дескриптора <COLGROUP>. И это логично: ведь дескриптор <COL> предназначен именно для переопределения свойств отдельных столбцов, входящих в группу <COLGROUP>.

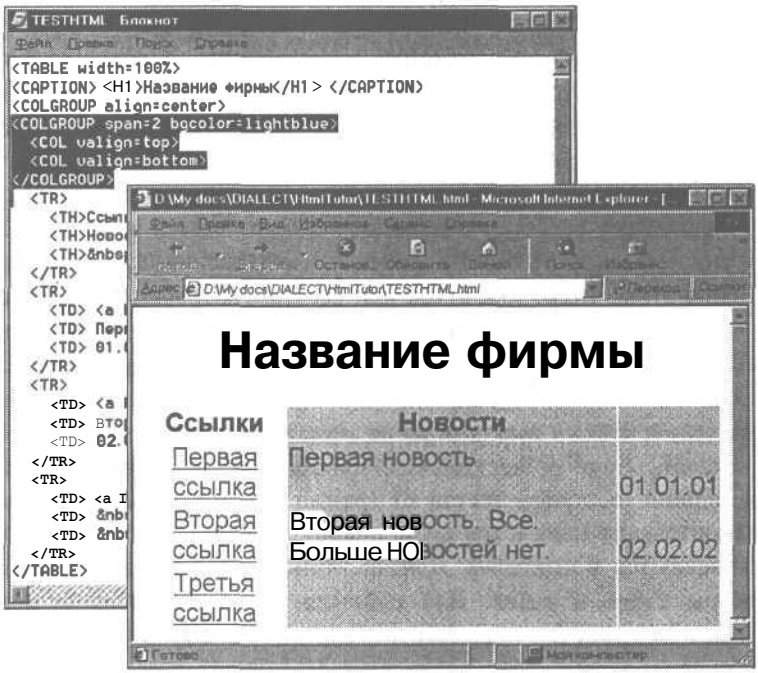


Рис. 10.23. Для того чтобы конкретизировать параметры отдельного столбца, принадлежащего группе, используется дескриптор <COL>, который помещается внутрь конструкции <COLGROUP>... </COLGROUP>

Кроме очевидных преимуществ, у таблицы, как средства Web-дизайна, есть и недостатки. В частности, содержимое таблицы выводится на экран только после того, как вся таблица будет получена с сервера. Поэтому, если таблица содержит много текста, посетитель страницы не может начать чтение, пока не загрузится все остальное: текст просто не появится на экране, не загрузившись целиком.

Кроме того, многие параметры таблиц, к сожалению, по-разному работают в разных броузерах. А часто встречаются ситуации, когда табличная верстка просто не позволяет решить поставленную задачу, например, если нужно снабдить прокруткой один из информационных блоков. Тогда приходится прибегать к другим конструкциям, таким как фреймы, которые мы обсудим в следующей главе.

Резюме

Таблицы используются в Web-дизайне чрезвычайно широко — не только для традиционных колонок цифр и прочей информации, которую удобно представлять в таком виде, но и как элемент дизайна. Для того чтобы расположить отдельные блоки информации в определенных местах страницы, такую страницу очень часто представляют в виде совокупности табличных ячеек.

Структура таблицы состоит из трех основных дескрипторов: <TABLE>, <TR> и <TD>. Дескриптор <TABLE> описывает параметры всей таблицы, а расположенные внутри него дескрипторы <TR> и <TD> — параметры строк и заключенных внутрь них отдельных ячеек, соответственно. Ячейки таблицы описываются построчно, слева направо.

С помощью параметров таблицы можно задавать ее ширину и высоту (параметры width и height), положение на странице (параметр align), толщину и цвет рамки (параметры border, bordercolor, bordercolordark и bordercolorlight), а также цвет фона (параметр bgcolor) и фоновое изображение (параметр background).

Многие параметры, такие как align или bgcolor, встречаются в нескольких дескрипторах (см. *Приложение Б)*. Но нужно помнить, что результат их применения в разных дескрипторах может отличаться. Так, например, параметр align в дескрипторе <TABLE> означает выравнивание таблицы относительно окна броузера, а в дескрипторах <TR> и <TD> — выравнивание текста относительно ячейки.

Следует сказать еще о нескольких параметрах, используемых в таблицах HTML. Для задания отступов текста от границ ячеек и расстояния между ячейками используются два параметра дескриптора <TABLE> — cellpadding и cellspacing. Отступы задаются в пикселях. Для частичного отображения рамок используются параметры frame и rules. Для слияния горизонтальных и вертикальных смежных ячеек применяются параметры дескриптора <TD> — colspan и rowspan.

В HTML предусмотрены также средства для определения заголовка и "шапки" таблицы — дескрипторы <CAPTION> и <TH>. Первый из них помещается внутри конструкции <TABLE>... </TABLE> и содержит заголовок всей таблицы. Второй используется вместо дескриптора <TD> для ячеек, которые нужно выделить.

Если все ячейки в строке имеют одинаковое форматирование, их параметры можно описать в объединяющем их дескрипторе <TR>. Если же такие ячейки образуют столбец, то их параметры описываются с помощью специального дескриптора <COLGROUP>. Количество смежных столбцов в группе определяется параметром span. Если же некоторому столбцу, принадлежащему группе, нужно присвоить дополнительные параметры форматирования, это делается с помощью дескриптора <COL>, расположенного внутри конструкции <COLGROUP>... </COLGROUP>.

При всех своих преимуществах таблицы имеют свои недостатки. В частности, они не позволяют снабдить прокруткой отдельные ячейки. Кроме того, содержимое таблицы не появляется на экране, пока не загрузится полностью. Эти особенности следует учитывать при выборе дизайна для страницы.

Тесты

1. Напишите код для верхней страницы на рис. 10.1.
2. Напишите код для нижней страницы на рис. 10.1.

Глава 11

Фреймы

В этой главе...

- Что такое фреймовая структура
- Горизонтальные и вертикальные фреймы
- Размеры фреймов
- Вложенные фреймы
- 4 Обрамление и отступы
- Ссылки
- Фрейм без фреймов

Что такое фреймовая структура

Вспомните, как туристы, наперебой щелкая фотозатворами, торопятся сохранить хоть крохи отпускной радости, солнца, счастья! Как будто все это можно смотать в рулончик пленки и положить в карман...Вам не кажется, что посетители Web-сайтов частенько делают нечто подобное?

Ну конечно же, делают: сохраняют понравившиеся страницы. Причем с гораздо большим успехом. Но иногда случаются промашки. Бывает, что сохраняется не весь код, а какой-то непонятный обрывок, и в окне броузера вместо яркой и интересной страницы огорченный пользователь видит окно, разделенное на две-три части, в каждой из которых написано: "Невозможно отобразить страницу" (рис. 11.1). И длинный список бесполезных советов...

Когда подобное случается с фотографиями и вместо довольных физиономий видишь только собственные пальцы сквозь проявленную пленку, как правило, все ясно: засветка. То ли пленка старая, то ли "мыльницу" пора на мыло. Но почему не сохраняется HTML-страница? Точнее, почему одни страницы сохраняются, а другие нет?

При более пристальном рассмотрении выясняется, что на самом деле сохранить такие страницы можно. Нужно только сохранять все, а не только HTML-код. В Internet Explorer для этого используется режим Web-страница, полностью. В результате такого сохранения мы получаем целый "выводок" файлов, один из которых "втягивает" в окно броузера остальные. Этот "главный" файл обычно самый маленький изо всех и содержит примерно такой код:

```
<FRAMESET параметры>
<FRAME src="source1.htm" name="frame1">
```

```
<FRAME src="source2.htm" name="frame2">
...
</FRAMESET>
```

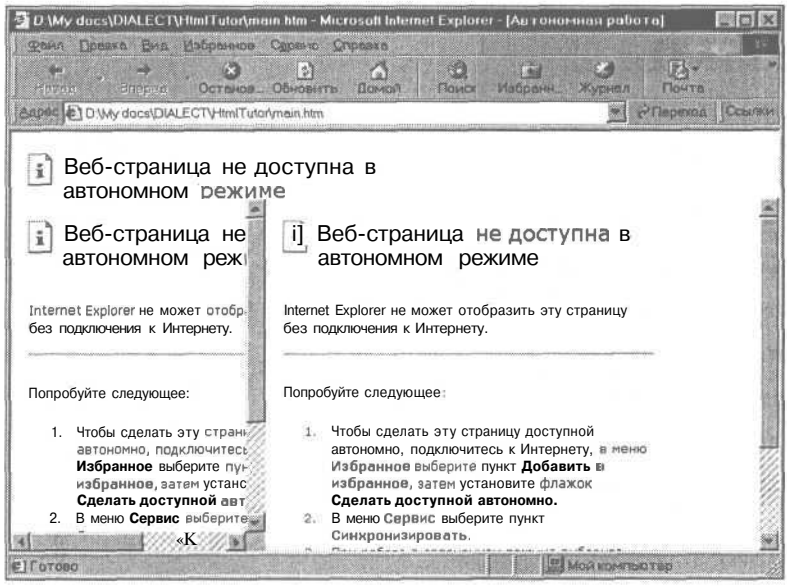


Рис. 11.1. Иногда вместо страницы сохраняется нечто непонятное

Первое, за что цепляется глаз в этой абракадабре, — имена файлов, которые служат значением параметра src. Этот параметр нам уже встречался при описании изображений (см. главу 8). Мы уже знаем, что он предназначен для "подключения" к странице внешних файлов. В данном случае это файлы с HTML-кодом содержимого фреймов. Именно эти файлы "выкачиваются" из Internet вместе с "главным". Именно они-то, как правило, и содержат нужную информацию.

Что же, в таком случае, содержится в "главном" файле?

Фреймовая структура.

Ну вот, еще один терминологический монстр... Но, с другой стороны, надо же *это* как-то называть! Собственно, все беды от того, что слово *фрейм,* подобно многим компьютерным терминам (чего стоит одно только слово *файл*!), осталось без нормального перевода на русский язык. Результат — барьер между теми, кто уже освоил эту премудрость (кстати, в остальном несложную), и теми, кто еще не дошел до этого блаженного состояния. Подозреваю, что именно подобным барьерам обязано своим появлением слово "чайник" в применении к homo sapiens.

Вариантов перевода английского слова *frame* на русский язык слишком много - "скелет", "костяк", "каркас", "рама"... И это далеко не все, а только наиболее употребительные. Остановимся на последнем варианте — "рама", "рамка". Web-дизайнеры любят делить окно броузера на более мелкие области и размещать в них логически обособленные части Web-страницы. Получается и красиво, и удобно.

Один способ такого разбиения нам уже знаком — таблицы (см. главу 10). Но у таблиц есть определенные недостатки. Главный из них – отсутствие прокрутки. Если содержимое ячейки не помещается в ней, то либо ячейка расширяется, либо посетитель чего-то не увидит. Средства стандартного HTML не позволяют снабдить каждую ячейку собственной полосой прокрутки. И тут-то на помощь приходят фреймы.

Как вы могли заметить из приведенного выше фрагмента кода, фреймовая структура состоит из нескольких файлов, объединенных с помощью конструкции <FRAMESET> (рис. 11.2).

Рис. 11.2. Фреймовая структура состоит из нескольких HTML-файлов, объединенных с помощью конструкции <FRAMESET>

Часто фреймы служат основой удобной и красивой системы навигации по сайту. Например, они позволяют разделить окно на "неподвижную" панель ссылок и основное поле, где выводится основная информация, интересующая посетителя. Конечно, то же самое можно сделать и без фреймов, но тогда пришлось бы создавать страницы, содержащие повторяющиеся фрагменты кода. В этом смысле фреймы обеспечивают значительную "экономию".

Однако у фреймов есть определенные ограничения, о которых следует помнить при конструировании страниц и разработке системы навигации. В первую очередь к таким ограничениям следует отнести невозможность при переходе по ссылке "одним щелчком" обновить сразу несколько фреймов. Вторым недостатком является уже упоминавшаяся сложность

при сохранении таких страниц: если посетитель не заметит фреймы, то он может сохранить только "каркас" страницы, без ее содержимого. Наконец, фреймы иногда вызывают сложности при общении с поисковыми серверами: посетитель, придя на страницу с поисковика, может попасть не во фреймовую структуру, а только на ее часть, открытую в отдельном окне (подробнее о работе поисковых серверов см. в главе 14).

Горизонтальные и вертикальные фреймы

Рассмотрим такой пример. Предположим, нам нужно разработать Web-сайт некоей фирмы, страницы которого делятся на две части: в левой находится одна и та же панель ссылок, а содержимое правой меняется в зависимости от того, какую ссылку выберет пользователь.

Вы могли заметить сходство этого примера с описанным в главе 10, посвященной таблицам. Это неудивительно: фреймы и таблицы занимают в мире HTML очень близкие ниши и часто соперничают за место под "виртуальным солнцем".

Что ж, попробуем. Создадим HTML-файлы для панели ссылок и списка новостей — references.html и news.html — и объединим их во фреймовую структуру (рис. 11.3).

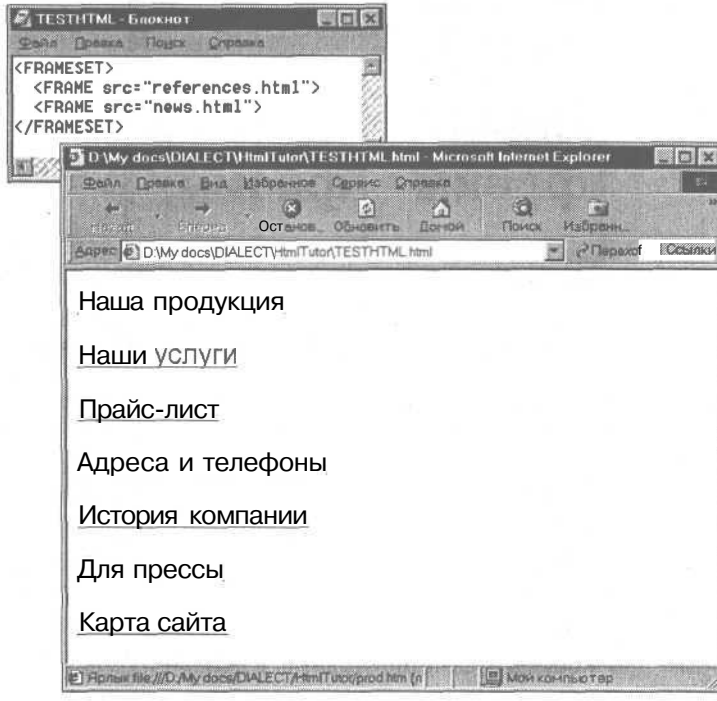


Рис. 11.3. Первая попытка объединить два файла во фреймовую структуру: один файл виден, другой — нет. Что-то тут не так...

Ссылки видны, а новости — нет! Что-то здесь не так...

Почему не видно второго фрейма? Потому что мы не описали *параметры* фреймовой структуры, а именно способ разбиения окна на фреймы — вдоль или поперек — и размеры областей.

Этот способ задается параметрами cols или rows, в зависимости от того, как требуется разделить окно броузера — по вертикали (на колонки) или по горизонтали (на строки). Значениями параметров cols или rows служат размеры областей разбиения — высота и ширина, соответственно, — которые перечисляются через запятую.

Размеры фреймов

Существует несколько способов задать размеры фреймов, на которые делится окно.

Самый простой и очевидный — *в пикселях*. Предположим, например, что ширина левого столбца должна равняться 250 пикселям, а правого — 550. Тогда эти значения можно задать с помощью кода, показанного на рис. 11.4. Вот теперь фреймовая структура работает правильно: в окне видны оба фрейма.

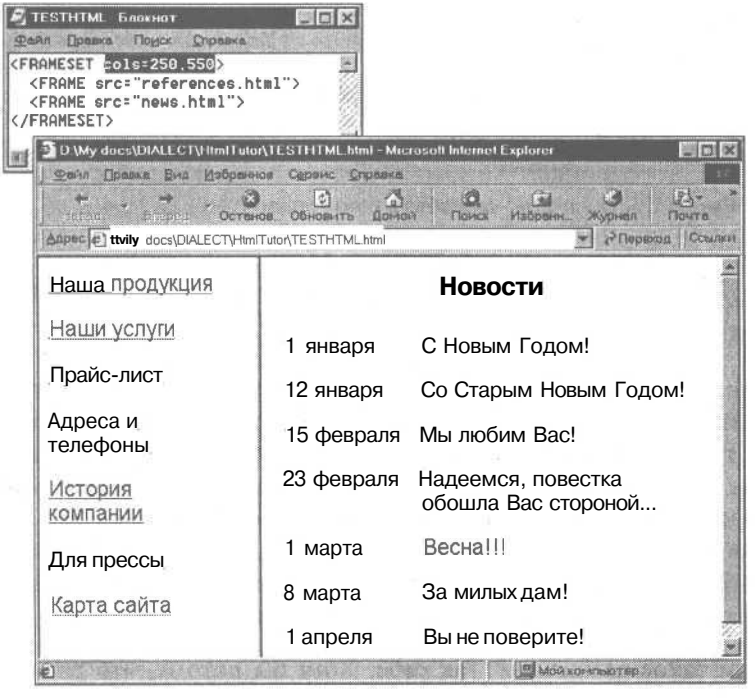


Рис. 11.4. Размеры вертикальных фреймов определяются параметром cols

А что получится, если использовать параметр rows? Проверим (рис. 11.5)... Такой код работает, но с колонками было все-таки лучше.

Действительно, вертикальные фреймы используются, пожалуй, более широко, чем горизонтальные.

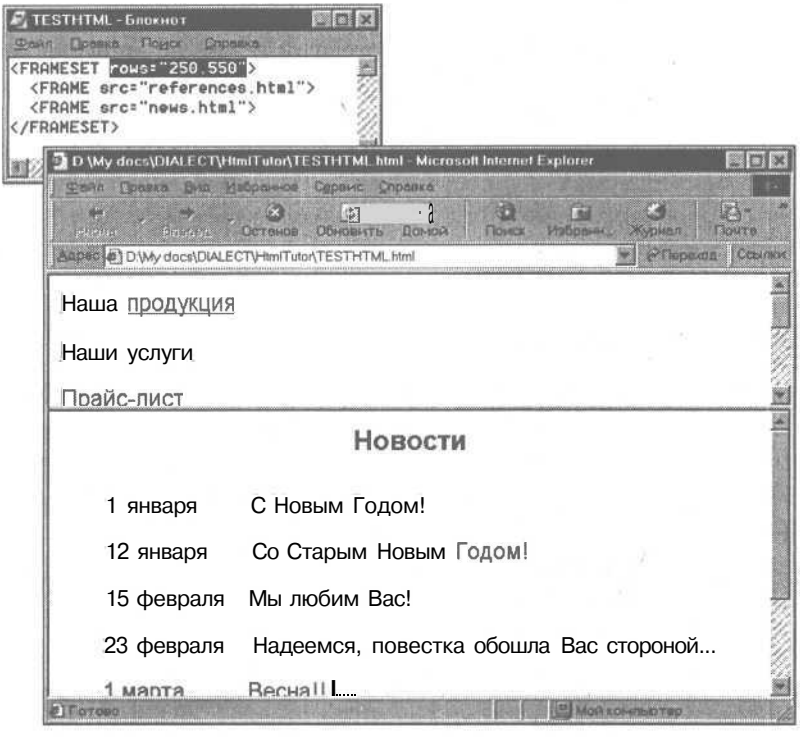


Рис. 11.5. Окно можно разбить и по горизонтали

А как должен выглядеть код, если окно делится на три колонки? Очень просто: после cols= через запятую идут не две, а три цифры. Что-то наподобие этого:

```
<FRAMESET cols=100,200,400>
```

Однако все хорошо в меру. Фрейм — не ячейка таблицы, которых могут быть десятки. Страницы из трех вертикальных фреймов, как, впрочем и из трех горизонтальных, — большая редкость. А уж о большем количестве и говорить нечего. Это объясняется, главным образом, громоздкостью фреймовых конструкций. Более мелкими блоками информации удобнее манипулировать, представив их в виде табличных ячеек, нежели в виде фреймов. Однако у фреймов есть свои преимущества, которые вы наверняка оцените, дочитав эту главу до конца.

Удобно ли такое разбиение? Так себе. Хотя бы потому, что размер окна броузера может измениться. И если броузеру не удается выдержать размеры фреймов согласно коду, то он делит окно по-своему.

Но ведь подобная задача у нас уже возникала — при задании размеров изображений и табличных ячеек. И там она успешно решалась с помощью относительных величин. Такой способ применяется и здесь: размеры фреймов можно задать *в процентах* (рис. 11.6).

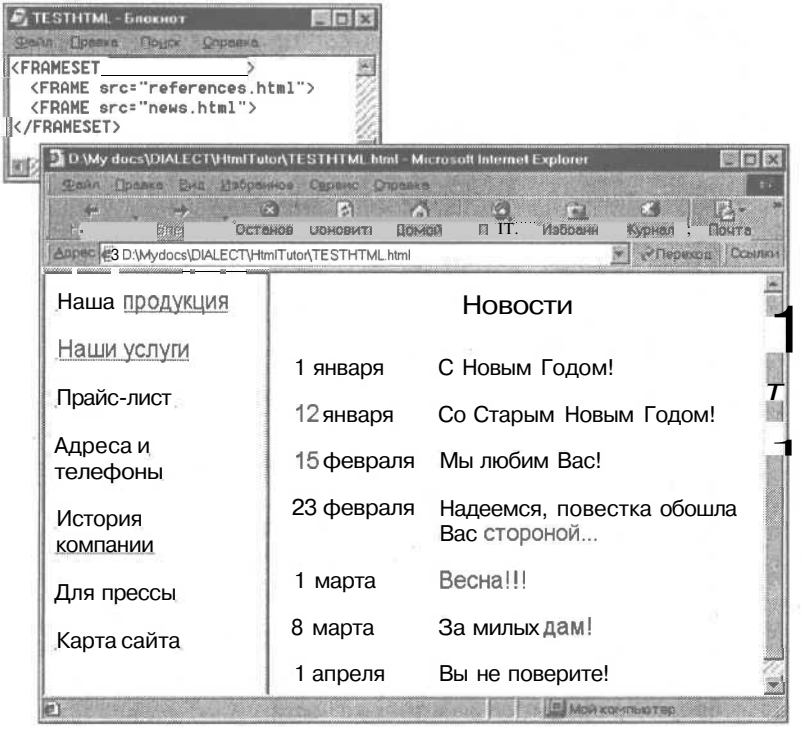


Рис. 11.6. Размеры фреймов можно задавать в процентах, например по правилу "золотого сечения"

Но ведь границу между фреймами можно перетаскивать и, таким образом, менять их размеры. Зачем тогда их задавать? Ведь зачастую это не принципиально. Нельзя ли пустить все на самотек, и пускай броузер сам разбирается?

Можно поступить и так — *не определять размеры областей.* Для этого цифры заменяются "звездочками", как при задании группы файлов с помощью маски. Тогда броузер будет делить окно на равные части (рис. 11.7).

Такой способ, конечно, простой, но результат выглядит, пожалуй, не вполне аккуратно. И посетителю страницы каждый раз придется самому перетаскивать границу фреймов, чтобы привести ее в более или менее удобный вид.

Однако описание в HTML-коде размеров обоих фреймов — занятие не только утомительное, но и явно излишнее: ведь и так ясно, что, если левый фрейм занимает 30% окна, то правый займет всю остальную часть, что бы мы там ни написали.

Именно поэтому рассмотренные выше способы редко применяются в "чистом виде". Гораздо чаще встречаются их комбинации. В самом деле, если мы хотим разделить окно на две части, из которых одна занимает 30% окна, зачем высчитывать и указывать явно, что вторая часть занимает 70%? Гораздо проще и логичнее написать, что первая часть занимает 30%, а вторая — все оставшееся пространство (рис. 11.8).

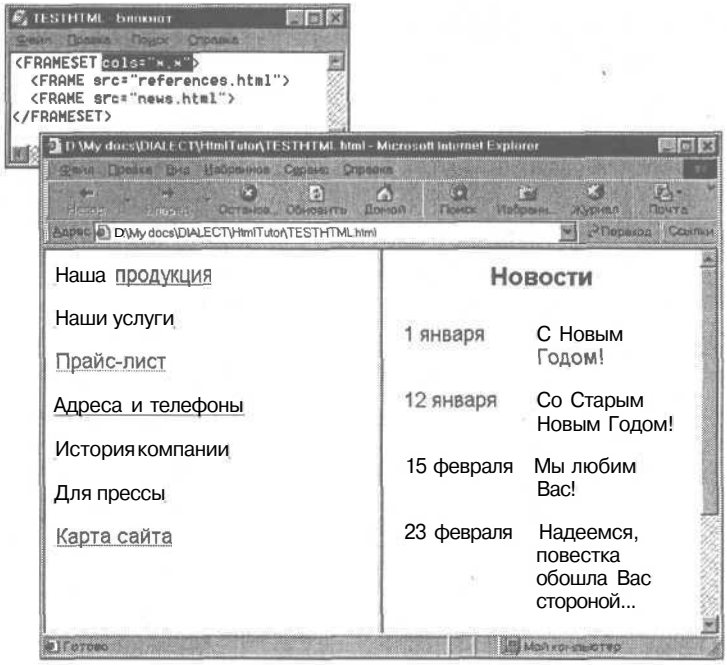


Рис. 11.7. Если заменить размеры областей "звездочками", броузер разделит окно на равные части

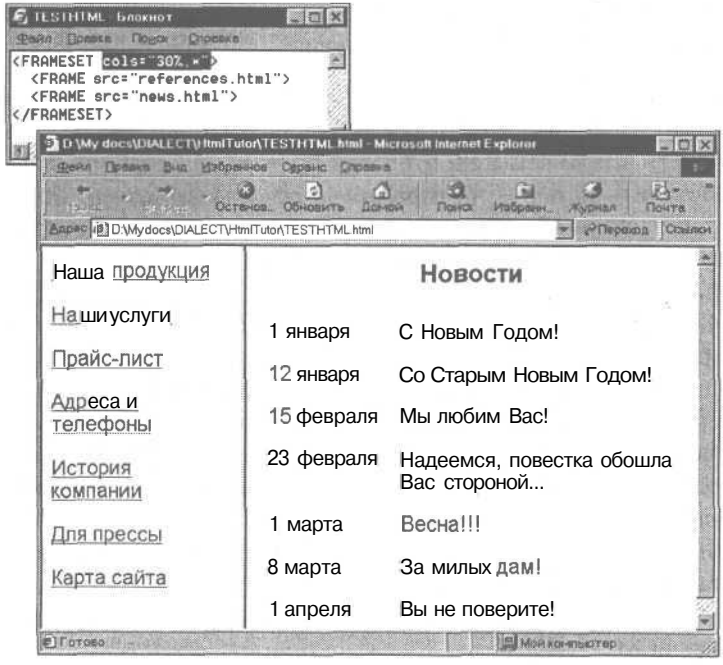


Рис. 11.8. Задавать ширину последнего фрейма не обязательно, вместо нее можно поставить "звездочку"

Наконец, есть еще один любопытный способ комбинированного задания размеров — специально для тех, кто не в ладу с процентами. Если перед "звездочкой" стоит цифра и, то окно делится так, чтобы эта часть была в *n* раз больше, чем остальные. Например, код

```
<FRAMESET cols=*,2*>
```

означает, что окно делится на две части, из которых вторая вдвое шире первой — первая часть займет 1/3, а вторая — 2/3 окна по ширине (рис. 11.9).

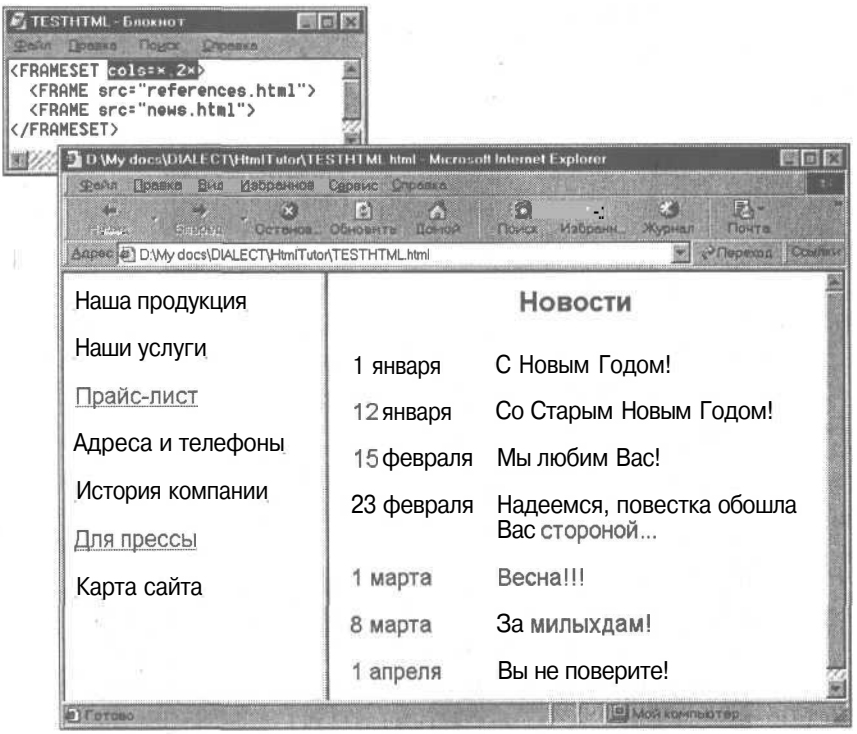


Рис. 11.9. Если перед "звездочкой" стоит цифра л, то окно делится так, чтобы эта часть была в п раз больше, чем остальные

Вложенные фреймы

Хорошо, а как разбить окно и по горизонтали, и по вертикали одновременно? Например, поместить вверху узкую горизонтальную панель с названием фирмы, а под ней две вертикальные: поуже — со ссылками и пошире — с новостями? Достаточно ли перечислить в дескрипторе <FRAMESET> размеры всех фреймов с помощью параметров rows и cols (рис. 11.10)? Нет, похоже, это не то, что нам нужно...

Но на самом деле все гораздо проще, чем мы думали: сначала окно разбивается на горизонтальные фреймы, а затем один из горизонтальных фреймов — на вертикальные. Или, наоборот, в зависимости от того, что мы хотим получить. Другими словами, структуры <FRAMESET> бывают вложенными (рис. 10.11).

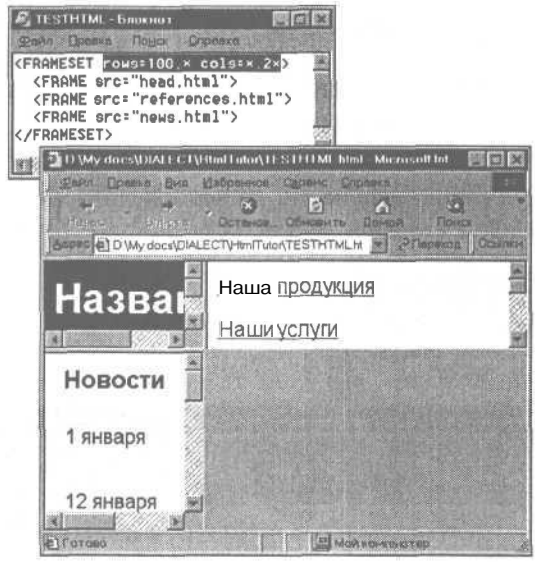


Рис. 11.10. Если просто перечислить в дескрипторе <FRAMESET> размеры и положение всех фреймов, ничего хорошего не получится

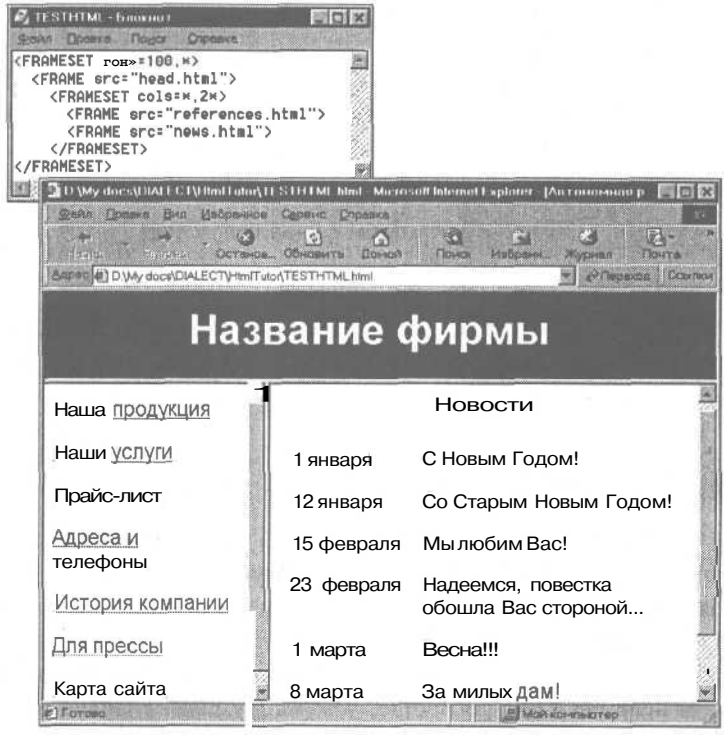


Рис. 11.11. Для того чтобы разделить окно на три фрейма — горизонтальный и два вертикальных, — используйте вложенную фреймовую структуру

Обрамление и отступы

Итак, как вы могли заметить, по умолчанию фреймы разделяются обычными серыми рамками Windows, которые можно перетаскивать с помощью мыши. Если содержимое фрейма не помещается в отведенной ему области окна, то у этой области появляется собственная полоса прокрутки. При перетаскивании ее бегунка перемещается только содержимое соответствующего фрейма; все остальное содержимое окна не изменяется.

Иногда в интересах дизайна требуется убрать рамку, изменить ее цвет или толщину, а также запретить прокрутку и сделать невидимыми полосы прокрутки. Рассмотрим параметры фреймовой структуры, позволяющие это сделать.

Итак, наличие рамки. За него "отвечает" параметр `frameborder`, который принимает одно из двух значений — yes или по. Вместо yes можно ставить цифру 1, а вместо по — 0. Впрочем, как вы, вероятно, догадываетесь, вместо `frameborder=yes` можно вообще ничего не ставить: это значение подразумевается по умолчанию.

Что получится, если в главном фрейме нашего примера поставить frameborder=0 - предсказать нетрудно. Рамки просто исчезнут, оставив "неприкаянные" полосы прокрутки (рис. 11.12).

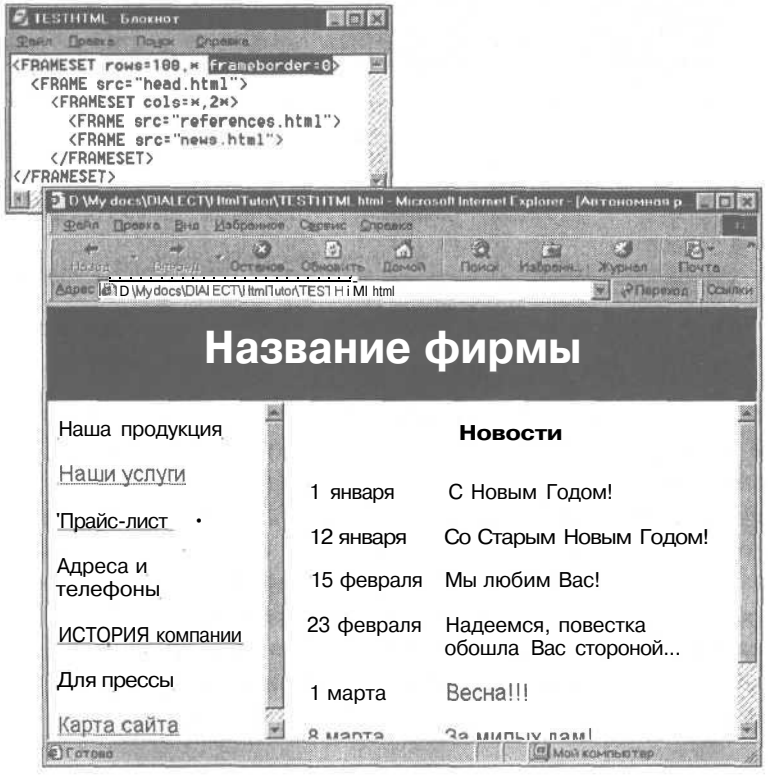


Рис. 11.12. Если в главном фрейме поставить frameborder=0, рамки исчезнут

Но самое интересное заключается в том, что параметр frameborder можно ставить не только здесь, но и во вложенных дескрипторах <FRAMESET>, и в дескрипторах <FRAME>, описывающих параметры отдельных фреймов. И в каждом случае результат будет выглядеть по-разному. Например, если в нашем примере присвоить такой параметр внутреннему дескриптору <FRAMESET>,то рамка между вертикальными фреймами исчезнет. Останется только горизонтальная рамка, как бы отчеркивающая панель с названием компании от остальной информации (рис. 11.13).

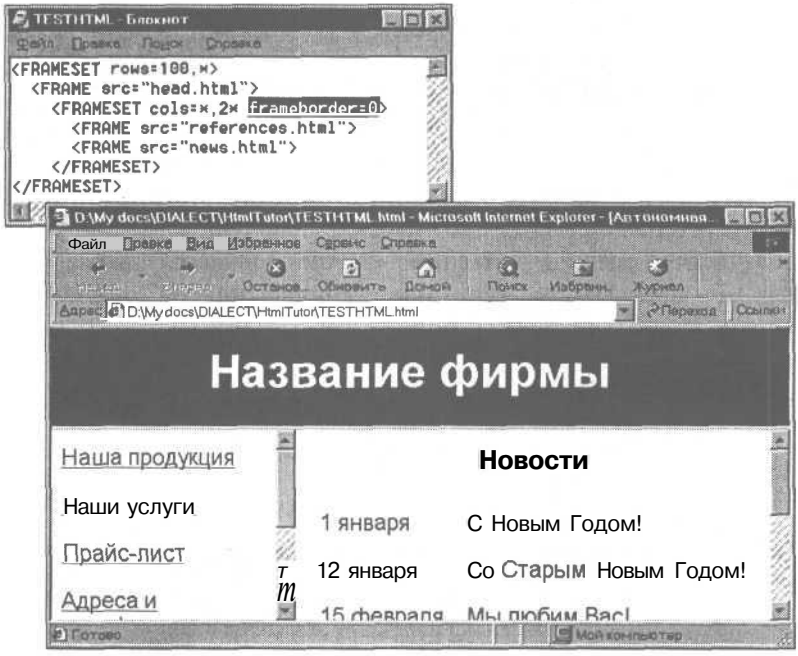


Рис. 11.13. Если поставить frameborder=0 во внутреннем дескрипторе <FRAMESET>, то рамка между вертикальными фреймами исчезнет

Толщина рамки задается в пикселях уже знакомым нам параметром border. Так, вместо стандартной рамки можно сделать толстую (рис. 11.14). Однако этот параметр применим только для дескрипторов <FRAMESET>.

По умолчанию рамка серого цвета, как и весь интерфейс Windows. Но, в отличие от последнего, ее цвет можно изменить, используя средства HTML, например параметр bordercolor. Этот параметр нам уже встречался — при определении цвета рамок таблиц. Поэтому вы уже не хуже меня должны знать, что является его значениями — мнемоническое имя или код цвета в формате #RRGGBB. Отметим также, что параметр bordercolor применим не только для всей фреймовой структуры, но и для отдельных фреймов (рис. 11.15).

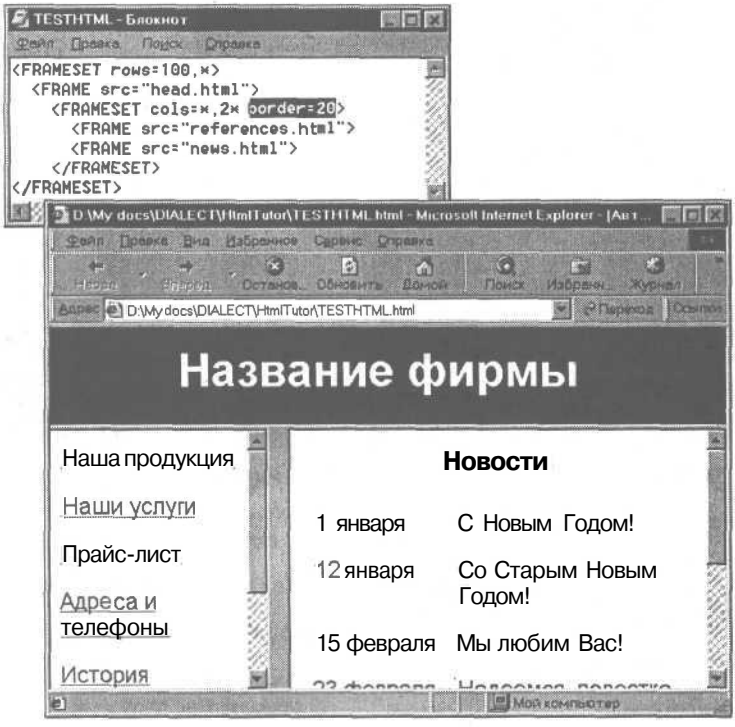


Рис. 11.14. С помощью параметра border можно из стандартной рамки сделать толстую

Наконец, нам остался еще один параметр — `framespacing`. Параметр весьма любопытный. Измеряется он в пикселях. Если мы поставим его и попробуем, что выйдет, то не получим ничего особенно интересного: просто рамки фреймов станут такой толщины, какая указана в этом параметре. Но если убрать рамку, все становится ясно: этот параметр делает то же, что и параметр cellspacing в таблицах — задает расстояние между фреймами. Результат его применения особенно наглядно смотрится, если "подложить" под фреймы цветной фон (рис. 11.16).

О том, как "подложить" фон под всю страницу, вы узнаете в главе 13.

Несмотря на то, что результат применения параметра framespacing лучше виден при цветных фреймах, вряд ли такой дизайн выглядит красиво. Страница смотрится лучше без этой белой полосы. Поэтому, если мы хотим увеличить расстояние между фреймами, то нам нужно либо отказаться от цветного фона, либо использовать для задания отступов другие средства.

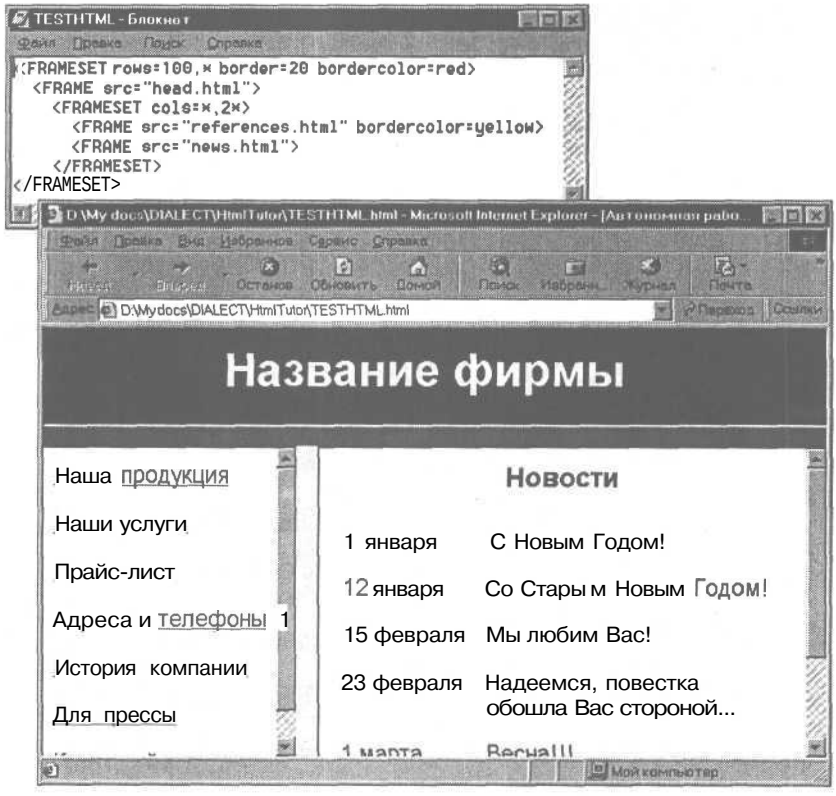


Рис. 11.15. С помощью параметра bordercolor рамки фреймов можно раскрасить в разные цвета

Этими средствами являются параметры marginheight и marginwidth. Они принадлежат дескриптору <FRAME> и определяют вертикальный и горизонтальный отступы содержимого фрейма от рамки. Измеряются эти отступы в пикселях (рис. 11.17).

Вообще, удачное сочетание параметров, описывающих свойства рамок, может сделать страницу прекрасной, а неудачное — отвратительной. Как добиться первого и избежать второго? Всего лишь проверять, как выглядит то или иное "украшение" при разных размерах окна и шрифта.

Вас уже, вероятно, давно интересует вопрос: зачем все эти ухищрения с положением рамок и отступами, если пользователь может просто взять и перетащить рамку, куда ему заблагорассудится?

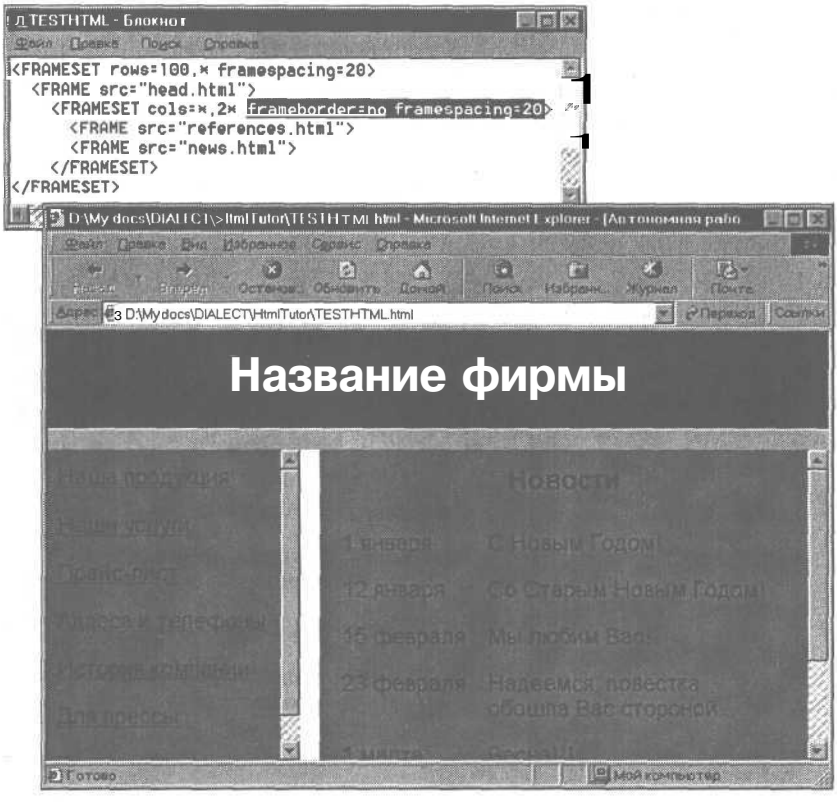


Рис. 11.16. Параметр framespacing определяет расстояние между фреймами и используется преимущественно для фреймов без рамки

Конечно, рамку можно просто убрать: нет рамки — не за что перетаскивать. Но ведь иногда нужно оставить рамку, но сделать ее неподвижной. Как этого добиться?

Для этого используется специальный параметр дескриптора <FRAME> – noresize. Значений у него нет. Просто, если он упомянут, то перетащить границу фрейма не удастся. Например, если изменить код нашего примера на следующий

```
<FRAMESET rows=100,*>
  <FRAME src="head.html" marginheight=5 noresize>
    <FRAMESET cols=*,2*>
      <FRAME src="references.html">
      <FRAME src="news.html" marginwidth=50>
    </FRAMESET>
</FRAMESET>
```

то границу между вертикальными фреймами, содержащими информацию из файлов references.html и news.html, по-прежнему можно будет перемещать вправо-влево, но граница верхнего фрейма с файлом head.html останется неподвижной.

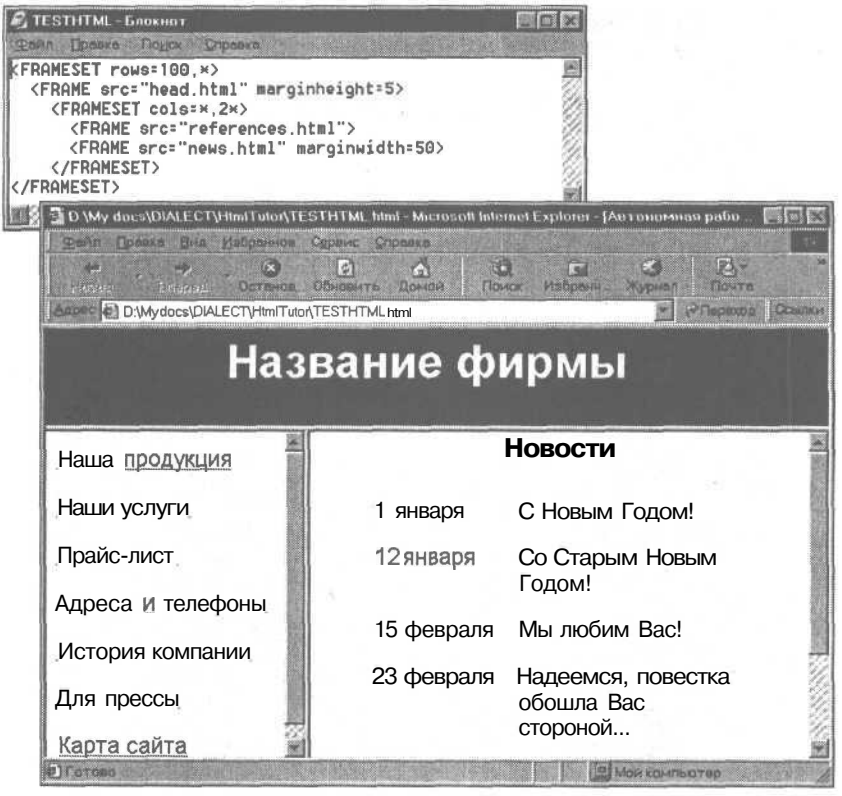


Рис. 11.17. Параметр `marginheight` определяет отступ содержимого фрейма от верхней рамки, а raarginwidth — от правой и левой рамок

Наконец, рассмотрим последний эффект, связанный с обрамлением фреймов. Существует довольно распространенный дизайнерский прием, который заключается в создании неподвижного фрейма. Как правило, такие фреймы содержат декоративные элементы — заголовки, эмблемы. Разумеется, их размер должен быть таким, чтобы содержимое помещалось в окне целиком, иначе теряется все впечатление, и полосы прокрутки его только портят.

В таких случаях, для того чтобы запретить прокрутку и убрать с экрана соответствующие полосы прокрутки, используется специальный параметр дескриптора `<FRAME>` — scrolling. Вообще-то, этот параметр имеет три значения — yes, no и auto. Первое означает обязательное наличие полос прокрутки независимо от того, нужны они или нет, второе — отсутствие полос прокрутки и запрет прокрутки даже в том случае, если она не мешает. Последнее значение соответствует состоянию фрейма, в котором он находится по умолчанию: если содержимое фрейма полностью помещается в отведенной ему части окна, то полосы прокрутки отсутствуют; если же часть содержимого выходит за пределы видимости, то такие полосы появляются.

Естественно, чаще всего используется режим scrolling=no. Так, в нашем примере верхний фрейм является как раз тем самым декоративным элементом, прокрутку которого было бы логично запретить (рис. 11.18)

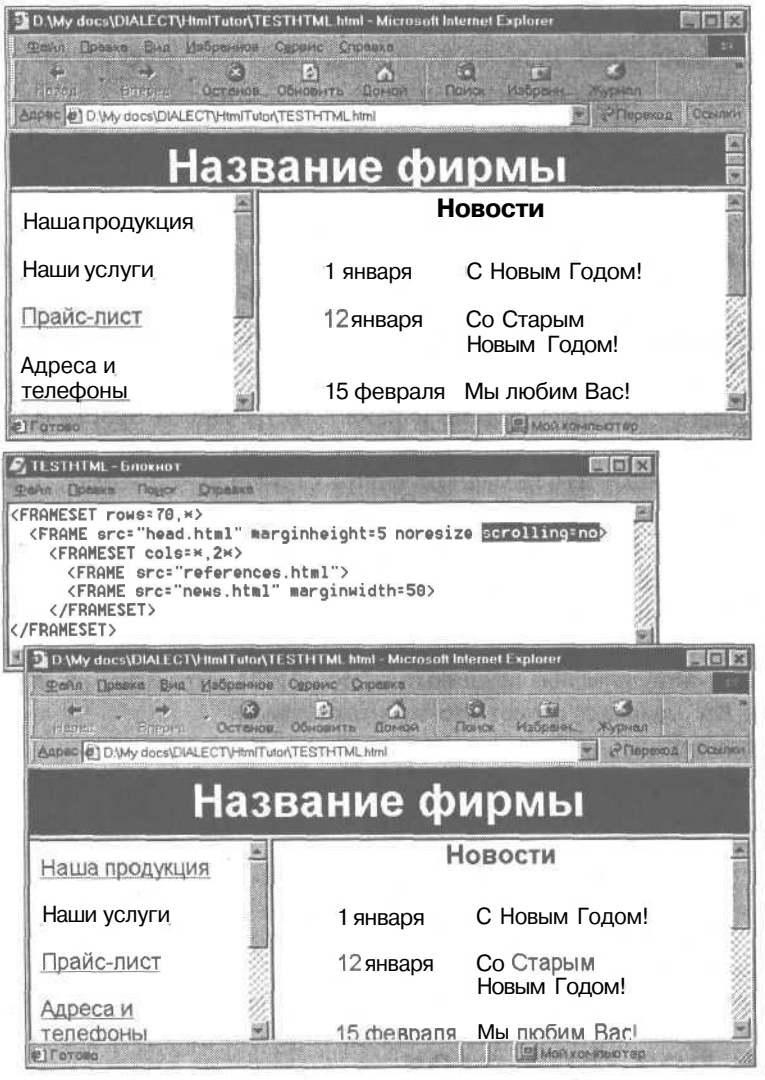


Рис. 11.18. Для того чтобы ненужные полосы прокрутки не появлялись на экране, используйте параметр scrolling со значением no

Ссылки

Левая панель, сплошь состоящая из гиперссылок, наводит на мысль: в какой же части окна, разделенного на фреймы, теперь будут выводиться страницы, на которые посетитель перешел по ссылке?

По умолчанию они открываются в том же фрейме, в котором находилась ссылка. А для случая, когда нужно поступить иначе, используются *именованные фреймы*. Имя фрейму присваивают с помощью параметра name:

```
<FRAME src="news.htm" name=mainframe>
```

Для того чтобы при переходе по ссылке новая страница открывалась на том месте, где раньше были новости, воспользуйтесь параметром target дескриптора <A>:

```
<A href="prod.html" target=mainframe>Наша продукция</a>
```

Для того чтобы освежить знания о гиперссылках, советую еще раз просмотреть главу 9.

При работе с фреймами удобно использовать зарезервированные имена окон, применяемые в качестве значений параметра target: _top и _parent позволяют загрузить страницу в том же окне, на месте всей фреймовой структуры; _self — в том же фрейме, _blank — в новом окне.

Иногда — правда, все реже и реже — встречаются броузеры, которые не поддерживают фреймы. Специально для них в HTML оставлен дескриптор <NOFRAMES>. Что помещают внутрь структуры <NOFRAMES>... </NOFRAMES>? Лучше всего, если удается (и если у создателя страницы есть такое желание) поместить там версию страницы без фреймов — пусть не так красиво, но все же хоть что-то... Но, поскольку броузеры, не поддерживающие фреймы, скоро будут встречаться примерно так же часто, как динозавры, время дескрипторов <NOFRAMES> тоже проходит. И если уж какой-нибудь особо вежливый автор HTML-страницы их использует, то он, как правило, пишет что-то в таком роде:

```
<FRAMESET cols=2*,*>
   <FRAME src="left.html">
   <FRAME src="right.html">
   <NOFRAMES>
         К сожалению, Ваш броузер устарел.
         Обновите его и приходите снова!
   </NOFRAMES>
</FRAMESET>
```

Фрейм без фреймов

Используя фреймы, удобно делить окно на две вертикальные или горизонтальные панели — или, как в нашем примере, на три. Но довольно часто приходится решать такую дизайнерскую задачу: создать на странице внутреннее, "вложенное" окно, с собственной полосой прокрутки и показывать в нем какую-то информацию.

Конечно, можно решить такую задачу с помощью "классических" фреймов, "отрезав" по сторонам окна куски соответствующей ширины (рис. 11.19). На рисунке специально оставлены рамки фреймов, чтобы было видно, как разделено окно. Понятно, что на реальной странице эти границы лучше убрать.

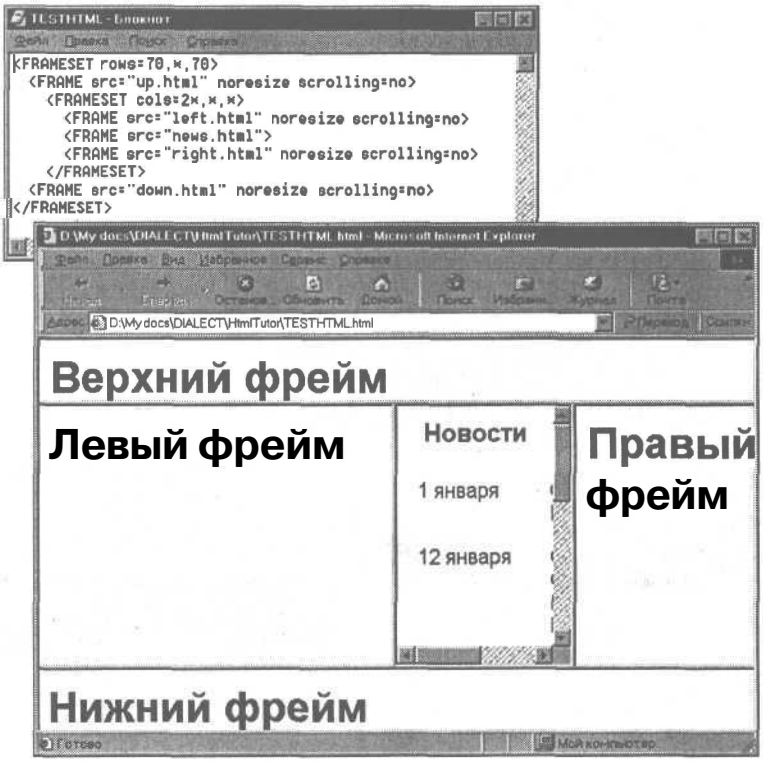


Рис. 11.19. Задачу создания на странице центрального "окна" с прокруткой можно решить с помощью "классической" фреймовой структуры. Но это громоздко и неудобно

И все-таки такое решение далеко от идеала. Хотя рамки и можно убрать, границы между фреймами все равно остаются, и с ними приходится считаться. И это уже не говоря о прочих неудобствах, связанных с тем, что цельный по замыслу кусок страницы приходится резать на четыре части, размещать в четырех файлах — left.html, `right.html`, `up.html` и `down.html` — и потом совмещать границы фреймов так, чтобы внешне все смотрелось как единое целое.

Поэтому для этой задачи существует гораздо более простое решение, так называемый *встроенный фрейм*.

Для создания встроенного фрейма используется дескриптор `<IFRAME>` (от английского *inline frame)*. Это парный дескриптор, очень похожий на дескриптор `<FRAME>`. Но дескриптор `<FRAME>` — непарный, а `<IFRAME>` — парный! Что же тогда находится между `<IFRAME>` и `</IFRAME>`?

Здесь самое время вспомнить, что, несмотря на то, что HTML — стандарт, далеко не все дескрипторы этого стандарта одинаково поддерживаются всеми броузерами. Именно на этот случай рассчитано содержимое дескриптора `<IFRAME>`: там, как правило, помещают информацию, которая как-нибудь компенсирует посетителю тот печальный факт, что его броузер не поддерживает встроенные фреймы (рис. 11.20).

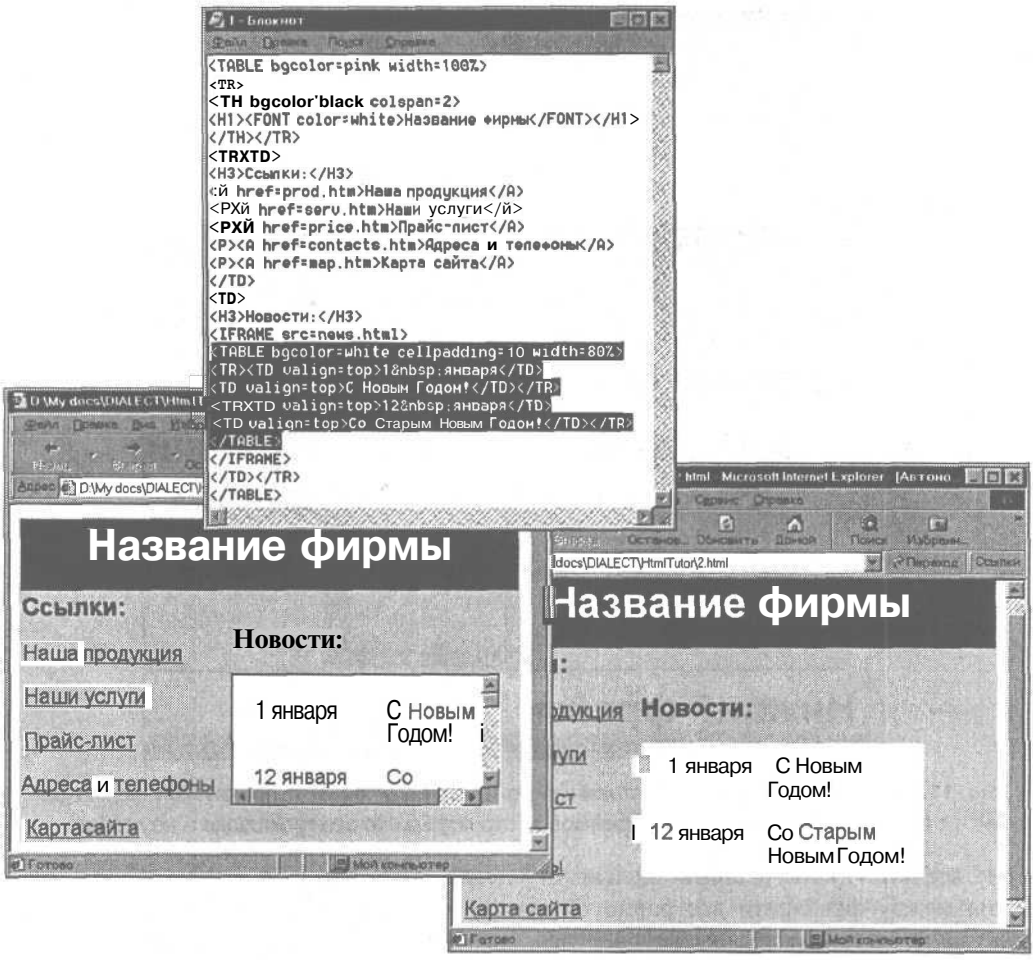


Рис. 11.20. Содержимое дескриптора <IFRAME> предназначено на тот случай, если броузер не поддерживает этот дескриптор

В принципе если вы не предусматриваете замены страницы с фреймом, закрывать дескриптор <IFRAME> не обязательно.

Для простоты в примере на рис. 11.20 дескриптор <IFRAME> имеет только один параметр — src. Значением этого параметра, как обычно, является файл, "подключаемый" для отображения в данной точке, — news.html. Однако отсутствие прочих параметров отрицательно сказывается на эстетичности страницы.

В частности, с помощью параметров width и height мы можем отрегулировать ширину и высоту встроенного фрейма. Как всегда, эти параметры могут измеряться в абсолютных величинах (пикселях) или в относительных (процентах от

размеров окна или табличной ячейки, в которую "вписан" фрейм). Благодаря этим параметрам мы можем расширить окно новостей в нашем примере (рис. 11.21), а то его и окном-то назвать неловко: так, форточка...

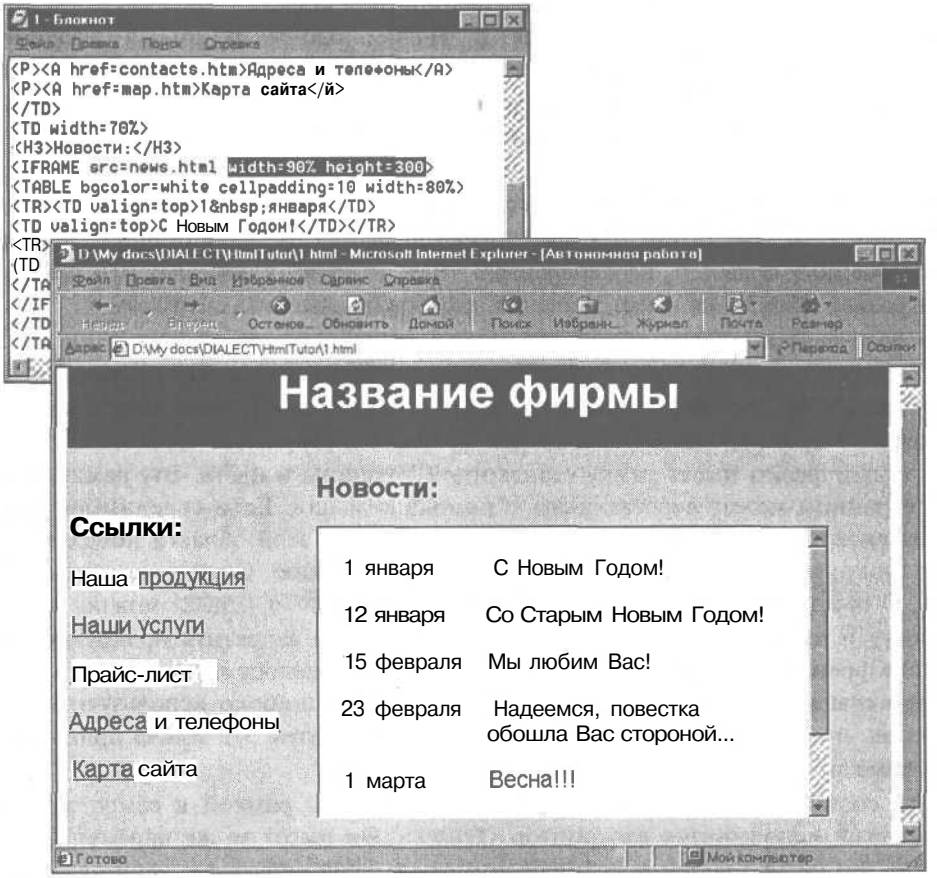


Рис. 11.21. Для того чтобы расширить встроенный фрейм, воспользуемся параметрами width и height — их можно задавать как в пикселях, так и в процентах

Как вы могли заметить, точно такие параметры есть в дескрипторе <FRAME>. Вообще, фрейм обычный и фрейм встроенный в смысле параметров очень похожи. Как и в <FRAME>, в дескрипторе <IFRAME> внутренние отступы — расстояния от границ встроенного фрейма до его содержимого сверху, снизу, справа и слева — определяются параметрами marginheight и marginwidth. Как и там, здесь можно отменить отображение рамки с помощью параметра frameborder и изменить режим прокрутки с помощью параметра scrolling. Наконец, встроенный фрейм, как и обычный, может иметь имя, описываемое параметром name, чтобы к нему можно было обращаться по гиперссылке.

Резюме

Фреймы — мощное средство HTML, позволяющее разделить пространство окна броузера на самостоятельные информационные зоны.

Фреймовая система описывается в HTML-коде с помощью дескриптора <FRAMESET>. В нем помещаются дескрипторы <FRAME>, описывающие параметры отдельных фреймов. HTML-коды содержимого этих фреймов находятся в отдельных файлах, имена которых указаны в дескрипторах <FRAME> с помощью параметров src.

Деление окна броузера на части осуществляется с помощью параметров дескриптора <FRAMESET>. Для деления по вертикали используется параметр cols, а по горизонтали — параметр rows. Их значениями являются размеры (ширина или высота) полученных областей, перечисленные через запятую, в пикселях или процентах. Если размер области не имеет значения, его можно заменить символом *.

В одном дескрипторе <FRAMESET> может присутствовать только один из этих параметров — или cols, или rows. Поэтому для того чтобы разделить окно и по вертикали, и по горизонтали, используются вложенные фреймы: вместо описания очередного фрейма <FRAME> помещается фреймовая структура <FRAMESET>.

Каждый фрейм имеет рамку стандартной толщины и цвета. Эту рамку посетитель страницы может перетаскивать с помощью мыши. Если содержимое фрейма не помещается в отведенной ему области окна, то у этой области появляется полоса прокрутки. Таковы свойства фрейма по умолчанию. Однако их можно изменить. С помощью параметров дескрипторов <FRAMESET> и <FRAME> можно изменить толщину и цвет рамки или вовсе ее убрать, а также запретить прокрутку содержимого фрейма и убрать с экрана соответствующую полосу прокрутки.

При ссылках на элементы фреймовой структуры широко используются имена фреймов, назначаемые с помощью параметра name. Затем эти имена присваиваются параметру target дескриптора гиперссылки **<A>.**

Для создания внутри окна области с собственной рамкой и самостоятельной прокруткой используется дескриптор <IFRAME>. Он имеет те же параметры, что и <FRAME>, но не нуждается в специальной фреймовой структуре, описываемой дескриптором <FRAMESET>, и может располагаться внутри обычного HTML-кода.

Тесты

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а) `<FRAMESET rows=70,*> ••• </FRAMESET>`

   б) `<FRAMESET rows=70%,*,70>... </FRAMESET>`

   в) `<FRAMESET cols=70,*%> ... </FRAMESET>`

   г) `<FRAMESET cols=70000,*> ... </FRAMESET>`

   д) `<FRAMESET rows=*,*> ... </FRAMESET>`

   е) `<FRAMESET rows=2*,*> ••• </FRAMESET>`

ж) `<FRAMESET rows=**,*> ... </FRAMESET>`

з) `<FRAMESET rows=*;*> ... </FRAMESET>`

и) `<FRAMESET rows=*> ... </FRAMESET>`

2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а)
```
<FRAMESET cols=*,*>
<FRAME src="source1.htm">
</FRAMESET>
```

б)
```
<FRAMESET cols=*,*>
<FRAME src="source1.htm">
<FRAME src="source2.htm">
</FRAMESET>
```

в)
```
<FRAMESET cols=*,* rows=*,*>
<FRAME src="source1.htm">
<FRAME src="source2.htm">
<FRAME src="source3.htm">
<FRAME src="source4.htm">
</FRAMESET>
```

г)
```
<FRAMESET cols=*,*,*,*>
<FRAME src="source1.htm">
<FRAME src="source2.htm">
<FRAME src="source3.htm">
<FRAME src="source4.htm">
</FRAMESET>
```

д)
```
<FRAMESET cols=*,*>
<FRAME src="source1.htm">
<FRAME src="source1.htm">
</FRAMESET>
```

3. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а)
```
<FRAMESET cols=*,* frameborder=no>
<FRAME src="source1.htm">
<FRAME src="source2.htm">
</FRAMESET>
```

б)
```
<FRAMESET cols=*,*>
<FRAME src="source1.htm" frameborder=no>
<FRAME src="source2.htm">
</FRAMESET>
```

в)
```
<FRAMESET cols=*,* border=no>
<FRAME src="source1.htm">
<FRAME src="source2.htm">
</FRAMESET>
```

г)
```
<FRAMESET cols=*,* border=10>
<FRAME src="source1.htm">
<FRAME src="source2.htm">
</FRAMESET>
```

д)
```
<FRAMESET cols=*,*>
<FRAME src="source1.htm" border=10>
<FRAME src="source2.htm">
</FRAMESET>
```

4. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а)
```
<FRAMESET cols=*,*>
<IFRAME src="source1.htm" border=10>
<IFRAME src="source2.htm">
</FRAMESET>
```

б)
```
<TABLE>
<TR><TD><IFRAME src="source1.htm" border=10></TD>
<TD><IFRAME src="source2.htm"></TD></TR>
</TABLE>
```

в)
```
<IFRAME> Содержимое встроенного фрейма</IFRAME>
```

Глава 12

Формы

В этой главе...

Основная схема формы

Web-сайт — это почти всегда диалог. Конечно, встречаются "односторонние" сайты, авторы которых стремятся только показать, но не услышать отзыв о показанном. Но даже там редко обходится без ссылки на автора: "Все, что вы думаете по этому поводу, пишите сюда".

Но чаще "сайтовладелец" желает получать о своих посетителях гораздо больше информации. Мы не будем говорить здесь о полулегальных способах отслеживать поведение посетителей без их ведома и согласия. Речь пойдет о способах получения информации от самих пользователей, — например, анкетных данных для вступления в виртуальный клуб или мнений по интересующему вас вопросу.

Как получить данные и передать их для обработки?

Для этой цели используются *формы* — элементы Web-страницы, содержащие такие знакомые нам поля ввода, списки, кнопки и прочие верные признаки графического интерфейса.

Подобно фреймам, таблицам и другим "крупногабаритным" элементам Web-страницы, форма — это блок HTML-кода, образованный специальными элементами HTML. Границами такого блока служат, как легко догадаться, дескрипторы <FORM>:

```
<FORM параметры>
...
</FORM>
```

Какие параметры имеет этот дескриптор?

Давайте задумаемся: какие вообще могут быть параметры у всех этих текстовых полей, списков, "точек", "галочек" и кнопочек, с которыми по нашей милости возится пользователь? Раз эти параметры выносятся "за скобки", в заглавный дескриптор блока, то они должны быть общими для всех описанных элементов интерфейса. Но ведь элементы эти такие разные, такие непохожие друг на друга! Единственное, пожалуй, что их объединяет, — это то, что все они предназначены для получения от пользователя какой-то информации.

Первое, что приходит в голову: эту информацию нужно как-то обрабатывать. Поэтому параметры дескриптора <FORM> должны описывать *то, что нужно делать дальше с полученными данными.* И это действительно так!

В самом деле: ввел, предположим, посетитель электронного магазина в одной строке адрес, куда доставить покупку, в другой — номер кредитной карточки, с которой перечислить деньги, выбрал из списка товар, щелкнул на кнопке "купить" и... что дальше? А дальше все это нужно передать на сервер и там как-то обработать.

Действие, которое следует выполнить с введенными в форму данными, определяется параметром action. Точнее, этот параметр определяет не столько само действие, сколько URL, по которому расположен обработчик данных. Чаще всего в качестве такого обработчика выступает специальная программа. Эта программа может быть расположена как на удаленном сервере, так и на компьютере посетителя страницы. Написана она может быть, в принципе, на любом языке. В частности, для простой обработки данных широко применяется язык сценариев JavaScript (см. главу 16). Впрочем, вместо адреса программы-обработчика значением параметра action может быть обычный адрес электронной почты. Тогда данные формы будут просто отправлены по этому адресу, а что с ними делать дальше — решит получатель.

Каким способом передаются данные? Уточним сразу: HTML не вмешивается в способы кодирования и передачи информации по сетям Internet. Однако существует два принципиально разных метода: первый заключается в передаче самих данных ("покупатель: Александр Петров; номер карточки: 123456789; покупка: костюм мужской и т.д."), второй — в передаче указателя на то место, где они находятся (например, URL). То, какой именно из них используется для данной формы, определяется параметром method. Передача данных соответствует значению post (в переводе с английского "переслать"), а передача ссылки — значению get ("получить"). По умолчанию используется последний вариант. Действительно, зачем лишний раз нагружать сеть? Лучше передать ссылку и этим ограничиться. Но передавать данные формы по электронной почте приходится полностью, а для этого используется значение post.

Вы, должно быть, слышали о существовании различных типов кодирования информации, передаваемой через Internet. Тип кодирования данных, введенных через форму, определяется параметром enctype (от английского *encryption type* — тип кодирования). В данном случае имеются в виду типы кодирования MIME. По умолчанию параметр enctype принимает значение application/x-www-form-urlencoded, но при отправке данных электронной почтой используется тип text/plain. Таким образом, для того чтобы данные формы передавались по электронной почте, код формы должен выглядеть так:

```
<FORM action=mailto:почтовый@адрес method=post enctype=text/plain>
...
</FORM>
```

Наконец, обратим внимание еще на одну деталь. Как правило, после заполнения формы содержащее ее окно закрывается, и на его месте появляется другое, — например, с сообщением о том, что данные успешно введены и получены. Как описать, в каком окне или фрейме будет открыта следующая страница? Правильно, с помощью уже знакомого нам параметра target (см. главу 11).

Из чего состоит форма

Уф... Кажется, с самой формой мы разобрались. Но, как учат философы, форма без содержания — ничто. Правда, другие философы с ними спорят и доказывают обратное... Но, если говорить не об абстрактной форме, а о формах HTML, то факт налицо: HTML-формы без содержания действительно не имеют смысла.

Что является содержанием HTML-формы? Пожалуй, ответить на этот вопрос вы могли бы и сами. Для этого достаточно вспомнить любую анкету или бланк, которую вам когда-либо приходилось заполнять. Давайте вспоминать вместе: что там было? Масса пустых строчек, куда нужно вписать фамилию, адрес, дату... Нужное подчеркнуть... Отметить "птичками" один или несколько из предлагаемых вариантов (а если анкета электронная, не забыть убрать выделенные варианты, предлагающие присылать вам по электронной почте всякую чепуху)... Да, и еще, конечно же, обычный текст — пояснения, что и куда нужно вписывать. Как правило, маловразумительные... Ну и всякие картинки, оживляющие пейзаж: логотипы, печати, рамки. Наконец, в электронных анкетах вместо барышни, которой нужно сдать бумажку, имеются кнопки: "Принять", "Отменить" и т.п.

Да, набралось порядочно. Давайте все это как-то классифицируем, чтобы не запутаться.

Первое, что бросается в глаза: все содержимое формы делится на две категории. В первую попадает все, что требует вмешательства пользователя: заполнить, подчеркнуть, отметить, нажать и т.п., во вторую — все остальное, что достаточно прочитать, просмотреть или можно вообще оставить без внимания. Назовем элементы первой категории — поля ввода, "птички", кнопки, списки — *активными,* а элементы второй категории — текст, картинки — *пассивными.*

Пассивные элементы формы нам хорошо знакомы. Они ничем не отличаются от любых других элементов Web-страницы. А вот к активным элементам стоит присмотреться внимательнее.

Например, какие общие свойства присущи таким элементам и, соответственно, какие общие параметры есть у описывающих их дескрипторов? Таких свойств два. Во-первых, это информация, которую вводит пользователь через данный элемент формы. Эта информация присваивается параметру value. Во-вторых, это уникальное в пределах формы имя, по которому данный элемент отличается от других. Это имя присваивается параметру name — аналогичный параметр нам уже встречался в фреймах (см. главу 11).

Текстовые строки

Строки для ввода текста на HTML-страницах встречаются сплошь и рядом. Тому, кто хоть раз пользовался поисковым Internet-сервером, не надо объяснять, что это такое: узкий вытянутый прямоугольник, внутри которого можно ввести с клавиатуры одну строку текста (рис. 12.1).

Рис. 12.1. Интерфейс поисковой системы — это тоже форма. Обычно она состоит из текстовой строки и кнопки ввода

Для ввода однострочных полей используется дескриптор <INPUT>. Это непарный дескриптор, обладающий целым "выводком" параметров, описывающих самые разнообразные свойства. В зависимости от значения параметра type этот дескриптор может "принимать вид" самых разных элементов формы. В частности, когда этот параметр имеет значение text, дескриптор <INPUT> "превращается" в текстовую строку.

Похоже, пора перейти от теории к практике. Создадим форму для ввода некоторой текстовой строки, воспользовавшись для этого уже известными нам сведениями о дескрипторах <FORM> и <INPUT>. Для простоты пока что будем предполагать, что вся вводимая информация затем пересылается по электронной почте (рис. 12.2).

Рис. 12.2. Простейшая форма, содержащая текстовую строку

Обратим внимание на несколько интересных моментов.

Прежде всего: что находится внутри формы? Кроме текстовой строки для ввода данных, — обычный форматированный текст, такой же, как везде. Текстовая строка имеет имя search и начальное значение 123456789123456789123456789, присвоенное параметру value. Как видим, этот параметр используется не только для передачи введенных данных, но и для вывода исходных значений.

Зачем мы выбрали такую последовательность цифр? Чтобы измерить длину строки. Как видим, в ней помещается 18 знаков. Как растянуть ее или, наоборот, сузить?

За это "отвечает" параметр size. Его значением является число символов, помещающихся в строке. Например, если нужно ввести что-то очень короткое — скажем, сегодняшнее число — то строку можно сократить до двух символов, присвоив параметру size значение 2. Но помешает ли это посетителю ввести более длинное число? Нет. Просто символы, введенные первыми, "уедут" из видимой части строки влево (рис. 12.3).

Для того чтобы ограничить также вводимую строку, используется параметр maxlength. Его значением служит количество символов, принимаемых формой и передаваемых для обработки. Если присвоить этому параметру значение 2, то пользователь просто не сможет ввести третий символ (рис. 12.4).

А теперь рассмотрим несколько специфических вариантов текстовых строк.

Довольно часто встречаются ситуации, когда вводимая строка не должна быть видна на экране. Типичный случай — пароль. Для того чтобы обеспечить конфиденциальность такого рода, используются текстовые строки специального вида — параметру type присваивается значение password (рис. 12.5).

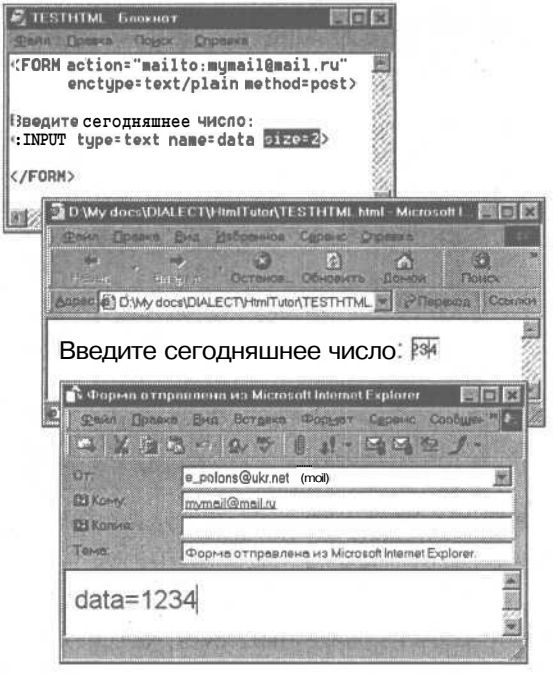


Рис. 12.3. Ограничение длины видимой текстовой строки с помощью параметра size не мешает пользователю ввести значение, превышающее ее длину

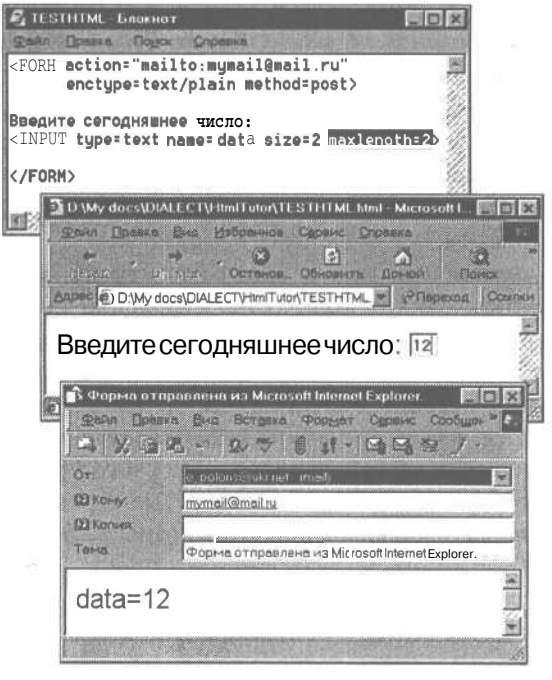


Рис. 12.4. Параметр `maxlength` ограничивает длину вводимой текстовой строки

Рис. 12.5. Ввод пароля: в окне броузера отображаются звездочки, но для обработки передается значение, введенное в действительности

Пожалуй, не менее часто возникает необходимость в передаче файла, имя которого вводится в HTML-форме. До сих пор в некоторых приложениях встречаются моменты, когда от пользователя требуется вводить имя вручную — полностью, вместе с длинным путем. И неоткуда скопировать... Если вы с такой проблемой сталкивались, то поймете меня, а если нет — поверьте на слово: это сущее наказание. Неудивительно, что пользователь всеми силами избегает попасть в такую ситуацию второй раз. Грамотные разработчики помещают рядом со строкой, куда нужно ввести имя файла, стандартную кнопку Обзор. Щелкнув на ней, пользователь открывает стандартное окно Windows, в котором и выбирает нужный файл. Для того чтобы такая кнопка появлялась рядом с текстовой строкой HTML-формы, параметру type присваивается значение file (рис. 12.6).

Иногда возникает необходимость передать из формы программе-обработчику некую скрытую информацию — такую, которую пользователь не то что не вводит, но даже и не видит. Для этого используется дескриптор <INPUT> с параметром type=hidden:

```
<INPUT type=hidden name=secret value="В этом письме содержится анкета">
```

Рис. 12.6. Для ввода имени файла используется текстовая строка с параметром `file`

Кнопки

Кнопки — главный элемент любой электронной формы. Не согласны? Что ж, возможно, вы правы. Но как сообщить о том, что ввод данных завершен, если форма лишена кнопки "Отправить"?

До сих пор мы действительно обходились без кнопок: пока дело касалось лишь текстовых строк, для ввода данных достаточно нажать клавишу <Enter>. Однако с другими элементами форм такой "фокус" не удастся. Поэтому, прежде чем обогатить нашу форму этими элементами, мы снабдим ее средствами управления в виде кнопок.

Как это ни странно, кнопки создаются с помощью того же дескриптора `<INPUT>`, что и текстовые строки. Однако значение параметра type в этом случае другое — в зависимости от назначения кнопки.

Чаще всего — практически всегда — в формах встречается кнопка для передачи данных программе-обработчику. Надписи на ней бывают разные — "Принять", "Отправить", "OK", "Поехали!". В сущности, это не важно. То, что происходит при щелчке определяется не надписью.

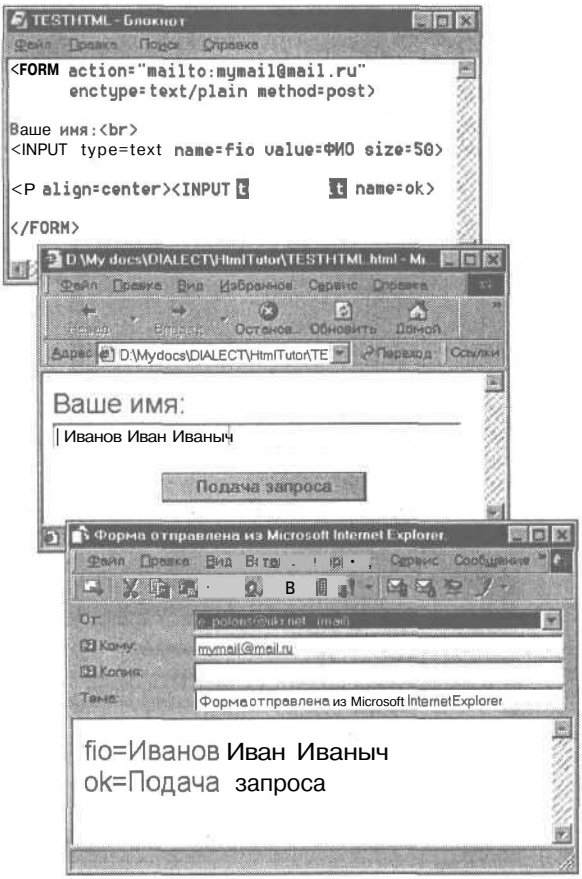


Рис. 12.7. Для того чтобы создать кнопку отправки данных, параметру type присваивается значение `submit`

Для того чтобы создать такую кнопку, параметру type присваивается значение submit. Какую роль здесь играют другие параметры, например name и value? Посмотрим. Создадим форму, показанную на рис. 12.7, и щелкнем на кнопке. Как видим, если опустить параметр value, то по умолчанию на кнопке появляется надпись "Подача запроса". И она же присваивается имени ok при передаче результатов. Если вдуматься, все должно быть немного иначе. Во-первых, надпись на кнопке могла бы быть более информативной и дружелюбной. Мне, например, нравится "Отправить". Давайте сразу присвоим это значение параметру value. Во-вторых, зачем передавать программе-обработчику надпись на кнопке? Нонсенс. Убираем параметр name и получаем как раз то, что нужно (рис. 12.8).

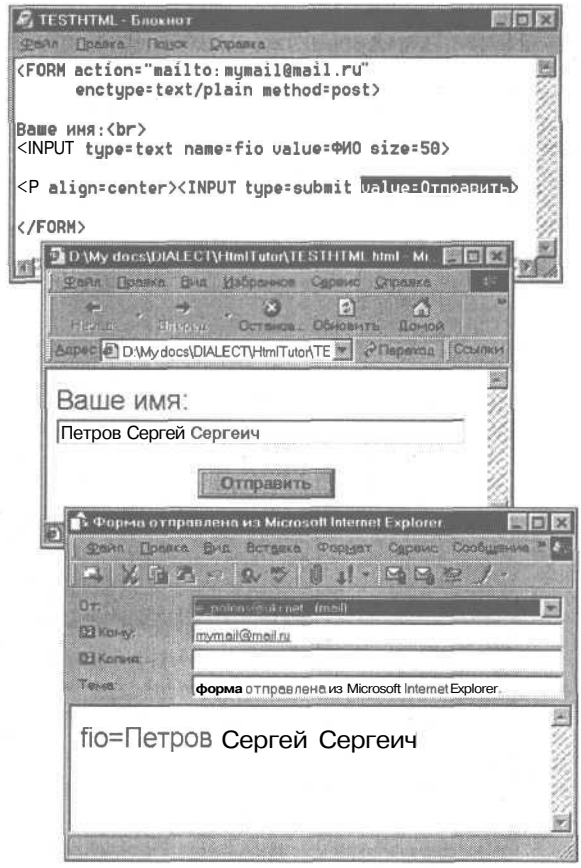


Рис. 12.8. Параметр value определяет надпись на кнопке, *а* параметр name вообще не обязателен

Если стандартный серый квадрат с надписью не "вписывается" в дизайн страницы, его можно заменить на любой рисунок. Для этого параметру type вместо submit присваивается значение image. После этого кнопка превратится в рисунок, а дескриптору `<INPUT>` можно присваивать все те параметры, что и обычным изображениям (см. главу 8). Кроме того, при этом программе-обработчику передаются дополнительные данные — координаты точки, в которой посетитель щелкнул на кнопке-картинке (рис. 12.9). Отсчет ведется в пикселях, от верхнего левого угла изображения.

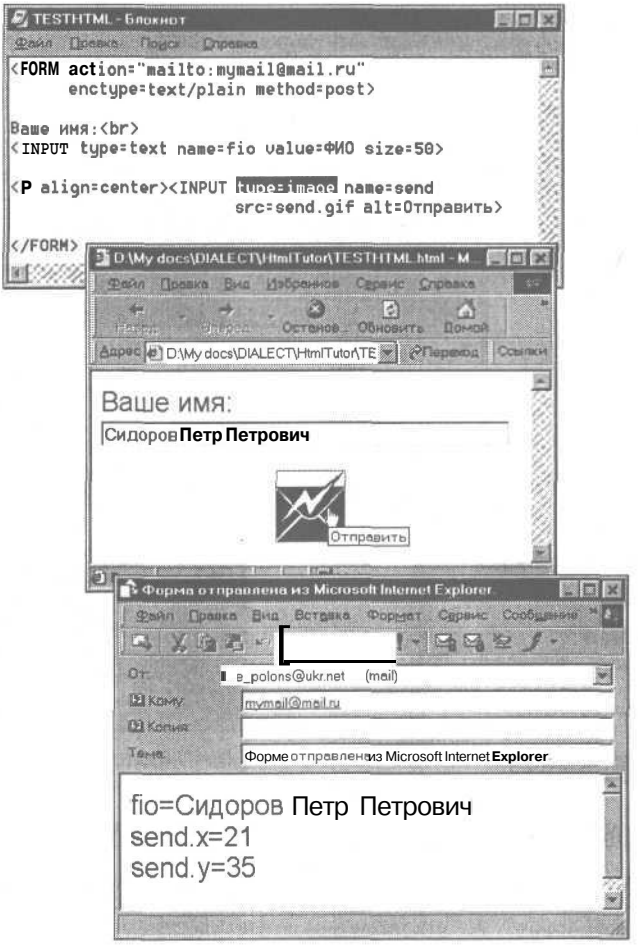


Рис. 12.9. Кнопка, превращенная в картинку, сохраняет все параметры изображения

Следующая по распространенности функция кнопки в электронной форме — вернуть все, как было. Конечно, если форма состоит из единственной строчки, посетитель вполне справится и сам. Но представьте себе длиннющую анкету, которую нужно целиком "обнулить". Обрадовала бы вас перспектива почистить все поля с помощью мыши и клавиатуры? По-моему, вопрос излишен.

Для создания кнопки возврата параметру type присваивается значение reset. В результате получаем точно такую же кнопку, что и в случае submit, только результат ее действия другой, вместо того чтобы отправить данные на обработку, данные просто удаляются, и форма приводится к исходному состоянию (рис. 12.10).

Обратите внимание: можно заменить рисунком кнопку отправки данных, но не кнопку отмены. Увы...

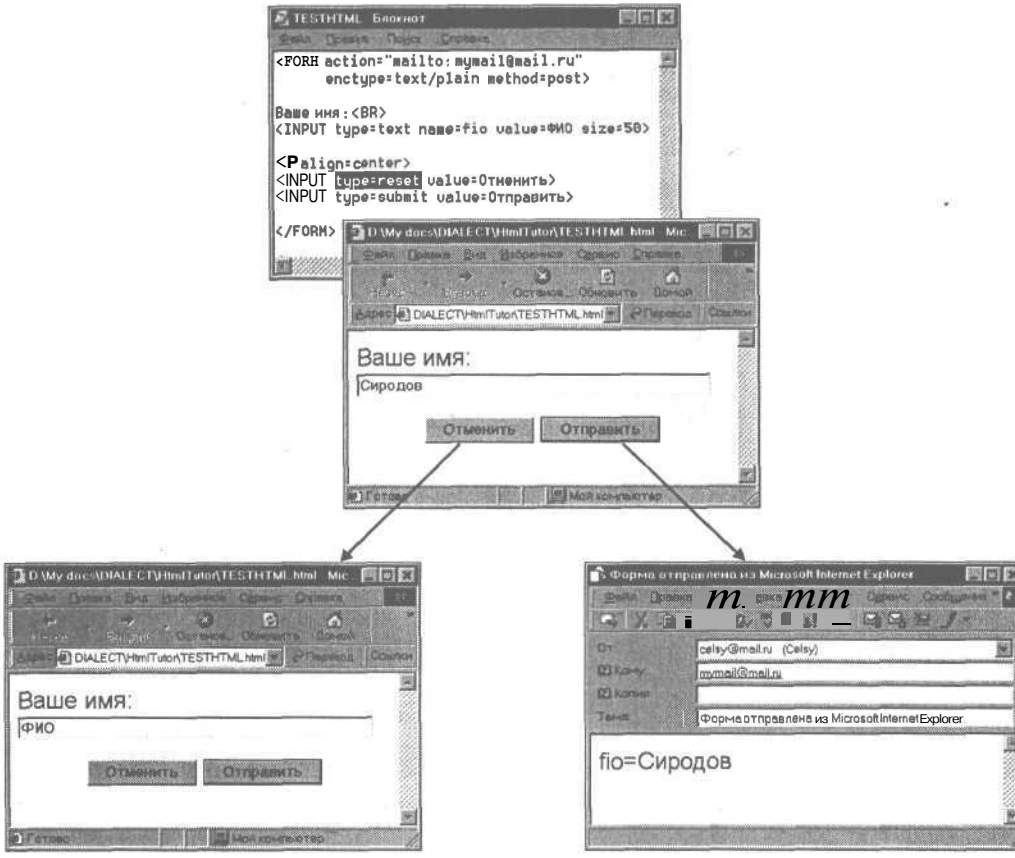


Рис. 12.10. С помощью значения reset создается кнопка, возвращающая форму в исходное состояние

На рис. 12.10 показан типичный недочет: надписи на кнопках, выполняющих противоположное действие, похожи. Что стоит посетителю, который торопится (а они вечно торопятся, ведь у провайдера счетчик минут тикает!) и поэтому читает "по диагонали", ошибиться и щелкнуть не на той кнопке? И что он затем скажет в ваш адрес?.. В этом смысле привычные, хоть и не всем понятные OK и Cancel куда лучше незнакомых надписей, заставляющих медлить и разбираться, что к чему.

Если необходимо создать кнопку, назначение которой отличается от приема данных и сброса, используется значение type=button. Действие, выполняемое при щелчке на кнопке, определяется параметром onclick (см. главу 16).

Текстовые поля

Не всегда текст, который нужно ввести, помещается в одной строке. Бывает, что он растягивается на несколько строк или даже абзацев. Конечно, можно обойтись текстовой строкой "бесконечной" длины (без указания значения параметра maxlength). Однако выглядит такая строка — без начала, без конца — неизящно, а пользоваться ею очень неудобно.

Поэтому для ввода крупных блоков текста предусмотрен другой элемент формы — *поле ввода*. Типичный пример такого поля — а заодно и нескольких видов текстовых строк — можно встретить на любом сайте, предоставляющем бесплатные услуги электронной почты (рис. 12.11).

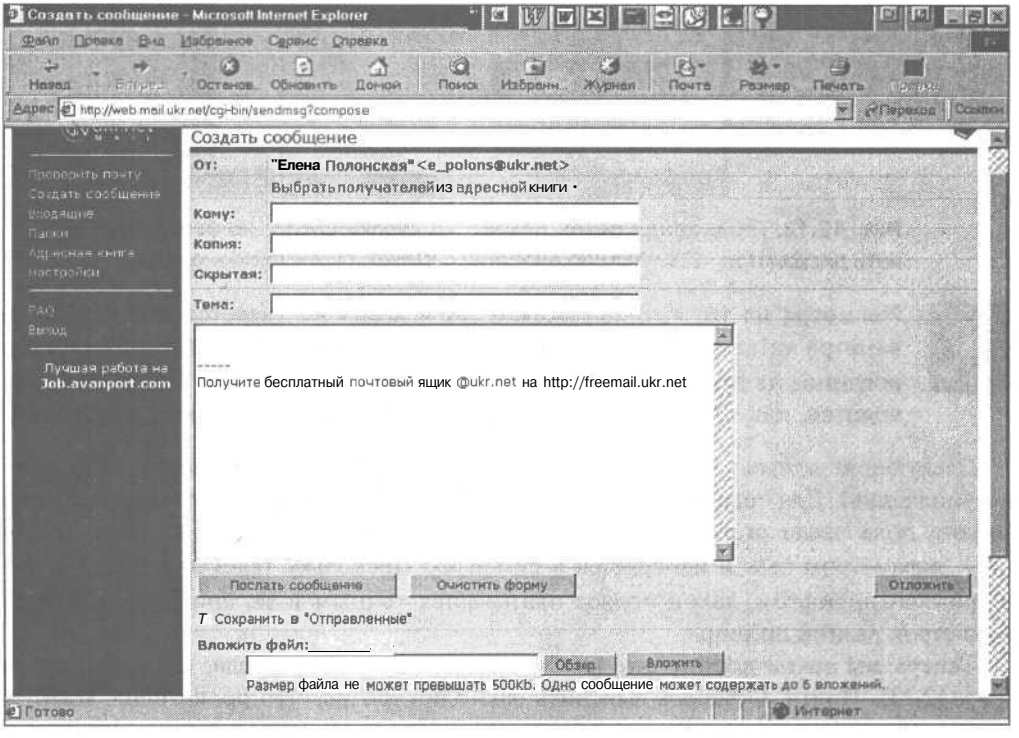


Рис. 12.11. Типичный пример поля ввода — форма электронного письма

Для создания текстового поля используется дескриптор <TEXTAREA>. Что получится, если просто использовать его интуитивно, применяя параметры name и value по аналогии с текстовыми строками? Явная ошибка (рис. 12.12).

Прежде всего: почему в поле ввода оказался весь остальной код? Потому что дескриптор <TEXTAREA> — парный, а мы его не закрыли. Внутри него помещается текст, который должен оказаться в поле ввода по умолчанию. Это логичнее, чем делать его значением параметра value: ведь текст может быть довольно длинным.

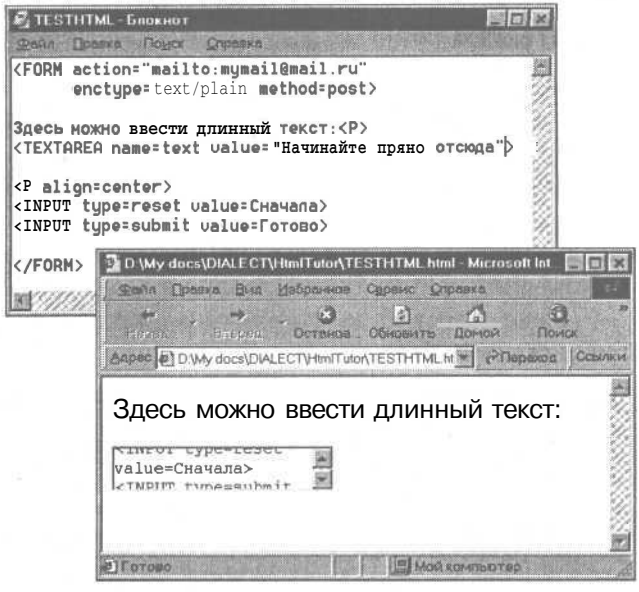


Рис. 12.12. Поле ввода очень похоже на строку ввода, но использовать дескриптор <TEXTAREA> по аналогии с <INPUT type=text> — ошибка

Несмотря на то, что содержимое поля ввода не является значением параметра value и, теоретически, может содержать дескрипторы форматирования, на практике оно форматированию не подлежит. Весь текст выводится, как правило, моноширинным шрифтом ("пишущая машинка").

Следующий вопрос: почему поле ввода такое маленькое? Очевидно, оно такое по умолчанию. Для того чтобы его "раздвинуть", нужно описать его размеры явно. Высота поля ввода определяется параметром rows и измеряется в строках, ширина — параметром cols и измеряется в символах. Поскольку текст выводится моноширинным шрифтом, такой подход оказывается точным и не приводит к появлению строк разной ширины.

Теперь мы знаем достаточно, чтобы создать такое поле ввода, как нам хочется (рис. 12.13). Почти. Осталось выяснить еще один нюанс. Что происходит, если содержимое не помещается в окне? Если строк слишком много, полоса прокрутки справа от них "оживает" и начинает оправдывать свое существование. А если строка слишком длинная? Разумеется, когда она приближается к правой границе окна, очередное слово появляется "этажом ниже". Но в каком виде такая "разорванная" строка передается на обработку? Отдельными кусками или целиком?

Оказывается, можно и так, и эдак. Способ представления текста, вводимого в окно, определяется параметром wrap. По умолчанию все происходит именно так, как ожидается: в окне текст автоматически разбивается на строки, но при передаче эта автоматическая разбивка не сохраняется: если вы ввели все одной строкой, то оно так и будет передано. Этот режим соответствует значению virtual. Если мы хотим, чтобы переход на новую строку в окне происходил только когда

пользователь нажимает <Enter>, мы должны присвоить параметру wrap значение off. Наконец, если мы хотим, чтобы переход на новую строку происходил автоматически и эта разбивка сохранялась при передаче текста на обработку, нужно использовать значение hard (рис. 12.14).

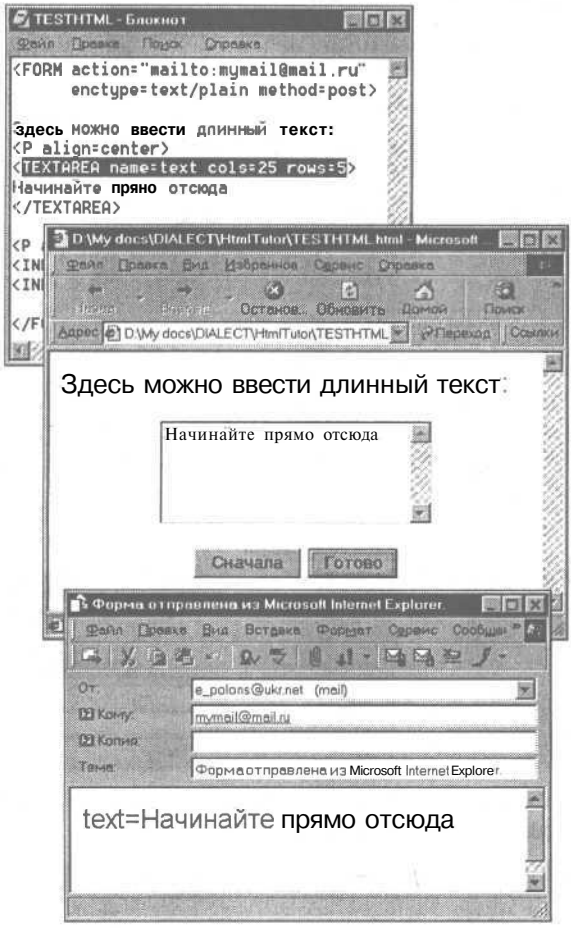
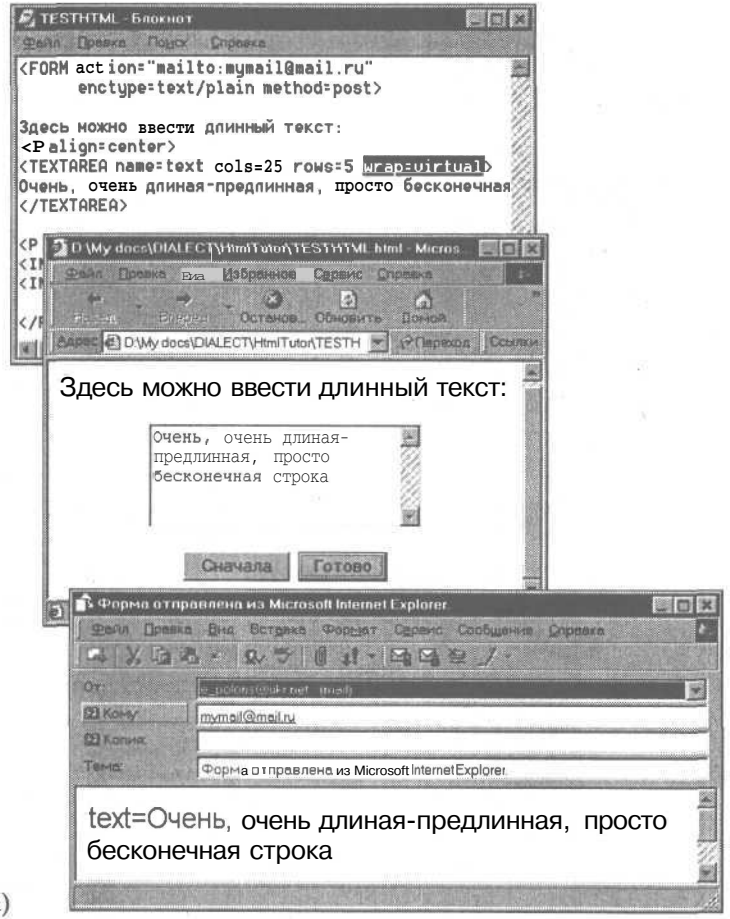
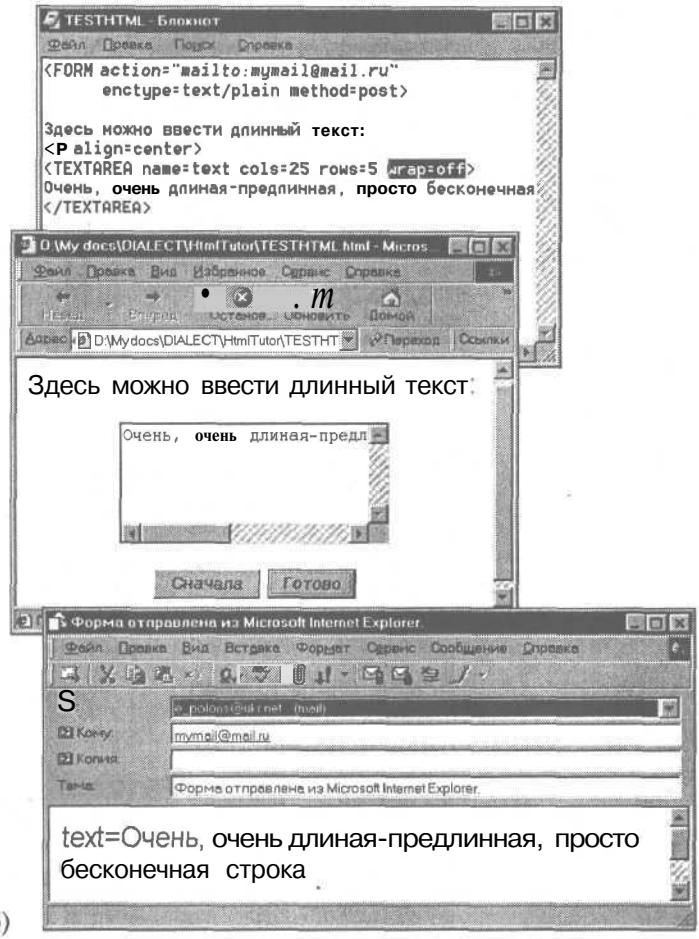


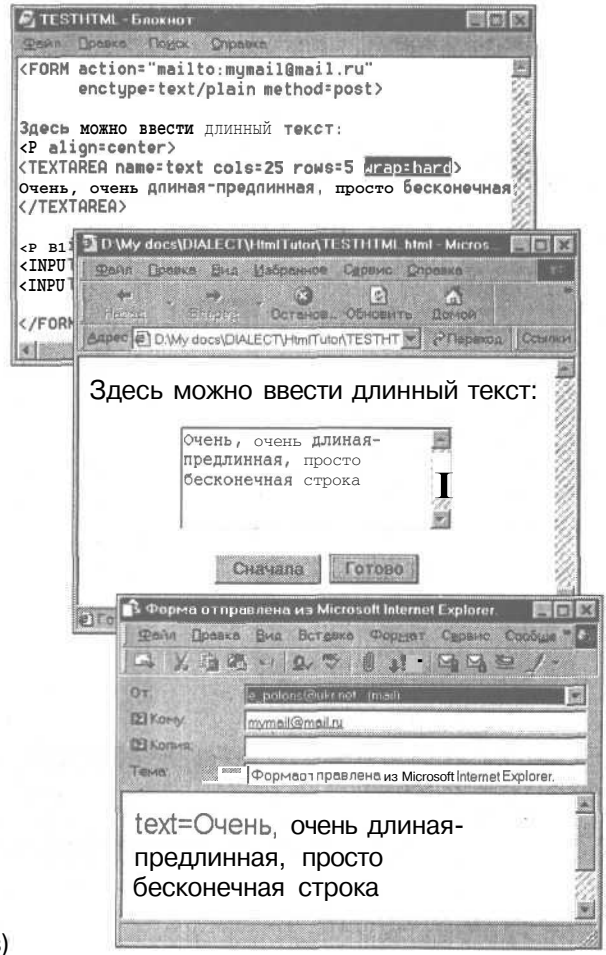
Рис. 12.13. Так работает "правильное" поле ввода


TESTHTML - Блокнот
Файл Правка Поиск Справка
<FORM action="mailto:mymail@mail.ru"
enctype=text/plain method=post>
Здесь можно ввести длинный текст:
<P align=center>
<TEXTAREA name=text cols=25 rows=5 wrap=virtual>
Очень, очень длинная-предлинная, просто бесконечная
</TEXTAREA>
Здесь можно ввести длинный текст:
Очень, очень длинная-
предлинная, просто
бесконечная строка
Сначала
Готово
Форма отправлена из Microsoft Internet Explorer
Файл Правка Вид Вставка Формат Сервис Сообщение Справка
Кому:
mymail@mail.ru
Копия:
Тема:
Форма отправлена из Microsoft Internet Explorer
text=Очень, очень длинная-предлинная, просто
бесконечная строка

а)


TESTHTML - Блокнот
Файл Правка Поиск Справка
<FORM action="mailto:mymail@mail.ru"
enctype=text/plain method=post>
Здесь можно ввести длинный текст:
<P align=center>
<TEXTAREA name=text cols=25 rows=5 wrap=off>
Очень, очень длиная-предлинная, просто бесконечная
</TEXTAREA>
D:\My docs\DIALECT\HtmlTutor\TESTHTML.html - Micros
Файл Правка Вид Избранное Сервис Справка
Назад Вперед Остановить Обновить Домой
Адрес D:\Mydocs\DIALECT\HtmlTutor\TESTHT
Переход Ссылки
Здесь можно ввести длинный текст:
Очень, очень длиная-предл
Сначала
Готово
Форма отправлена из Microsoft Internet Explorer
Файл Правка Вид Вставка Формат Сервис Сообщение Справка
Кому:
mymail@mail.ru
Копия:
Тема:
Форма отправлена из Microsoft Internet Explorer.
text=Очень, очень длиная-предлинная, просто бесконечная строка

б)



в)

Рис. 12.14. Три режима отображения и передачи длинных строк, определяемые параметром wrap: перенос при вводе (а), отсутствие переноса (б), и перенос при вводе и передаче (в)

Списки вариантов

В электронных формах существует два типа списков, из которых посетителю страницы предлагается что-то выбрать. В первом случае допускается выбор нескольких вариантов, во втором — только одного. Мы будем называть список первого типа *списком вариантов*, а список второго типа — *списком-переключателем.*

Обычно пункты списков вариантов снабжены квадратными "окошками", в которых при выборе появляются "птички" (checkbox). Для создания такого списка используется уже знакомый нам дескриптор <INPUT> с параметром type=checkbox. Как это выглядит? Очевидно, таких дескрипторов должно быть столько же, сколько вариантов в списке. В остальном — кажется, мы уже знаем, как пользоваться параметрами name и value... Однако, видимо, в данном случае ими нужно пользоваться как-то иначе (рис. 12.15).

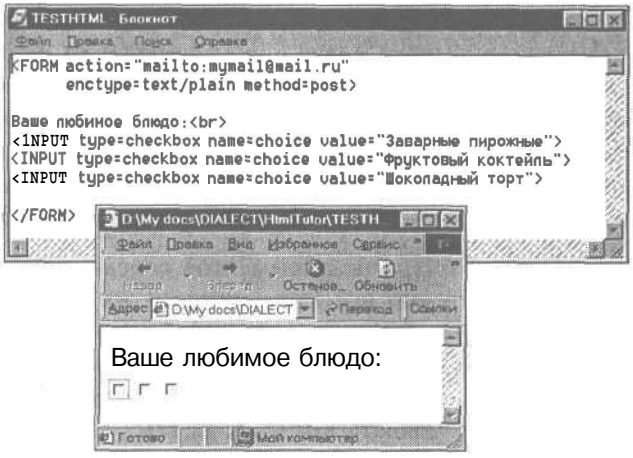


Рис. 12.15. Попытка создать список вариантов: использование знакомых параметров не приводит к желаемому результату

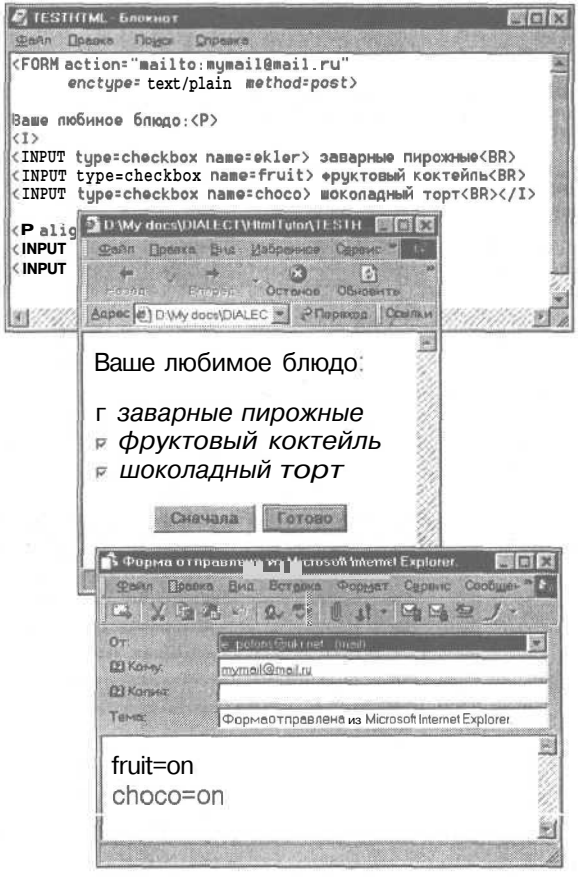


Рис. 12.16. Код списка вариантов состоит из самостоятельных дескрипторов <INPUT>; в качестве пояснений используется обычный текст

Для создания списка вариантов в самом деле нужно столько дескрипторов <INPUT>, сколько в нем есть вариантов. Однако имена у них должны быть разные — у каждого варианта свое. Зачем это нужно? Дело в том, что, хотя логически список является цельным элементом формы, с точки зрения кода это всего лишь набор разрозненных, несвязанных дескрипторов. У каждого — свое имя и значение. А для того чтобы стало ясно, что именно нам предлагают выбрать, нужно снабдить эти квадратики пояснениями. Обычными текстовыми — параметр value здесь явно не годится (рис. 12.16).

Обратите внимание: имена всех элементов списка — разные. Что будет, если сделать одинаковыми все или некоторые из них? Если вы думаете, что завизжит сирена и броузер выдаст сообщение об ошибке, то ошибаетесь. Броузер молча "проглотит" вашу оплошность и впоследствии будет вести себя так, как будто ничего не случилось. Но случилось ли что-то на самом деле? Решать вам (рис. 12.17).

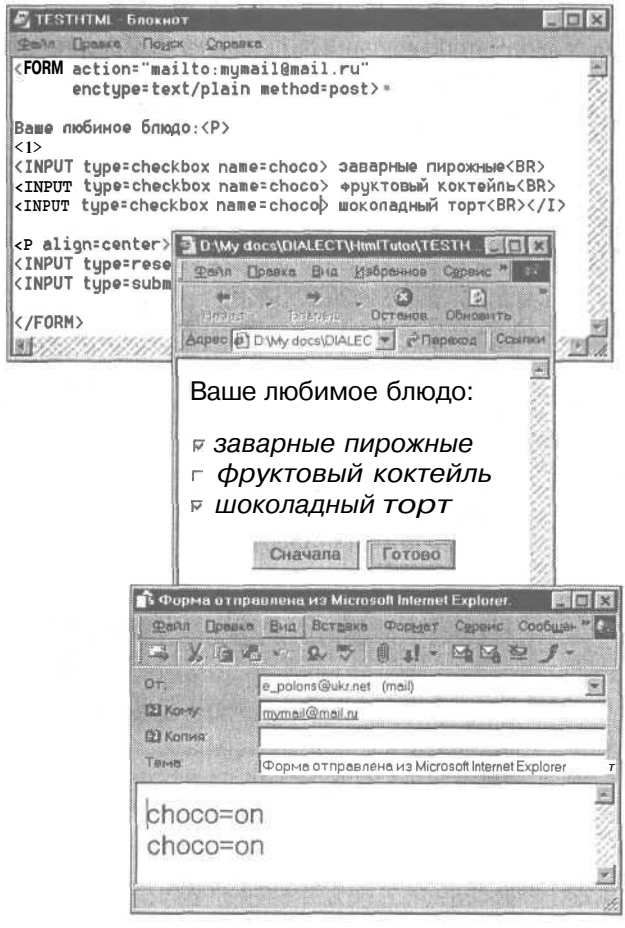


Рис. 12.17. Вот что получается, если разные варианты списка имеют одинаковые значения параметра name

Отметим еще одну важную особенность. Как передаются данные о выбранных вариантах? Очень просто. Если соответствующий пункт списка выбран, его имя включается в состав передаваемых данных со значением on (см. рис. 12.16). А как поступить, чтобы один из пунктов был выбран по умолчанию? Естественно предположить, что для этого нужно включить в соответствующий дескриптор <INPUT> параметр value со значением on. Однако, если вы проверите эту версию на практике, то убедитесь, что она не работает: каким бы ни было значение параметра value, в случае списка вариантов оно игнорируется. Если мы хотим, чтобы какой-то из пунктов был выбран по умолчанию, то нам понадобится дополнительный параметр — checked. Значений он не имеет — просто включите его в состав соответствующего дескриптора (рис. 12.18).

Рис. 12.18. Для того чтобы один из пунктов списка вариантов был выбран по умолчанию, используется не имеющий значений параметр checked

Заполняя различные Internet-анкеты, вы могли обратить внимание на то, что такие списки, "заполненные" по умолчанию, встречаются достаточно часто — чаще, чем хотелось бы. Обычно в конце длинной-предлинной

формы мелким шрифтом стоит что-то вроде "Я хочу регулярно получать по почте рекламу продукции", а рядом — незаметная "птичка". И кто обвинит владельца сайта в распространении спама? Ведь вы сами его "попросили", не дочитав до конца и поторопившись щелкнуть на кнопке...

Списки-переключатели

В списках-переключателях слева от пунктов имеются кружки, причем в центре кружка, соответствующего выбранному пункту, появляется жирная точка (radiobutton). Для создания таких списков используется тот же дескриптор <INPUT>, что и для списка вариантов, но параметру type присваивается значение radio. Как и в списке вариантов, каждому элементу списка-переключателя соответствует отдельный дескриптор <INPUT>. Однако, в отличие от списка вариантов, все элементы списка-переключателя имеют одно имя. Иначе и не может быть: ведь мы хотим получить только одно значение из всего перечня. Наконец, здесь, как и в списке вариантов, для выбора одного из пунктов по умолчанию используется параметр checked (рис. 12.19).

Рис. 12.19. Список-переключатель формируется из дескрипторов <INPUT> с параметром type=radio

Раскрывающиеся списки

Списки часто бывают очень длинными и скучными. И занимают много места. Если форма бумажная — ничего не поделаешь. Тяжко вздыхаем и заправляем в принтер еще один лист. Но если форма электронная и мы не хотим зря занимать место, можно воспользоваться раскрывающимся списком. Что это такое, знает любой, кто имел дело с Windows дольше получаса: строка, в которой что-то написано, а справа — небольшая кнопка со стрелкой. Если щелкнуть на стрелке, вниз "выпадает" список. Щелкаем на одном из его пунктов — и список сворачивается обратно, а в строке появляется выбранный пункт.

Как сделать подобный список на HTML-странице? Сам список создается с помощью дескриптора <SELECT>, а отдельные элементы — с помощью дескрипторов <OPTION>. Вся конструкция имеет вид, показанный на рис. 12.20.

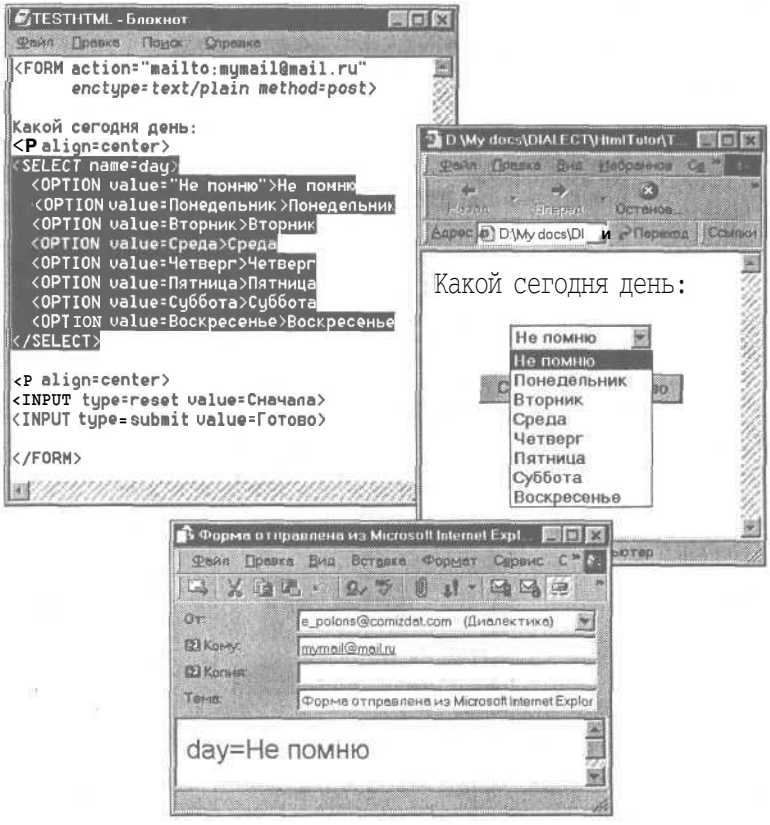


Рис. 12.20. Раскрывающийся список создан с помощью дескрипторов <SELECT> и <OPTION>

Вообще-то, дескриптор <OPTION> — парный. Но закрывать его не обязательно, так же как уже знакомые нам дескрипторы **<P>**, <LI> и др.

Как видим, система "<SELECT> — <OPTION>" работает просто: программе-обработчику передается имя дескриптора <SELECT> и значение параметра value того из дескрипторов <OPTION>, который соответствует выбранному пункту списка.

Если мы хотим, чтобы по умолчанию был выбран определенный пункт списка, мы должны либо расположить его дескриптор <OPTION> первым, либо использовать уже знакомый нам параметр checked или selected.

Однако на этом возможности раскрывающегося списка не исчерпываются. По умолчанию список представляет собой одну строку, которая "раскрывается" при щелчке на кнопке. Но мы можем модифицировать список так, чтобы он принял вид окна, содержащего несколько строк и — при необходимости — полосу прокрутки. Для этого нам потребуется параметр size дескриптора <SELECT>. Этот параметр определяет количество строк, из которых состоит список в "закрытом" состоянии, и по умолчанию равен 1. Что получится, если сделать его равным, например 3? Окно с полосой прокрутки, в котором можно выбрать нужный день (рис. 12.21).

Рис. 12.21. Параметр size определяет количество одновременно видимых строк списка

Ну как, нравится? По-моему, скорее необычно, чем красиво или полезно. Но не будем торопиться с вердиктами. В таком виде, возможно, и впрямь не очень практично. Но ведь иногда из списка требуется выбрать не одно, а сразу несколько вариантов. И тогда возможность видеть их все сразу может очень пригодиться. Для того чтобы список позволял выбрать сразу несколько вариантов, ис-

пользуется параметр multiple. Как и checked, этот параметр не имеет значений. Просто проставляем его — и все (рис. 12.22). Обратите внимание: в результате программе-обработчику передается несколько значений с одинаковым именем.

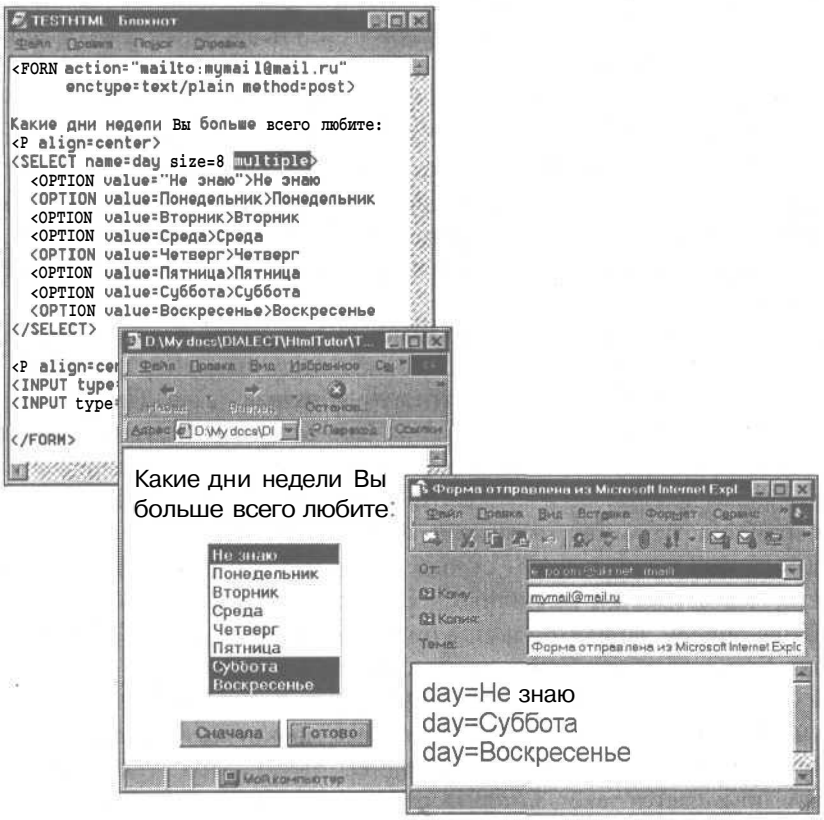


Рис. 12.22. Благодаря параметру multiple из списка можно выбрать сразу несколько значений

Для того чтобы выбрать несколько пунктов из такого списка, используется стандартный подход Windows: если эти пункты следуют подряд, выбираем первый, нажимаем клавишу <Shift> и, удерживая ее, выбираем последний пункт. Если же пункты разбросаны, выбираем их в произвольном порядке, удерживая нажатой клавишу <Ctrl>.

При разработке электронной формы рекомендуется придерживаться следующего правила: то, что посетитель страницы видит на экране одновременно, должно представлять собой законченный блок информации. По возможности, разумеется: если вам это не удастся, никто не подаст на вас в суд. Однако мало кому понравится пользоваться прокруткой только для того, чтобы добраться до кнопки отправки. Тому, чтобы форма выглядела эстетично и чтобы ею было удобно пользоваться, очень способствует рациональная комбинация различных списков.

Резюме

Электронная форма — эффективное средство, благодаря которому HTML-страница превращается из "пассивной", лишь предоставляющей информацию пользователю, в "активную", позволяющую принять информацию от пользователя и передать ее для обработки.

Способ обработки и передачи данных определяется дескриптором <FORM>, внутри которого и заключается код формы. Средство обработки определяется параметром action, метод передачи — параметром method, а тип кодирования — параметром enctype.

Внутреннее содержание формы можно разделить на две части: активное и пассивное. Пассивными элементами формы являются все комментирующие и декоративные элементы, которые могут там содержаться. Это обычные составляющие HTML-страницы.

Активные элементы формы предназначены для ввода данных. Это строки и поля ввода, списки и кнопки. У каждого активного элемента формы — как и у всей формы — есть два основных параметра — name и value. Первый определяет имя элемента, по которому его можно отличить от других элементов формы, второй — значение, которое передается через этот элемент. Большинство активных элементов формы описывается дескриптором <INPUT>, а их вид определяется значением параметра type. Так, значение text соответствует строке ввода, file — строке выбора файла, password — строке ввода пароля; значения submit, reset и button определяют кнопки различных видов, а значения checkbox и radio — два типа списков: список вариантов и список-переключатель, соответственно.

Еще два вида элементов ввода, используемых в формах, создаются с помощью следующих дескрипторов. Дескриптор <TEXTAREA> позволяет создавать поля ввода — прямоугольные окна с собственными средствами прокрутки, в которые можно вводить произвольный текст (без форматирования). Дескриптор <TEXTAREA> — парный. Внутри него помещается текст, который содержится в поле ввода по умолчанию.

Наконец, еще один распространенный элемент электронных форм — раскрывающийся список — создается с помощью конструкции HTML, образуемой дескрипторами <SELECT> и <OPTION>. Первый является парным и заключает в себе весь список, вторые предназначены для создания отдельных пунктов. Списки, созданные таким образом, могут состоять из любого количества строк (если список состоит из одной строки, то он является "раскрывающимся"), а также, в зависимости от параметра multiple, позволяют выбрать один или несколько элементов.

Данные, вводимые посредством формы, обрабатываются не средствами HTML. Они могут передаваться по электронной почте или непосредственно программе-обработчику. Язык, на котором может быть написана такая программа, значения не имеет. В частности, для обработки таких данных могут использоваться сценарии на языке JavaScript.

Тест

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а)
   ```
   <FORM action=mailto:mymail@mail.dom method=post enctype=application/
   x-www-form-urlencoded>
   ...
   </FORM>
   ```

   б)
   ```
   <FORM action=mailto:mymail@mail.dom method=get enctype=text/plain>
   ...
   </FORM>
   ```

   в)
   ```
   <FORM action=mailto:mymail@mail.dom method=post enctype=text/plain>
   ...
   </FORM>
   ```

   г)
   ```
   <FORM action=mailto:mymail@mail.dom method=post enctype=plain>
   ...
   </FORM>
   ```

   д)
   ```
   <FORM action=mailto:mymail@mail.dom method=post enctype=text>
   ...
   </FORM>
   ```

2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а) `<INPUT type=text value="Введите текст:">`

   б) `<INPUT type=text name=itext value="Введите текст:">`

   в) `<INPUT type=text name=itext> Введите текст </INPUT>`

   г) `Введите текст: <INPUT type=text name=itext>`

3. Сколько символов можно ввести в следующей строке и какова видимая длина этой строки?

   ```
   <INPUT type=text name=itext size=10 maxlength=20>
   ```

   а) 10 и 20, соответственно

   б) 20 и 10, соответственно

4. Какой или какие из следующих фрагментов HTML-кода кнопки отправки данных содержат ошибки?

   а) `<INPUT type=button value=submit>`

   б) `<INPUT type=submit value=OK>`

   в) `<INPUT type=submit name=OK>OK</INPUT>`

   г) `<INPUT type=submit name=OK>`

   д) `<INPUT type=image src=ok.gif value=OK>`

5. Какой или какие из следующих фрагментов HTML-кода кода обнуления данных формы содержат ошибки?

   а) `<INPUT type=button value=reset>`

   б) `<INPUT type=reset value=Cancel>`

в) <INPUT type=reset name=Cancel>Cancel</INPUT>

г) <INPUT type=reset name=Cancel>

д) <INPUT type=image src=cancel.gif value=Cancel>

6. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <TEXTAREA>Введите длинный текст</TEXTAREA>

б) <TEXTAREA name=ltext>Введите длинный текст</TEXTAREA>

в) <TEXTAREA name=ltext value=Введите длинный текст >Введите длинный текст</TEXTAREA>

г) <TEXTAREA name=ltext value=Введите длинный текст></TEXTAREA>

д) <TEXTAREA name=ltext value=Введите длинный текст>

7. Какой или какие из следующих списков позволяют выбрать несколько вариантов?

а)
```
<INPUT type=checkbox name=v1>Вариант 1
<INPUT type=checkbox name=v2>Вариант 2
<INPUT type=checkbox name=v3>Вариант 3
```

б)
```
<INPUT type=checkbox name=v1 checked>Вариант 1
<INPUT type=checkbox name=v2 checked>Вариант 2
<INPUT type=checkbox name=v3 checked>Вариант 3
```

в)
```
<INPUT type=radio name=v1 checked>Вариант 1
<INPUT type=radio name=v1 checked>Вариант 2
<INPUT type=radio name=v1 checked>Вариант 3
```

г)
```
<INPUT type=radio name=v1>Вариант 1
<INPUT type=radio name=v1>Вариант 2
<INPUT type=radio name=v1>Вариант 3
```

д)
```
<SELECT name=v1>
<OPTION value="Вариант 1">
<OPTION value="Вариант 2">
<OPTION value="Вариант 3">
</SELECT>
```

е)
```
<SELECT name=v1>
<OPTION value="Вариант 1" checked>
<OPTION value="Вариант 2" checked>
<OPTION value="Вариант 3" checked>
</SELECT>
```

ж)
```
<SELECT name=v1 checked>
<OPTION value="Вариант 1">
<OPTION value="Вариант 2">
<OPTION value="Вариант 3">
</SELECT>
```

з)
```
<SELECT name=v1 multiple>
<OPTION value="Вариант 1">
<OPTION value="Вариант 2">
<OPTION value="Вариант 3">
</SELECT>
```

Глава 13

Структура HTML-документа

В этой главе...

- Основные параметры Web-страницы
- Отступы от края окна
- Прокрутка
- Цвет фона
- Цвет текста

Основные параметры Web-страницы

Как мы уже знаем, броузер быстро распознает содержимое файла как HTML-код, если расширение этого файла — htm или html (см. *Введение)*. Однако, вообще говоря, документ HTML имеет более четкую структуру.

Начало и конец HTML-кода обозначаются дескрипторами <HTML> и </HTML>:

```
<HTML>
Содержимое Web-страницы
</HTML>
```

Впрочем, практически все современные броузеры сами "понимают", где начинается и, где заканчивается код Web-страницы. Тем не менее правильно структурировать ее не только желательно, но зачастую и необходимо.

Хорошая Web-страница делится на две части: *тело* — то, что выводится в окне броузера, и *заголовочную часть* — раздел, в котором содержится справочная информация, используемая броузером и различными службами Internet, такими как поисковые серверы. Для того чтобы отделить тело от заголовка, используются дескрипторы <HEAD> и <BODY>:

```
<HTML>
<HEAD>
<!-- Заголовок Web-страницы -->
</HEAD>
<BODY>
<!-- Тело Web-страницы -->
</BODY>
</HTML>
```

Вообще-то, без дескриптора <HTML> можно было бы обойтись. Броузеры последних поколений действуют в соответствии с известной юридической формулой — *все, что не запрещено, разрешено* — и добросовестно выводят в окно все, что не "забаррикадировано" специальными дескрипторами.

Дескриптор <BODY> не только структурирует HTML-документ, но и позволяет определить его основные свойства — такие, которые распространяются на всю страницу. Если эти параметры не указывать, то соответствующие свойства страницы — отступы, цвета и др. — будут такими, какие заданы по умолчанию. А теперь рассмотрим эти параметры подробнее.

HTML-страницы, содержащие фреймовую структуру, не нуждаются в дескрипторах <HTML> и <BODY>. Подробнее об этом читайте в главе 11.

Отступы от края окна

Для Web-страницы поля, или отступы — то же, что и поля для обычной, печатной страницы. Расстояния от текста до краев листа... простите, окна броузера. Только измеряются они не в миллиметрах, а в пикселях или в процентах.

Горизонтальные отступы задаются с помощью трех параметров —leftmargin, rightmargin и marginwidth. Те, кто учил в школе английский, уже догадались - первый определяет ширину левого отступа, второй — правого и третий — обоих горизонтальных отступов. Выходит, если поля разные, пользуемся параметрами rightmargin и marginwidth, а если одинаковые, — то marginwidth?

Как бы не так. Как вы уже могли заметить, стандарт HTML трактуется разными разработчиками довольно широко — в том смысле, что они игнорируют отдельные предусмотренные стандартом параметры и даже целые дескрипторы и создают вместо них свои. Так вышло и с параметрами отступов: в Internet Explorer используются rightmargin и leftmargin, а в Mozilla — marginwidth. Таким образом, в IE мы получаем дополнительную "степень свободы" - возможность задавать правый и левый отступы разной ширины.

Броузеры игнорируют незнакомые параметры. Web-дизайнеры пользуются этим свойством, задавая в дескрипторе <BODY> параметры для обоих типов броузеров — и IE, и Mozilla (рис. 13.1).

А как обстоят дела с отступами в остальных броузерах? Ответ здесь один: пробуйте и узнаете. Не зря Web-дизайнеры хранят на своих компьютерах по несколько версий разных броузеров и пробуют, пробуют, пробуют...

Вертикальные отступы задаются аналогично. В Internet Explorer верхний и нижний отступы определяются соответственно параметрами topmargin и bottommargin. В броузерах Mozilla они одинаковые и определяются параметром marginheight.

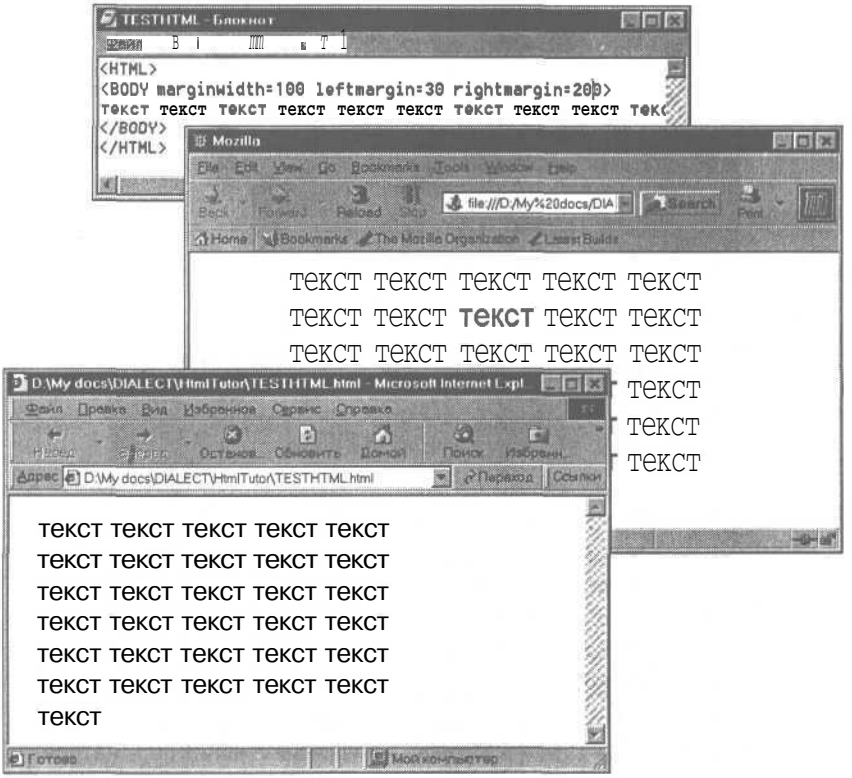


Рис. 13.1. Горизонтальные отступы в Internet Explorer задаются параметрами `rightmargin` и leftmargin, в Mozilla — `marginwidth`

Изменяя размер окна броузера, вы вскоре убедитесь, что если вся страница в окне не помещается и требуется прокрутка, то отступы "отсчитываются" от краев всего текста, а не того фрагмента, что виден в окне.

А не многовато ли параметров — целых четыре?

Действительно, отступы с противоположных сторон часто бывают одинаковые. В этом случае можно обойтись без `bottommargin` и rightmargin, а отступы снизу и справа будут такими же, как сверху и слева, задаваемые соответственно параметрами `topmargin` и leftmargin (рис. 13.2).

Иногда это может быть красиво, например, если на начальной странице нет ничего, кроме эмблемы и пары кнопок. Но только если по по вашему недосмотру эмблема не получится больше, чем экран...

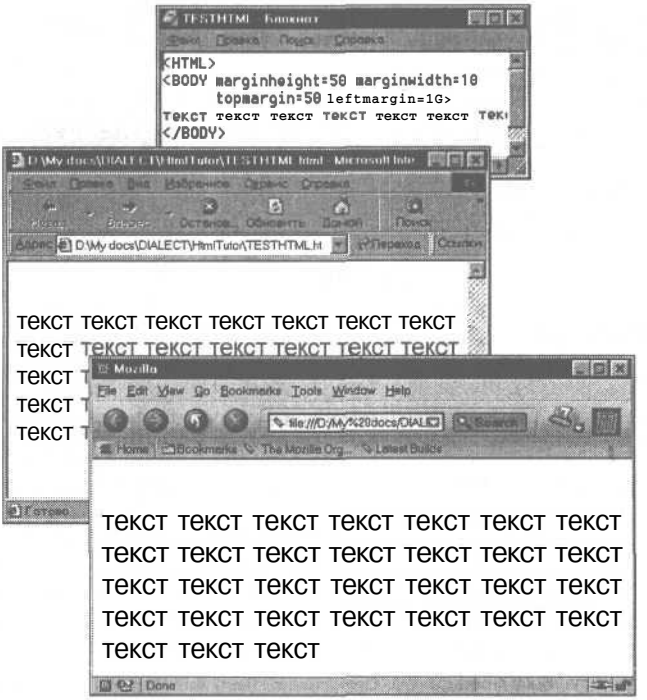


Рис. 13.2. У этой страницы одинаковые отступы с обеих сторон — и в IE, и в Mozilla — и никаких лишних параметров

Рис. 13.3. В этом окне прокрутка не появится никогда — даже если содержимое не будет в нем помещаться

Цвет фона

Иногда хочется "подложить" под текст какой-нибудь веселенький орнамент, текстуру или хотя бы просто цвет. И здесь нам на помощь приходят уже знакомые параметры. Только здесь, на новом месте, они "работают" уже для всей страницы. Цвет фона определяется параметром bgcolor. Значением этого параметра, как всегда, является код цвета в формате lRRGGBB (см. главу 10).

Если же мы хотим, чтобы фоном служила не однотонная подложка, а узор или текстура, используем другой знакомый нам параметр — background. Его значением, как мы помним, является URL графического файла. Используется этот параметр так же, как и для таблиц (см. главу 10).

Что получится, если задать в дескрипторе <BODY> сразу два параметра — и bgcolor, и background? На первый взгляд, ничего особенного: просто узор ложится "поверх" фонового цвета и полностью его закрывает. Однако иногда совместное использование этих параметров дает интересные и полезные результаты.

Поскольку предполагается, что фоновое изображение несет мало смысловой нагрузки, то оно загружается последним. Вероятно, вам случалось открыть страницу, просмотреть ее (или даже прочесть), совсем уже собраться уходить, как вдруг выяснялось, что у нее, оказывается, есть еще и фон! Просто он все это время "грузился". Однако фоновый цвет, в отличие от изображения, появляется в окне броузера практически сразу. Этот факт часто используют, чтобы, пока загружается фоновое изображение, страница представала перед посетителями хотя бы в "родных" тонах. Достаточно только подобрать цвет, максимально близкий к тонам фонового рисунка. Кроме того, если в качестве фонового изображения используется узор с прозрачным фоном (см. главу 8), то он "накладывается" на цвет, заданный параметром bgcolor, что позволяет добиться различных интересных эффектов.

Наконец, еще одно любопытное свойство фонового изображения определяется параметром bgproperties: если присвоить ему значение fixed, то фоновое изображение не "прокручивается" вместе с текстом, а стоит на месте. К сожалению, это свойство поддерживается только Internet Explorer, поэтому вряд ли стоит не его основании создавать какие-то важные для страницы эффекты: будет жаль, если они останутся незамеченными посетителями, пользующимися другими броузерами.

Предположим, мы имеем дело с фреймовой структурой. Можно ли сделать так, чтобы все фреймы были одного цвета, причем этот цвет задавался в одной точке кода? Например, так:

```
<BODY bgcolor=pink>
<FRAMESET параметры>
   <FRAME src="head.htm">
   <FRAMESET cols=*,2*>
         <FRAME src="references.htm">
         <FRAME src="news.htm">
```

```
    </FRAMESET>
</FRAMESET>
<BODY>
```

Нет, нельзя. Дескриптор <FRAMESET> нельзя располагать внутри <BODY>.

Цвет текста

По умолчанию текст на странице выводится черным цветом, гиперссылки — синим, использованные ссылки — фиолетовым. Но эти цвета не всегда "вписываются" в дизайн страницы. Конечно, можно воспользоваться уже знакомым нам дескриптором <FONT> (см. главу 4). Однако делать это каждый раз слишком скучно и неэффективно: страница быстро превращается в трудно читаемое нагромождение дескрипторов.

Вместо этого можно задать используемые по умолчанию цвета текста и ссылок в дескрипторе <BODY>. Цвет текста определяется параметром text. Что является значением этого параметра? Разумеется, константа или код цвета в формате #RRGGBB (рис. 13.4).

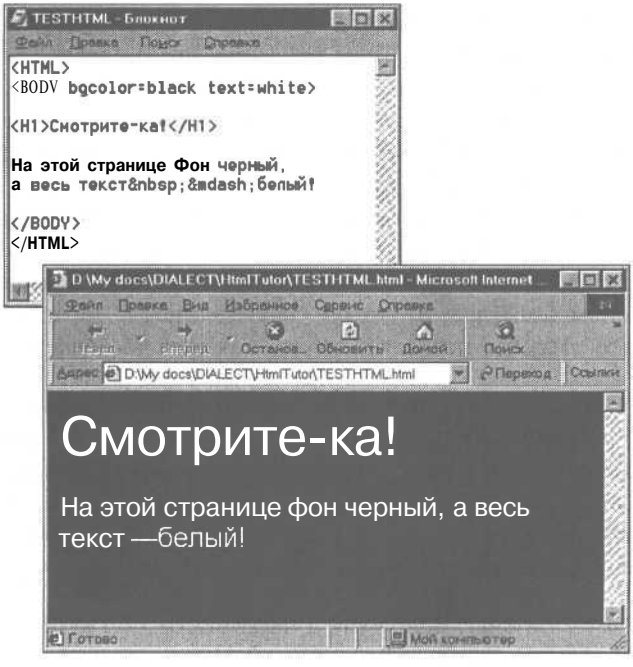


Рис. 13.4. Параметры дескриптора <BODY> позволяют задать цвет фона и текста страницы

Аналогичным образом определяются цвета ссылок. Но здесь используется не один, а три параметра: link для цвета непосещенных ссылок, vlink (от *visited link)* для посещенных и alink — для активной ссылки, т.е. для той, на которой посетитель щелкает в данный момент.

На некоторых страницах цвет ссылки меняется еще до щелчка — как только посетитель наведет указатель мыши на ссылку. Это делается не с помощью HTML-кода, а другими средствами, например, путем использования сценариев JavaScript.

Резюме

Код Web-страницы делится на две зоны: заголовок и тело. Содержимое заголовка будет подробно описано в главе 14. Тело Web-страницы содержит код, "отвечающий" за то, что, собственно, выводится в окне броузера и заключено между дескрипторами <BODY> и </BODY>. Параметры дескриптора <BODY> описывают свойства страницы в целом — вертикальные и горизонтальные отступы, цвет фона и фоновый рисунок, режим прокрутки, используемые по умолчанию цвета текста и ссылок.

Тест

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?
   а) <BODY margin=10>
   б) <BODY marginwidth=10>
   в) <BODY leftmargin=10>
   г) <BODY leftmargin=10%>
2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?
   а) <BODY text=4>
   б) <BODY text=Times>
   в) <BODY text=black>
   г) <BODY text=0>
   д) <BODY text=#1234>
   е) <BODY text=1234>
3. Как должен выглядеть код, чтобы гиперссылки в тексте были зеленого цвета?
   а) <BODY link=green>
   б) <BODY alink=green>
   в) <BODY blink=green>
   г) <BODY xlink=green>
   д) <BODY ylink=green>
   е) <BODY vlink=green>
4. Как должен выглядеть код, чтобы фон страницы был желтого цвета?
   а) <BODY color=yellow>
   б) <BODY bgcolor=yellow>
   в) <BODY background=yellow>

Глава 14

Внешние параметры Web-страницы

В этой главе...

- Кто ваши посетители?
- Как страницу назовете...
- Назначение дескриптора <META>

Кто ваши посетители

Если бы речь шла о простом текстовом или даже гипертекстовом документе, было бы вполне достаточно описать его форматирование. Но Web-страницы живут в Internet самостоятельной жизнью, о которой их владельцы часто только догадываются по счетчику посещаемости и другим косвенным признакам. Для того чтобы страница жила полноценной жизнью, а не просто занимала место на сервере, ей, в первую очередь, нужен хорошо продуманный заголовок. Информация, размещенная в заголовке, служит одним из признаков, по которым поисковые системы определяют тематику страницы и то, насколько она соответствует критерию поиска.

Впрочем, так ли уж нужен заголовок?

Для того чтобы ответить на этот вопрос, необходимо прежде всего определить, для кого предназначена страница и как она будет попадать к своим читателям. Планируете ли вы разместить ее в Internet или только в локальной сети, или это вообще электронный документ, распространяемый на компакт-дисках? Кто будут ваши посетители? Сколько их? Каким образом они узнают о вашей странице?

Например, страница, рассчитанная на строго ограниченный круг людей (члены некоторого клуба, студенты определенного курса конкретного вуза, члены вашей семьи), вообще говоря, мало нуждается в заголовочной части. Вы сами сообщите всем этим людям, где им следует искать страницу, которую вы для них создали.

Но если страница рассчитана на широкий круг неизвестных вам людей, заголовочная информация необходима: по ней эти люди найдут вашу страницу в Internet, воспользовавшись поисковым сервером.

Те, кто пользовался поисковыми серверами Internet, знают, что эти мощные системы часто выдают по запросу ссылки на тысячи и десятки тысяч страниц. Подобрать критерий, по которому вам выдадут 3–5 ссылок, среди которых есть хоть одна полезная, удается довольно редко.

Поэтому большинство интернетчиков (к которому отношусь и я, и, вероятнее всего, вы тоже) поступает просто: ограничивается просмотром тех ссылок, которые попали на первую страницу, в крайнем случае на первую пару-тройку страниц.

А теперь поставим себя на место владельца страницы. Устроит ли его, если его сайт будет значиться в списке ссылок, скажем сто девяносто восьмым? Вряд ли. С тем же успехом он мог бы вообще туда не попасть. Напротив, чем чаще его сайт будет в первой десятке, тем лучше.

Что нужно делать, чтобы достичь этой цели? Для этого нужно знать основные принципы работы поисковых серверов Internet. Вкратце они таковы. В основе поискового сервера лежит специальная программа, именуемая спайдером (от англ. *spider* — паук). Этот паук "ползает" по "Всемирной Паутине" и ищет там новые страницы. Каждую такую страницу он заносит в свой каталог. При этом он старается как можно лучше определить тематику страницы, чтобы потом можно было быстро и точно определить, насколько она подходит к критерию поиска, заданному посетителем сервера. Страницы в списке выдаваемых ссылок обычно располагаются так: самые подходящие (по мнению сервера) вначале, менее подходящие в конце.

Как спайдер, компьютерная программа, определяет, чему посвящена страница? Оказывается, мы, люди, создаем информацию по определенным правилам, понятным программе. Так, самые важные слова мы выносим в заголовки и в подрисуночные подписи, выделяем более крупным шрифтом, цветом, курсивом. Наконец, просто помещаем их ближе к началу предложения. Зная эти правила, опытные специалисты по "раскрутке" (повышению посещаемости) сайтов умеют, лишь слегка подредактировав текст, увеличить число ежедневных посетителей страницы.

Какое отношение ко всему этому имеет заголовок страницы? Самое прямое. Он содержит информацию, которую спайдеры анализируют самым тщательнейшим образом, справедливо считая, что уже по одним данным заголовка можно достаточно полно определить тематику страницы.

Как страницу назовете...

Какого рода вспомогательную информацию можно разместить между дескрипторами <HEAD> и </HEAD>? Во-первых, — *название*. Нет, это не имя файла и не то, что пишут вверху страницы большими и красивыми буквами. Название Web-страницы пишется в *заголовке окна броузера*. В коде оно выделяется дескрипторами <TITLE> (рис. 14.1).

Можно ли сделать заголовком страницы имя файла? Ведь Windows допускает длинные имена с кириллицей? Да, можно. Но лучше не надо. Дело в том, что многие серверы Internet работают под управлением ОС семейства UNIX. А там правила именования файлов жестче, чем в Windows. Поэтому, если не хотите столкнуться с самыми неожиданными проблемами, лучше придерживайтесь старого доброго форма-

та 8.3: восьмисимвольное имя, трехсимвольное расширение, все буквы строчные, никаких пробелов и никакой кириллицы. Не огорчайтесь: все равно эти имена будут читать главным образом броузеры и Web-серверы, а им не до красот русского языка.

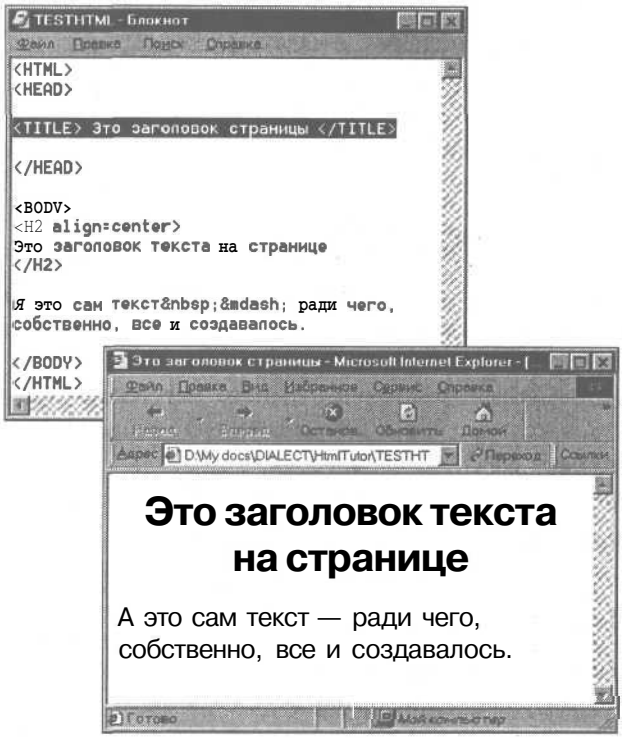


Рис. 14.1. Заголовок страницы и заголовок текста — разные вещи

Другое дело — заголовок страницы. Его читают не только машины, но и люди, поэтому приходится угождать и тем, и другим. И себя не забыть. В связи с этим рекомендуется помнить следующие правила.

* Заголовок — это первое, что видит и успевает прочесть несколько раз (а еще лучше запомнить) посетитель страницы, пока она загружается. Скучный заголовок может стать последним, что узнает посетитель о вашей странице. Вдобавок этот текст выводится на соответствующей кнопке панели задач Windows, в списке "закладок" броузера (если ваш сайт так понравится посетителю, что он его туда занесет), а также в списках, выдаваемых поисковыми серверами. Учитывая последнее обстоятельство, рекомендуется поместить в заголовок ключевые слова, характеризующие вашу страницу. Увы, знание HTML здесь не поможет — только ваша фантазия и смекалка.

♦ Как заметили психологи и как вы можете заметить сейчас сами, ПРОПИСНЫЕ БУКВЫ и восклицательные знаки производят на посетителя скорее отталкивающее, чем притягивающее действие!

* К сожалению, внутри дескриптора <TITLE> нельзя использовать другие дескрипторы. А значит, никакого курсива, цветного текста и крупного шрифта. Только сила слова убедит посетителя заглянуть на ваш сайт.

* Рекомендуемая максимальная длина заголовка — 60 символов, считая пробелы. А вообще, чем лаконичнее, тем лучше. Ведь заголовок должен полностью поместиться в строке броузера. Иначе броузер просто оборвет фразу на середине и поставит многоточие. Вряд ли вам хотелось бы, чтобы пользователи увидели в окне что-то вроде "Рассказ о том, как однажды случилось...".

Назначение дескриптора <META>

Второй важный дескриптор заголовочного раздела — <META>. В отличие от <TITLE>, он содержит главным образом "машинную" информацию — для броузеров и поисковых серверов.

Дескриптор <META> "самодостаточен". У него нет текста, формат которого он определяет, а следовательно, нет и закрывающего дескриптора. Зато есть много параметров, именуемых здесь *мета-записями*.

Основной формат дескриптора <META> таков:

```
<META name="имя мета-записи" content="значение">
```

Иногда вместо параметра name в дескрипторе <META> используют параметр http-equiv:

```
<META http-equiv="тип значения" content="значение">
```

Такие мета-дескрипторы эквивалентны заголовкам HTTP и позволяют передать броузеру дополнительную информацию о странице.

В отличие от других дескрипторов с параметрами, в каждом дескрипторе <META> допускается использование только одной мета-записи. Поэтому, когда нужно использовать несколько мета-записей, создают целый список из дескрипторов <META>:

```
<META name="мета-запись 1" content="значение 1">
<META http-equiv="мета-запись 2" content="значение 2">
...
<META name="мета-запись л" content="значение п">
```

Рассмотрим назначение некоторых наиболее распространенных мета-записей.

Keywords

Эта мета-запись используется для перечисления (через запятую) слов и выражений, максимально соответствующих теме страницы. Поисковые серверы анализируют эти слова (а также другие параметры страницы) и "принимают решение", соответствует ли данная страница запросу. Значением параметра keywords обычно служит длинный перечень, чем-то похожий на заклинание, например:

```
<META name="keywords" content="моя первая страничка, Web-дезайнер, Веб-дизайнер,
Web designer, разработать Web-страницу, чайник">
```

К выбору ключевых слов (как, впрочем, и ко всему в Web-дизайне) следует отнестись творчески, учитывая особенности поведения людей и машин.

* Обычно поисковые серверы неплохо разбираются в правилах грамматики. Поэтому, если в списке ключевых слов есть, например "Web-дизайнер", то страница найдется и по запросу "Web-дизайн". Следовательно, список можно сократить, оставив самое длинное из подобных слов.

* Далеко не каждый посетитель поисковой системы знает русский язык так же хорошо, как вы или как программа проверки орфографии. Поэтому иногда имеет смысл поставить рядом с "дизайнером" — "дезайнера".

- В Internet встречаются "универсальные" списки ключевых слов, якобы гарантирующие попадание вашей страницы в первую десятку чуть ли не по любому запросу. Действительно, таким образом можно ненадолго повысить посещаемость. Но нужны ли вам посетители, которые приходят незаинтересованными, а уходят недовольными? И нужна ли вашей странице и вам дурная слава, которую, как известно, легче найти, чем потерять? А некоторые поисковые системы могут за подобные "трюки" исключить страницу из своего индексного списка.

- "Выбросить" из списка могут и за слишком длинный перечень ключевых слов, а также за явно преднамеренное частое повторение одного и того же. Рекомендуемый предел — 200 символов и не более 5-6 повторений.

- Наконец, если фантазии маловато, можно пойти на хитрость: найти с помощью нескольких самых популярных поисковых серверов страницы, максимально близкие к вашей по теме, и позаимствовать ключевые слова у тех, что попали в начало списка. Но имейте в виду: эти страницы могут возглавить список вовсе не из-за ключевых слов, а по каким-то другим причинам, которых вы не знаете.

Description

У электронных публикаций, как у журнальных статей, есть аннотации. Эти аннотации и заносятся в мета-запись description.

Вы, вероятно, замечали, что обычно поисковые системы выдают по запросу адреса с описаниями. Иногда эти описания — бессмысленные обрывки каких-то названий, перечни кодировок и прочая галиматья. Из-за этого, бывает, создается впечатление, будто роешься в мусорной куче... Если вам встретится такое, с позволения сказать, "описание" страницы, знайте: ее создатель не стал возиться с мета-записью description, а вверху страницы стояли те самые переключатели кодировок.

Если же не полениться, то с помощью мета-записи description можно убить сразу двух зайцев: и сделать хорошую, привлекательную аннотацию, и лишний раз упомянуть ключевые слова:

```
<META name=" description" content="Это моя первая Web-страничка. Я уже полчаса
как Web-дизайнер!">
```

Что-то вроде этого... Хотя если постараться, можно придумать и поумнее.

Как и в случае со списком ключевых слов, аннотация не должна быть слишком длинной. Если она превысит 100–200 символов, поисковик просто оборвет текст на полуслове.

Robots

Назначение этой мета-записи — проинструктировать спайдера: как быть с обнаруженной страницей? Из ее значений наиболее часто употребляются index, noidex, folow и nofollow, определяющие, соответственна, нужно ли индексировать эту страницу, а также нужно ли переходить на те страницы, на которые она ссылается. Если нужно указать несколько значений мета-записи robots, они перечисляются через запятую. Например, для того чтобы страница не попала в индексный список поисковых систем (то ли потому, что не предназначена для посторонних, то ли потому что просто недоделана), а другие страницы сайта, на которые она ссылается, туда попали, пишут следующее:

```
<META name="robots" content="noindex, follow">
```

К сожалению, не все поисковики это понимают. Надежнее создать в корневом каталоге сервера файл robots.txt и занести туда список страниц, не подлежащих индексированию. Впрочем, все, что касается сервера, лучше делать вместе с администратором этого сервера.

Generator

Люди редко используют эту мета-запись. Зато ее очень любят визуальные HTML-редакторы, заявляющие таким образом о своем "авторстве":

```
<META name="GENERATOR" content="Microsoft FrontPage Express 2.0">
```

или

```
<META name=Generator content="Microsoft Word 9">
```

Что еще за Word 9? Оказывается, это просто "внутреннее" название программы, известной нам как Word 2000...

Author и copyright

А вот эти дескрипторы предназначены для защиты прав настоящих авторов страницы:

```
<META name="autor" content="И.П. Сидоров, u_popa_byla@on-ee.lub.il">
<META name="copyright" content="описание ваших авторских прав и того, что вы сделаете их нарушителю">
```

Content-type

Для задания мета-записи content-type используется параметр http-equiv. С ее помощью можно (и нужно!) "намекнуть" броузеру о типе и кодировке Web-страницы, например:

```
<META http-equiv=content-type content="text/html; charset=windows-1251">
```

Expires

Для задания мета-записи expires используется параметр http-equiv. Она определяет дату, по наступлении которой страница устареет, и броузеру придется снова обратиться за ней на сайт, а не пытаться загрузить старую версию из кэша. Например, дескриптор

```
<META http-equiv="expires" content="Wed, 20 Jun 2001 00:00:01 GMT">
```

означает, что страница устареет 20 июня 2003 года.

Refresh

Для задания мета-записи refresh используется параметр http-equiv. Эта запись используется в тех случаях, когда через определенный промежуток времени на месте данной страницы должна загрузиться другая, например, если это страница ожидания загрузки, или заставка, или виртуальный тест, ответ на который должен прийти не позже заданного времени.

Значением мета-записи refresh является пара "время; URL". Например, запись

```
<META http-equiv=refresh content="20; url=http://server/next.html">
```

означает, что по истечении 20 с сервер должен загрузить страницу http://server/next.html.

Резюме

Информация, расположенная в заголовочной части Web-страницы, предназначена главным образом для поисковых систем. Ее назначение — максимально точное сжатое описание страницы, которое позволило бы правильно позиционировать страницу в каталоге поискового сервера и в списке ссылок, выданных пользователю в ответ на запрос. Среди главных элементов заголовочной части следует отметить название страницы, помещаемое между дескрипторами <TITLE> и </TITLE>, а также аннотацию и список ключевых слов, вводимые с помощью мета-записей keywords и description.

Тесты

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а) `<HEAD>Заголовок страницы</HEAD>`

   б) `<TITLE text="Заголовок страницы">`

в) <TITLE>Заголовок страницы</TITLE>

г) <TITLE><U>Заголовок страницы</U></TITLE>

д) <TITLE align=center>Заголовок страницы</TITLE>

2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <META name=keywords>список ключевых слов</META>

б) <META name=keywords contents="список ключевых слов">

в) <META name=keyword content="список ключевых слов">

г) <META name=keywords content="список ключевых слов">

д) <META http-equiv=keywords content="список ключевых слов">

3. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <META name=description content="аннотация к странице" name=copyright content="А.Б. Иванов">

б) <META name1=description content="аннотация к странице" name2=copyright content="А.Б. Иванов">

в) <META name=description content="аннотация к странице" http-equiv=copyright content="А.Б. Иванов">

г) <META name=description content="аннотация к странице">
<META name=copyright content="А.Б. Иванов">

Д) <META>
name=description content="аннотация к странице"
name=copyright content="А.Б. Иванов"
</META>

Глава 15

Основные принципы каскадных таблиц стилей

В этой главе...

Зачем нужны таблицы стилей

В сущности, мы уже знакомы со всем, чем богат "классический" HTML. Действительно, этот инструмент предоставляет все необходимые возможности для создания Web-страниц. Однако все это довольно грубые инструменты. Например, с помощью параметра size мы можем выбрать только один из семи размеров шрифта, и эти размеры все равно будут зависеть от настройки броузера. О междустрочном интервале и говорить нечего: он просто никак не определяется. И вообще, те, кто знакомы со стилями Word, уже давно должны были соскучиться по этому мощнейшему средству форматирования.

Действительно, в "классическом" HTML — таком, который мы знали до сих пор, — нельзя задать какое-нибудь сложное форматирование (например, сочетание цвета, размера и начертания) одним дескриптором. Вместо того чтобы один раз описать такое форматирование, а потом только задавать его с помощью какого-нибудь короткого ключевого слова, приходится каждый раз писать длинный код наподобие такого:

<FONT size=5 color=red><I>красный</I> курсив размера 5</I></FONT> обычный текст
<FONT size=5 color=red><I>опять</I> красный курсив</I></FONT> опять обычный текст

Однако на самом деле в HTML такое средство есть и называется оно CSS (Cascading Style Sheets — каскадные таблицы стилей). По принципу работы они очень похожи на стили форматирования, используемые в текстовых процессорах и, в частности в MS Word: стилем называется описание какого-нибудь сложного форматирования. Такому стилю присваивается имя, по которому и происходит обращение к стилю в нужный момент. Таким образом мы избегаем скучного перечисления каждый раз одних и тех же параметров, а в случае HTML еще и существенно сокращаем код страницы.

Простейшее описание стиля

Стиль всегда создается на базе какого-либо дескриптора. Проще всего сделать это с помощью параметра style, значением которого служит описание стиля. Предположим, мы хотим создать стиль с такими параметрами: курсив, 18 пунктов, белый на черном фоне. Для этого можно использовать такой код:

```
<I style="font-size: 18; color: white; background-color: black">
Это текст, к которому применен стиль.</I>
```

Как видим, принцип очень прост. Стиль состоит из атрибутов, которые в некотором смысле подобны параметрам дескриптора. Каждый атрибут имеет предопределенное имя и область возможных значений. Для присвоения атрибуту значения указывается имя этого атрибута, затем двоеточие и само значение. Атрибуты перечисляются через точку с запятой. Расстановка пробелов, как и в случае с параметрами дескрипторов, может быть произвольной. Именно такой список, который в данном случае служит значением параметра style, и образует стиль. В данном случае были использованы три атрибута: font-size — для определения размера шрифта, color – для определения цвета шрифта и background-color — для определения цвета фона.

Какой дескриптор сделать базовым для описания стиля? В сущности, это может быть любой парный дескриптор, применяемый для форматирования текста. Если же ни одно из свойств стиля не совпадает с каким-нибудь дескриптором, можно воспользоваться дескриптором <DIV>. Это парный дескриптор, название которого происходит от английского слова *division* — раздел. Дескриптор <DIV> предназначен для описания форматирования произвольной области текста. Его единственным атрибутом является align, который в данном случае принимает те же значения, что и в дескрипторе абзаца <P> (см. главу 2). Этот дескриптор широко используется для описания стилей.

Итак, технология описания стиля вроде бы понятна. Но как его использовать повторно? Нетрудно убедиться, что стиль, описанный таким образом, распространяется только до ближайшего закрывающего дескриптора </I> (рис. 15.1).

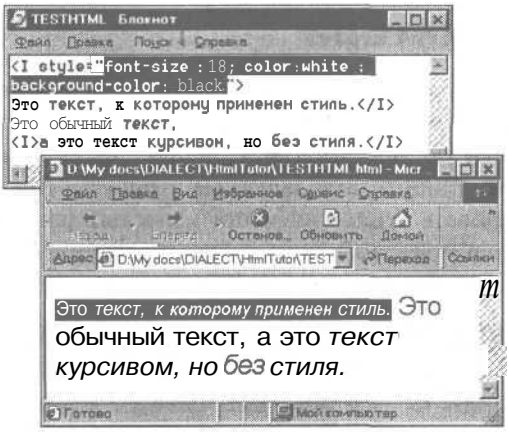


Рис. 15.1. Описание стиля внутри дескриптора: атрибуты разделяются точками с запятой; пробелы расставляются произвольно; форматирование распространяется до соответствующего закрывающего дескриптора

Таблицы стилей

Как создать стиль "многоразового использования"?

Вспомним: в названии главы упоминаются не просто стили, а "таблицы стилей". Это неспроста. Стиль, помещенный внутрь такой таблицы, в отличие от стиля, описанного прямо в коде, можно использовать многократно.

Таблица стилей представляет собой парный дескриптор <STYLE>, внутри которого последовательно описаны используемые стили;

```
<STYLE>
      идентификатор_1 {атрибут_1: значение; атрибут_2: значение; ...}
      идентификатор_2 {атрибут_1: значение; атрибут_2: значение; ...}
      ...
      идентификатор_N {атрибут_1: значение; атрибут_2: значение; ...}
</STYLE>
```

Как видим, описание стиля представляет собой некий идентификатор, по которому броузер впоследствии распознает данный стиль, и список атрибутов, составленный по уже знакомому нам принципу (между значением и атрибутом — двоеточие, элементы списка разделены точками с запятой). Список заключается в фигурные скобки.

Что может служить идентификатором стиля? В отличие от стилей, используемых для форматирования текста в текстовых процессорах, применяемые в HTML стили CSS не имеют имен. А те заменители имен, которые они имеют, используются несколько иначе. Давайте рассмотрим применение идентификаторов стиля на примере.

Часто для всего сайта создаются общие таблицы стилей. Такие таблицы выносятся в отдельные файлы. Для связи с ними используется дескриптор <LINK>. Это одинарный дескриптор, одним из параметров которого служит URL таблицы. Например, для подключения таблицы стилей, которая находится в файле styles.css, можно воспользоваться следующим дескриптором:

```
<LINK rel="stylesheet" href="styles.css" type="text/css">
```

Стиль дескриптора

Что получится, если взять описание стиля, которое мы уже создали (см. рис. 15.1), и поместить его в таблицу стилей согласно уже известным нам правилам? Результат показан на рис. 15.2. Как видим, теперь все фрагменты текста, выделенные курсивом, приобретают заданные атрибуты. Для того чтобы их снять, приходится корректировать стиль "на месте", каждый раз отдельно. Причем, как видно на рисунке, атрибуты стиля можно переопределить как с помощью другого стиля, так и с помощью обычных дескрипторов, таких как <FONT>.

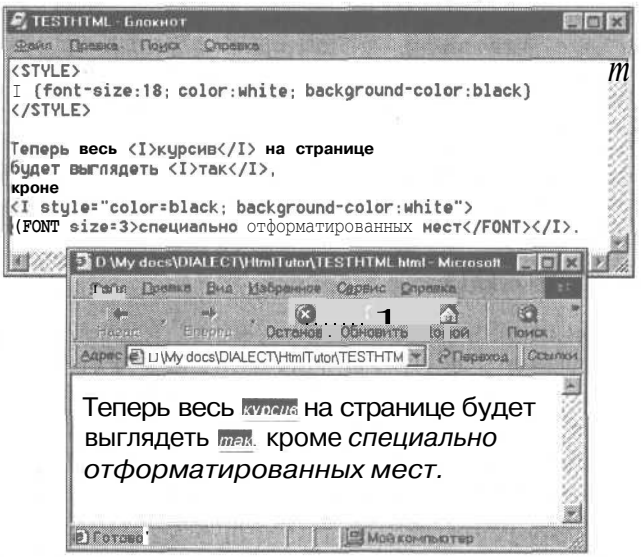


Рис. 15.2. Свойства дескриптора, переопределенные в таблице стилей, действуют в пределах всей страницы. Но их можно переопределить "на месте" с помощью другого стиля или обычного дескриптора HTML

Обратите внимание: при описании стиля внутри конструкции <STYLE> имя дескриптора не заключается в угловые скобки.

В каскадных таблицах стилей можно задать стиль не только для отдельного дескриптора, но и для их комбинации. Например, если нужно, чтобы некий стиль применялся не к любому курсиву, а только к полужирному, используется код, показанный на рис. 15.3. Таким образом, стиль применяется только для комбинации вложенных дескрипторов.

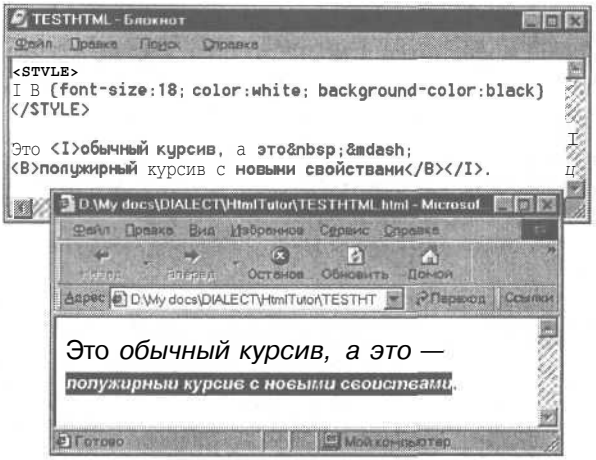


Рис. 15.3. Стиль может применяться не только для одного дескриптора, а и для нескольких вложенных

Подклассы дескрипторов

Устраивает ли нас стиль, переопределяющий свойства дескриптора?

Если на всей странице вместо, например, простого курсива мы хотим использовать такой, как показан на рис. 15.2, то, безусловно, да. Но если наряду с новым стилем нам нужен и обычный курсив, то такой метод не совсем подходит.

Вместо того чтобы каждый раз переопределять готовый стиль, было бы гораздо проще сразу, уже в таблице стилей, создать несколько вариантов курсива и использовать их по мере необходимости. Для этого в таблице стилей создается объект, именуемый *подклассом* дескриптора. Имя подкласса дескриптора формируется из имени дескриптора и имени подкласса, разделенными точкой, а описание этого объекта выглядит так же, как описание любого другого стиля:

```
имя_дескриптора.класс {атрибут_1: значение; атрибут_2: значение; ...}
```

Например, подкласс дескриптора `<I>` с уже описанными выше свойствами должен выглядеть так:

```
I.inverse {font-size:18; color:white; background-color:black}
```

Как вы уже, вероятно, догадались, inverse — имя нового подкласса. Для того чтобы воспользоваться им в HTML-коде, применяется специальный параметр — class, — которому и присваивается имя подкласса (рис. 15.4).

В объектно-ориентированном программировании, если вам знаком такой термин, также существует понятие классов. Эти классы и классы CSS в чем-то аналогичны: те и другие описывают свойства неких объектов. В данном случае под объектами понимаются элементы Web-страницы.

Действительно, таким образом можно "уточнить" форматирование, обеспечиваемое дескриптором, и даже создать несколько вариантов форматирования на базе одного дескриптора. Однако использовать созданный нами подкласс inverse для другого дескриптора, например <P>, нельзя (рис. 15.5).

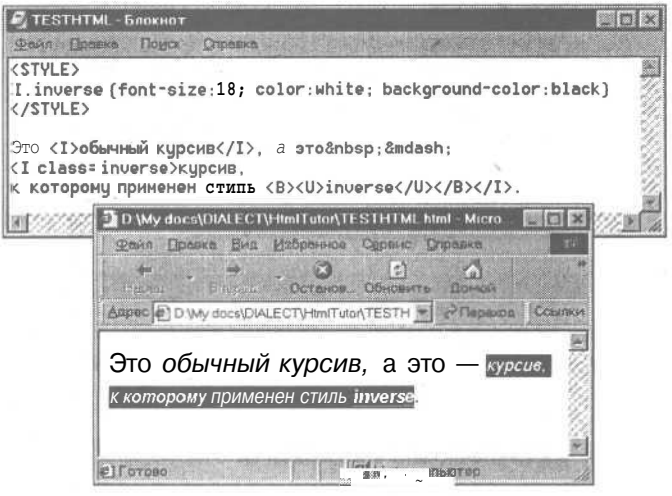


Рис. 15.4. Подкласс дескриптора — это стиль, в котором к свойствам данного дескриптора добавлены новые

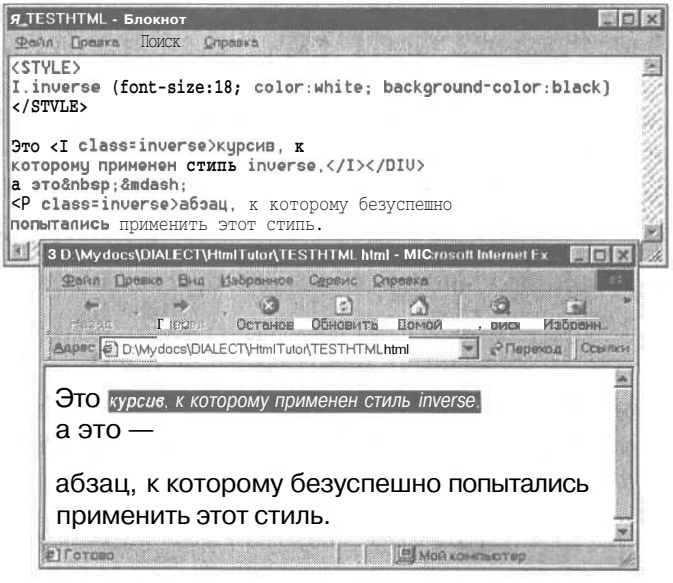


Рис. 15.5. Попытка применить стиль, описанный в виде подкласса одного дескриптора, к другому дескриптору, ни к чему не приводит

Как создать стиль, применимый сразу к нескольким дескрипторам? Например, стиль, который применял бы те же свойства, — размер 18 пунктов, белый текст на черном фоне — не только к курсиву, но и к полужирному шрифту, и вообще к любому другому дескриптору?

Иногда в таких случаях можно воспользоваться описанным выше дескриптором <DIV>: создать в таблице стилей его подкласс и пользоваться им по мере необходимости. Однако при этом следует иметь в виду, что содержимое дескриптора <DIV>, как и дескрипторов абзацев и заголовков, выводится в отдельном абзаце (рис. 15.6).

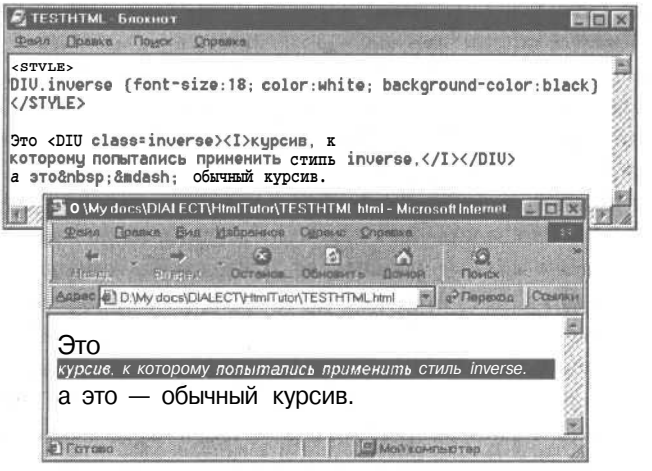


Рис. 15.6. Помните, что, применяя к фрагменту текста стиль с помощью дескриптора <DIV>, вы выделяете этот фрагмент в отдельный абзац

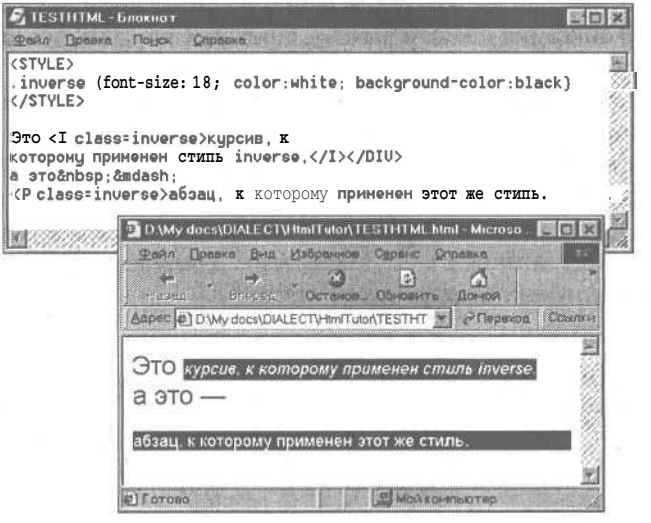


Рис. 15.7. Класс, созданный без указания имени дескриптора, применим к любому дескриптору

Есть более универсальное и изящное решение поставленной задачи. Оно заключается в создании в таблице стилей класса без указания дескриптора. Такой класс будет применим к любому дескриптору (рис. 15.7).

Описание цветов

Мы уже знаем несколько способов работы с цветом в HTML, как для шрифта, так и для фона, "подкладываемого" под всю страницу или ячейки таблиц. Как правило, все они сводятся к заданию некоего параметра, значением которого является код в формате lRRGGBB.

На первый взгляд, этот способ довольно удобен. Однако стоит поработать с ним подольше, как становится ясно, что он далеко не так универсален, как кажется. С его помощью нельзя, например, изменить фон отдельного слова или буквы. А для того чтобы изменить фон абзаца, приходится выделять этот абзац в отдельную таблицу, превращая понятный фрагмент текста в путаницу кодов. С помощью каскадных таблиц стилей можно задать цвет для более мелких элементов страницы.

Как мы уже имели случай убедиться, цвет текста определяется атрибутом color. Значением этого атрибута может быть один из стандартных цветов, имеющих предопределенное имя (см. также главу 4), или уже знакомый нам код в формате lRRGGBB. Однако здесь есть и еще один способ определения цвета — по отдельным компонентам.

Вам еще не надоело, подбирая нужный цвет в графическом редакторе, каждый раз преобразовывать все три составляющие в шестнадцатеричный формат? Скорее всего, вы, поэкспериментировав пару раз и познав на практике всю "прелесть" этой рутинной работы, решили пожертвовать несколькими миллионами цветов в пользу скорости и удобства и с тех пор пользуетесь стандартной палитрой. Пожалуй, в большинстве случаев это разумно. Но позвольте предложить вашему вниманию еще один способ покомпонентного описания цвета: однажды, когда вам понадобится выйти за пределы стандартной палитры, он еще сослужит вам добрую службу.

Этот способ применяется при описании цветов в атрибутах стиля CSS. Он заключается в задании интенсивности красной, зеленой и синей составляющих цвета в абсолютных или относительных значениях. Основное преимущество этого подхода заключается в том, что значения эти задаются не в шестнадцатеричных, а в десятичных (ура!) числах. Самому большому значению интенсивности соответствует 255, самому низкому — 0. Например, для того, чтобы текст выводился красным курсивом, нужно создать такой стиль:

```
I.red {color: rgb(255,0,0)}
```

Этот код аналогичен двум другим, уже знакомым нам вариантам:

```
I.red {color: #FF0000}
```

и

```
I.red {color: red}
```

Почему максимальным значением служит именно 255? Об этом нетрудно догадаться, если вспомнить, что, независимо от способа определения цвета, для описания каждой его составляющей отводится одинаковое количество памяти. В данном случае, это 8 двоичных разрядов. Максимальное число, которое можно в них записать, это 255 в десятичной и FF в шестнадцатеричной системе счисления. Вот так все просто объясняется...

Если же мы не желаем помнить, каким числом описывается максимальная интенсивность цвета, можно воспользоваться относительными значениями:

```
I.red {color: rgb(100%,0,0)}
```

Наконец, возможно комбинированное задание фона: одни составляющие можно описать в относительных, а другие в абсолютных единицах:

```
I.yellow {color: rgb(100%,100%,0)} <!-- желтый цвет -->
I.orange {color: rgb(100%,100,0)} <!-- оранжевый цвет -->
```

Задание фона

Таким образом можно задать цвет не только текста, но и фона, на котором он выводится. Для этого используется уже встречавшийся нам атрибут background-color. С его помощью можно изменить фон не только ячеек таблиц и всей HTML-страницы, но и отдельных абзацев, слов и даже букв.

Фоном всех этих объектов может служить и узор, составленный из изображений, форматы которых поддерживает броузер (см. также главу 8). Файл, в котором хранится фоновое изображение, указывается с помощью атрибута background-image (рис. 15.8). Значение, указываемое в скобках, должно соответствовать правилу составления URL-адресов.

До сих пор мы довольствовались тем, что, если фоновое изображение оказывалось меньше той области, на которую распространяется, оно "размножалось" на манер обоев Windows. Изменить этот порядок стандартными средствами HTML нельзя. Но с помощью стилей CSS можно управлять размещением фонового изображения. За это "отвечает" атрибут background-repeat. По умолчанию он имеет значение repeat, в соответствии с которым изображение размножается по всей отведенной ему области. Если мы хотим размножить фоновое изображение только по горизонтали, в пределах одной строки, то используем значение repeat-x, а если только по вертикали — значение repeat-y. Наконец, если фоновое изображение не подлежит размножению, то используется значение no-repeat (рис. 15.9).

Давайте вспомним: какое еще привлекательное свойство фоновых изображений оказалось малодоступно нам из-за скудости средств HTML? Прокрутка. Возможность прокрутки текста, в то время как фоновое изображение остается неподвижным. В определенной степени это можно реализовать с помощью параметров дескриптора <BODY>. Но даже он поддерживается не всеми броузерами.

Рис. 15.8. С помощью атрибута background-image можно задать фоновое изображение для различных элементов страницы

Рис. 15.9. С помощью атрибута background-repeat можно управлять "размножением" фонового изображения

Атрибут CSS background-attachment позволяет задать режим прокрутки для любого фона, в любом месте. Он принимает одно из двух значений: fixed или scroll. Например, если мы хотим, чтобы фоновое изображение оставалось неподвижным, в то время как остальное содержимое страницы прокручивается, мы можем использовать следующий код:

```
<BODY style=" background-image:url(back.gif); background-attachment=fixed">
<!-- содержимое страницы -->
</BODY>
```

Вспомним еще одно свойство фоновых изображений, недоступное нам прежде: задание отступов. С помощью атрибута background-position можно задать отступы фонового изображения от границ его области, а также выровнять его относительно границ и центра.

Отступы задаются относительно верхнего левого угла области, покрываемой фоновым изображением в виде двух значений, записываемых через пробел, и измеряются в сантиметрах или в процентах. Первое значение соответствует отступу по горизонтали, второе — по вертикали. Например, запись background-position: 1cm 2cm означает, что фоновое изображение будет сдвинуто на 1 см вправо относительно левой и на 2 см вниз относительно верхней границ своей области.

Лучше задавать отступы в процентах: тогда вы не будете привязаны к размерам фоновых изображений и заполняемых ими областей. В случае если эти размеры уменьшатся (в том числе без вашего ведома, например, если пользователь изменит размер шрифта в броузере), величина отступов изменится автоматически.

Как это выглядит на практике? На первый взгляд, ничего не меняется. Созданный нами в предыдущем примере стиль .bg по-прежнему заполняет фоновым изображением всю назначенную ему область (рис. 15.10). Однако это только первое впечатление. Внимательнее присмотревшись, мы обнаружим, что фоновая картинка действительно сдвинулась на указанное расстояние, однако это расстояние было тут же заполнено в соответствии с правилом повторения узора. Если же мы запретим повторение фонового узора, то увидим, что изображение и в самом деле сместилось (рис. 15.11).

Для выравнивания фонового изображения по горизонтали используются значения left, center и right, а по вертикали — top, center и bottom. Порядок их перечисления неважен: и background-position: top right, и background-position: top right соответствуют тому же положению, что и background-position: 0 100%.

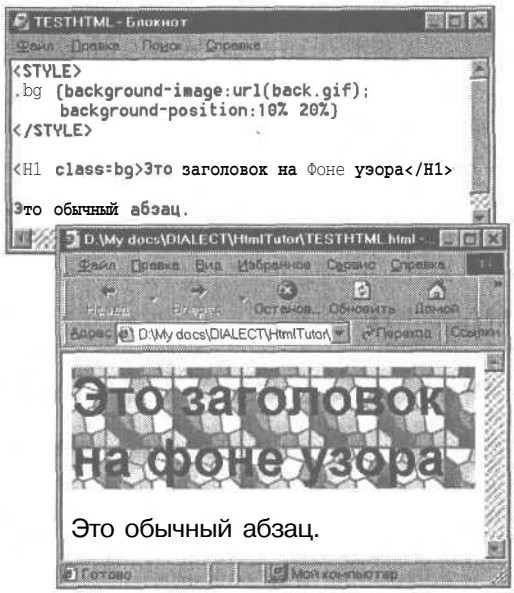


Рис. 15.10. Похоже, что фоновое изображение не сместилось. Но это только кажется из-за повторения узора

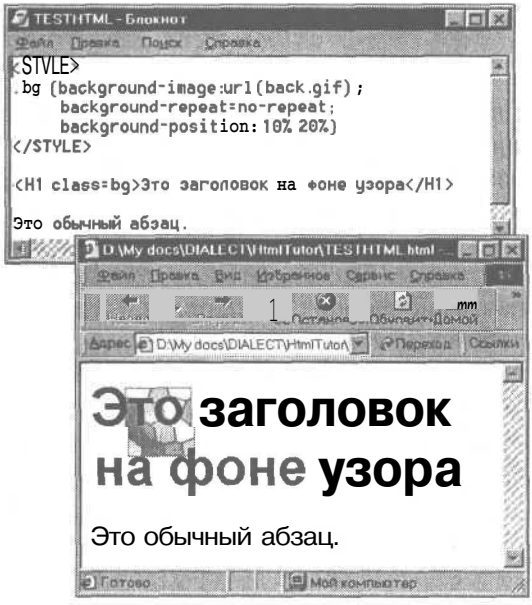


Рис. 15.11. Если запретить повторение узора, фоновое изображение сдвинется вниз и вправо

Атрибутов, описывающих свойства фона, так много и их названия так длинны, что описывать их по одному становится довольно утомительно. Вместо этого допускается описание всех свойств фона с помощью одного параметра background, причем порядок перечисления не имеет значения. Как и в случае с отступами, значения указываются через пробел. Например, стиль .bg, показанный на рис. 15.11, можно описать единственным атрибутом background:

```
.bg {background: url(back.gif) no-repeat 10% 20%}
```

Атрибуты шрифта

Как вы уже знаете, пользователь броузера может менять шрифт страницы, настраивая его так, чтобы было удобно читать. Однако в Internet довольно часто встречаются страницы, шрифты которых остаются постоянными, независимо от настройки броузера. Стремясь сохранить дизайн страниц, их создатели прибегают к помощи каскадных таблиц стилей.

Размер шрифта определяется атрибутом font-size, который мы использовали в начале этой главы. Его значением является размер шрифта в пунктах. Есть у этого атрибута и предопределенные значения: xx-small, x-small, small, medium, large, x-large, xx-large. Они соответствуют стандартным размерам шрифта, определяемых параметром size дескриптора <FONT> (см. главу 4). Относительным размерам шрифта size=+1 и size=-1 соответствуют значения smaller и larger, соответственно.

Многие атрибуты CSS дублируют параметры дескриптора <FONT>. Однако, в отличие от них, стили, созданные с помощью этих атрибутов, применимы к любому парному дескриптору HTML, внутри которого заключен текст, что очень удобно.

Впрочем, точное соответствие атрибутов CSS и параметров HTML наблюдается не всегда. Например, при разных размерах шрифта, установленных в броузере, один и тот же код приводит к разным результатам (рис. 15.12).

В CSS есть и гораздо более тонкие средства задания относительного размера шрифта. Прежде всего, это задание процентного соотношения по отношению к текущему шрифту (рис. 15.13)

Есть и другой способ — с помощью суффикса em, который ставят после цифры, обозначающей кратность увеличения или уменьшения размера шрифта. Например, в примере, показанном на рис. 15.13, вместо font-size: 50% можно было бы написать font-size: 0.5em. Другой суффикс, используемый для задания размера шрифта, ex, определяет масштаб шрифта относительно "опорной точки", соответствующей значению size=1.

Следующим свойством, довольно грубо определяемым средствами HTML, является "жирность" шрифта. Для ее описания существует единственное средство, дескриптор <B>, так что текст может находиться лишь в одном из двух состояний — "нормальный" и "полужирный".

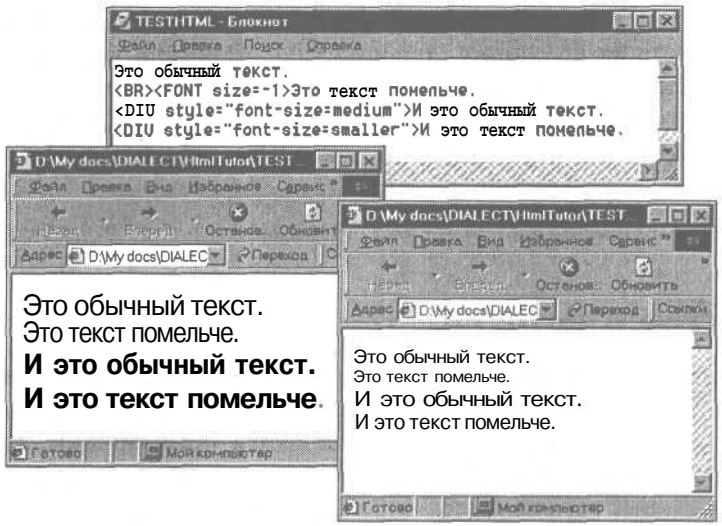


Рис. 15.12. При разных размерах шрифта, установленных в броузере, один и тот же код выглядит по-разному: в режиме крупного шрифта размер двух нижних строчек почти одинаков, однако в режиме среднего шрифта нижняя строчка явно мельче

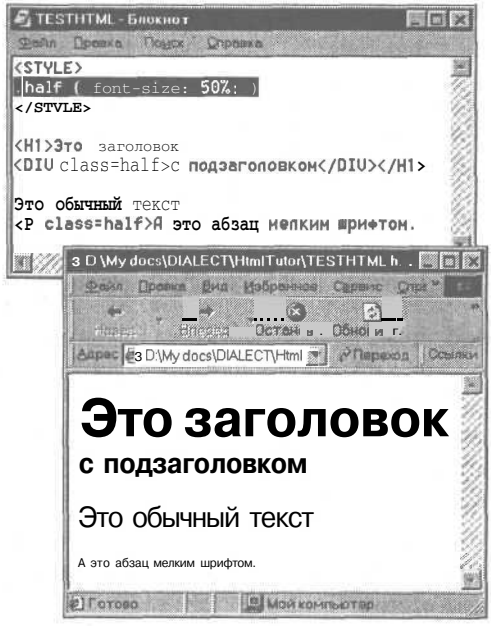


Рис. 15.13. Простейший способ точно задать изменение размера шрифта — использовать процентное соотношение

Атрибут font-weight определяет 9 степеней "жирности" шрифта, задаваемых числами от 100 до 900. Считается, что обычному шрифту соответствует значение 400, а полужирному — 700 (вместо этой цифры можно указывать ключевое слово bold). Теперь понятно, почему это начертание называется "полужирным": есть еще вдвое жирнее! Для задания относительной "жирности" используются значения bolder и lighter. Первое обеспечивает более жирный шрифт, второе — более тонкий.

Что касается курсива, то здесь в роли "дублера" дескриптора <I> выступает атрибут font-style. Он имеет всего два значения: normal (отсутствие курсива) и italic, или oblique (курсив). Аналогично работает атрибут font-variant: в режиме normal он соответствует обычному шрифту, в режиме small-caps текст выводится "малыми прописными" буквами (рис. 15.14).

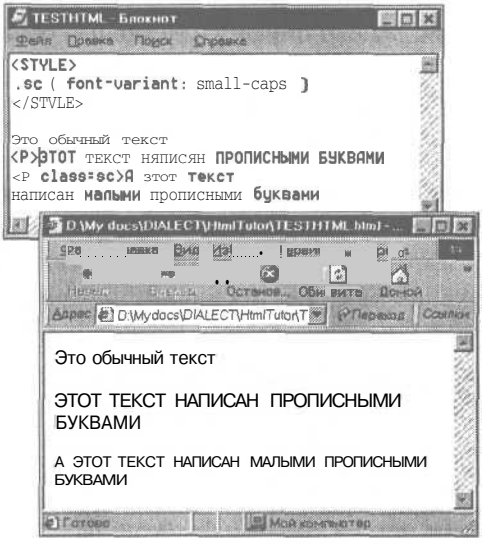


Рис. 15.14. Атрибут font-variant позволяет вывести текст "малыми прописными" буквами

Атрибуты CSS позволяют обойти "камень преткновения", за который многие так сильно — и не без оснований — "осуждают" дескриптор <FONT>. Я имею в виду параметр face. Как мы уже знаем, он, с одной стороны, полезен тем, что позволяет задать гарнитуру шрифта, лучше всего подходящую к дизайну страницы. Однако, с другой стороны, если такой гарнитуры не оказывается на компьютере посетителя этой страницы, параметр face оказывается бесполезен.

Атрибут font-family компенсирует этот недостаток, позволяя не только задать точный шрифт, но и семейство, к которому он принадлежит. Таких семейств предусмотрено пять: serif (с засечками), sans-serif (без засечек), cursive (курсив), monospace (моноширинный) и fantasy (произвольный). Если нам неважно, каким именно шрифтом будет отображаться текст, лишь бы это был шрифт без засечек, мы можем создать такой стиль:

```
.sserif { font-family: sans-serif }
```

Если же нам хотелось бы, чтобы текст отображался шрифтом Verdana, но, на худой конец, сойдет любой шрифт без засечек, то пишем так:

```
.sserif { font-family: Verdana, sans-serif }
```

Как видим, варианты перечисляются через запятую, в порядке убывания приоритетности.

Напоминаем: название шрифта, состоящее из нескольких слов, как, например, Times New Roman, заключается в кавычки:

```
.sserif { font-family: "Times New Roman", serif }
```

Все атрибуты CSS, начинающиеся со слова font, как и атрибуты фонового изображения, можно описать с помощью общего атрибута font, перечислив их через пробел. Например, описание курсива с засечками размером 12 пунктов выглядит так:

```
.curs { font: serif 12 italic}
```

Атрибут text-transform (рис. 15.15) позволяет создать стиль, в котором бы все слова начинались с большой буквы (значение capitalize), выводились бы полностью прописными (uppercase) или строчными (lowercase) буквами. По умолчанию этот атрибут имеет значение none, т.е. прописные и строчные буквы в тексте сохраняются такими же, как и в коде.

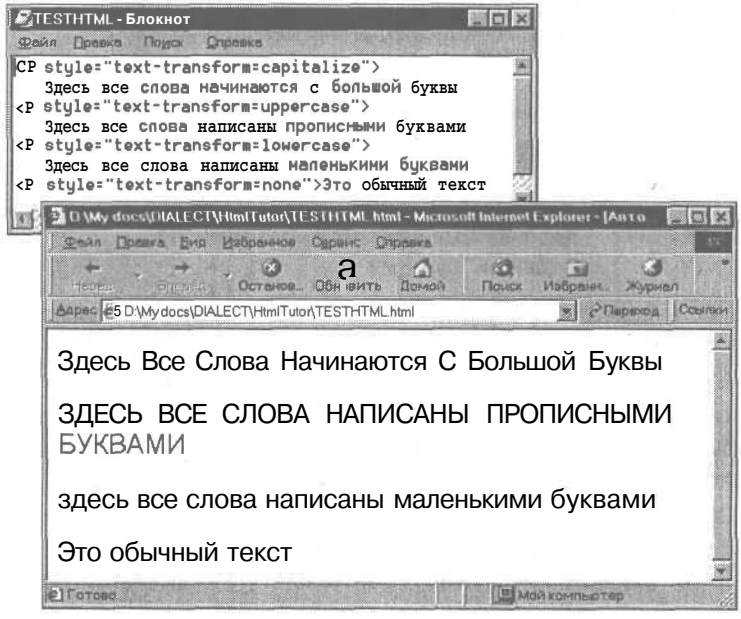


Рис. 15.15. Атрибут text-transform позволяет управлять выводом прописных и строчных букв

Режимы подчеркивания определяются атрибутом text-decoration (рис. 15.16): для обычного подчеркивания используется значение underline, для надчеркивания — overline и для зачеркнутого текста — line-through. Кроме того, атрибут text-decoration имеет также значение blink, позволяющее создавать мигающий текст. Впрочем, последним не рекомендуется злоупотреблять, так как от подобных эффектов у посетителей страницы быстро устают глаза.

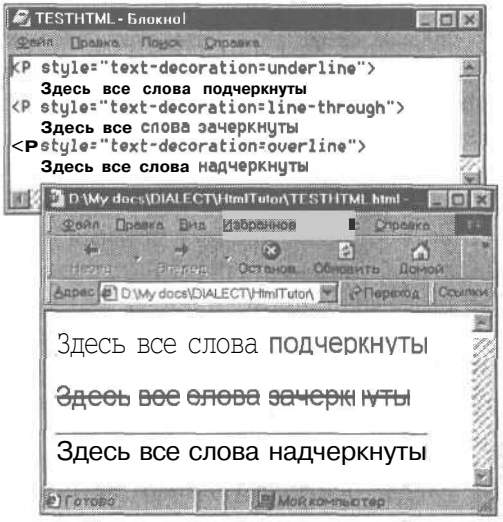


Рис. 15.16. Положение горизонтальной линии — подчеркивание, зачеркивание или надчеркивание — определяется атрибутом text-decoration

Наконец, атрибут letter-spacing позволяет регулировать расстояние между буквами (рис. 15.17).

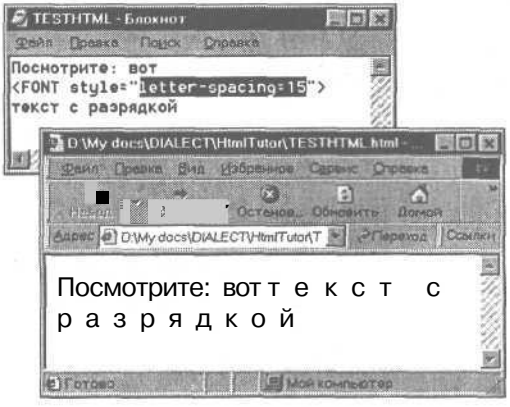


Рис. 15.17. Разрядка текста задается с помощью параметра letter-spacing

Обратите внимание на одну особенность, проиллюстрированную на рис. 15.17: в качестве "нейтрального" дескриптора использован <FONT>. Действительно, этот дескриптор идеально подходит для присваивания стиля фрагменту текста, не выделенному в самостоятельный абзац.

Атрибуты абзаца

Как и в предыдущих случаях, свойства абзаца, описываемые атрибутами CSS, повторяют и во многом дополняют свойства, описываемые средствами HTML. Так, атрибут text-align определяет выравнивание текста и принимает значения... правильно, left, right, center и justify. Зачем дублировать параметр align из "джентльменского набора" HTML? Действительно, польза от них в "чистом виде" минимальна. Но если на странице часто используется определенный комбинированный вид форматирования, то лучше создать стиль, в котором, наряду с другими параметрами, присутствует нужный тип выравнивания, чем каждый раз применять параметр align.

Однако атрибуты CSS позволяют описать и другие параметры абзаца. К таким параметрам относится, в первую очередь, междустрочный интервал. Дескрипторы и параметры HTML никак не влияют на это значение. Зато атрибут line-height позволяет задать его как в абсолютных значениях (пунктах), так и в относительных (процентах).

Теперь, имея в руках такой тонкий инструмент, нам придется быть внимательнее при подборе шрифта: слишком крупный шрифт при слишком маленьком междустрочном интервале приведет к тому, что строчки станут наползать друг на друга (рис. 15.18). Учитывая возможность регулирования размера шрифта в броузере, рекомендуется одновременно с определением междустрочного интервала строго указывать и размер шрифта.

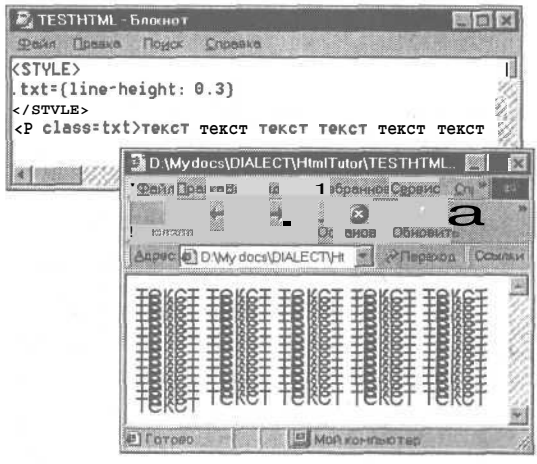


Рис. 15.18. Если междустрочный интервал слишком мал, строчки наползают друг на друга

Следующим свойством абзаца, о котором в стандартном HTML "забыли", является "красная строка". До сих пор мы довольствовались тем, чего можно добиться с помощью неразрывных пробелов и прозрачного GIF, "растянутого" на нужную длину. Однако вместо всех этих хитростей можно один раз создать нужный стиль CSS и затем использовать его по мере необходимости.

Атрибут, "отвечающий" за "красную строку", называется text-indent. Его значением служит величина отступа в сантиметрах или процентах. Кстати, она может быть и отрицательной. В этом случае вместо отступа получается выступ (рис. 15.19).

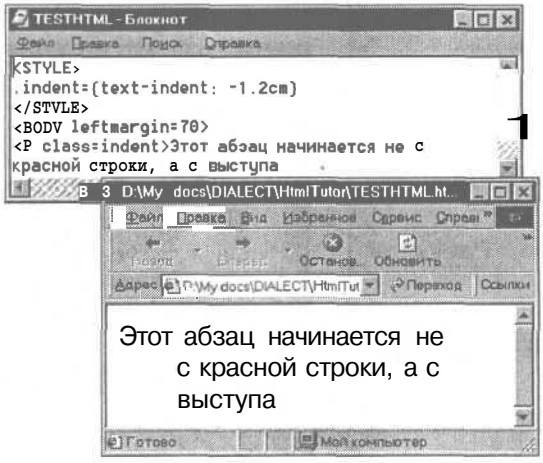


Рис. 15.19. С помощью атрибута text-indent можно создавать абзацы не только с отступом, но и с выступом

Гиперссылки

CSS позволяет переопределить не только стили гиперссылок, свойства которых описываются параметрами дескриптора <BODY>, но и создать новый стиль для ссылки, на которую в данный момент наведен указатель мыши. Эти стили имеют предопределенные имена и могут иметь в своем составе те же атрибуты, что и для обычного текста. Стиль a:link описывает вид непосещенной, стиль a:visited — посещенной, a:active — активной гиперссылки и a:hover — гиперссылки, на которую наведен указатель мыши. Например, если мы хотим, чтобы по умолчанию все ссылки на странице были темно-зелеными без подчеркивания, посещенные — серыми и тоже без подчеркивания, а при наведении на ссылку указателя мыши ее цвет менялся на коричневый, нужно создать такие стили:

```
a:link {color:darkgreen; text-decoration:none}
a:visited {color:gray; text-decoration:none}
a:hover {color:maroon; text-decoration:underline}
```

Отступы и рамки

Теперь поговорим об атрибутах, описывающих общие свойства большинства элементов Web-страницы, — отступах и рамках.

Всю область, занимаемую элементом, будь то ячейка таблицы, рисунок или абзац текста, можно разделить на пять зон (рис. 15.20): внутреннюю зону, внутренний отступ, рамку, внешний отступ и внешнюю зону.

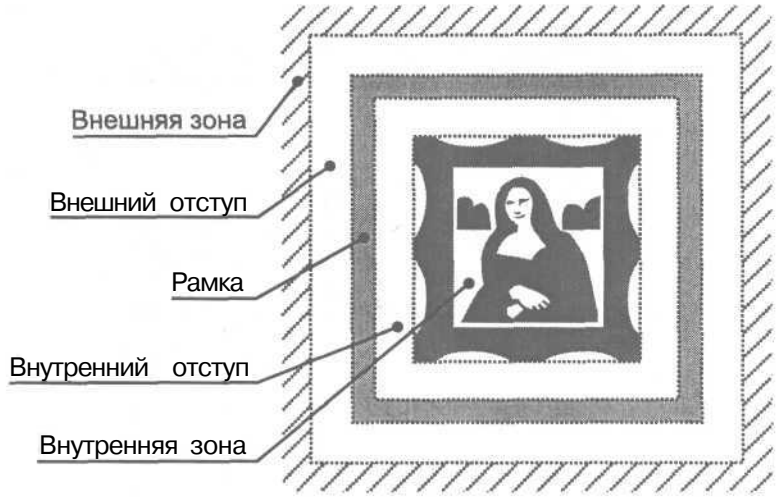


Рис. 15.20. Область, занимаемую элементом Web-страницы, можно разделить на пять зон

Во внешней зоне располагаются соседние элементы страницы. Например, это может быть текст, окружающий изображение, или ячейки таблицы, окружающие данную ячейку. Для этой зоны можно задать режим обтекания с помощью атрибута float, который принимает одно из трех значений: left (обтекание слева), right (обтекание справа) и none (отсутствие обтекания).

Между внешней зоной и рамкой элемента находится зона внешнего отступа. Ее ширина определяется атрибутом margin. Значением параметра margin в общем случае является последовательность из четырех чисел, разделенных пробелами. Эти числа соответствуют величине верхнего, правого, нижнего и левого отступов. По умолчанию отступы измеряются в пикселях (px), однако возможно задание и других единиц — сантиметров (cm) и процентов (%). Например, стиль

```
.indent {margin: 1 2 3 4}
```

обеспечивает отступ сверху на 1 пиксель, справа на 2 пикселя, снизу на 3 пикселя и слева на 4 пикселя.

Если вертикальные или горизонтальные отступы равны, можно указать только один из них. Например, запись

```
.indent {margin: 1 2}
```

означает, что верхний и нижний отступы составляют 1 пиксель, а правый и левый — 2 пикселя.

Наконец, если значением атрибута margin служит единственное число, это означает, что все отступы одинаковые.

Для того чтобы задать определенный внешний отступ, используются атрибуты margin-top, margin-right, margin-bottom и margin-left — для верхнего, правого, нижнего и левого отступов соответственно. Их значением служит число, определяющее величину отступа аналогично правилам для атрибута margin.

Подобным образом определяется величина внутреннего отступа, отделяющего рамку от самого элемента страницы. Только соответствующий атрибут называется padding, а атрибуты, определяющие величину отступов сверху, справа, снизу и слева, — padding-top, padding-right, padding-bottom и padding-left.

Для рамки элемента определены три свойства: толщина, цвет и стиль линии. Толщина и цвет задаются так же, как аналогичные значения других атрибутов. Что же касается стиля, то здесь возможны следующие варианты:

- none — рамка отсутствует;
- dotted — рамка состоит из точек;
- dashed — пунктирная рамка;
- solid — сплошная рамка;
- double — двойная рамка;
- groove, ridge, inset, outset — различные варианты "трехмерных" рамок.

Свойства рамки могут быть одинаковыми для всех четырех сторон, а могут быть и различными. В случае если они одинаковы, их можно присвоить общему атрибуту border. Например, двойная красная рамка толщиной 10 пикселей описывается таким стилем (рис. 15.21):

```
.red2 {border: 20cm red double}
```

Для описания свойств одной из сторон рамки используются атрибуты, где после слова border указывается соответствующая сторона — left, right, top или bottom. Например, для того чтобы описать те же свойства, но только для верхней границы элемента (рис. 15.22), используется стиль

```
.red2 {border-top: 20cm red double}
```

Если требуется описать только одно из свойств рамки, можно воспользоваться тем же атрибутом border, а можно использовать одну из его "производных": border-width для описания ширины, border-color для описания цвета или border-style для описания стиля. Наконец, если нужно описать свойство только одной из сторон рамки, используется "третья производная", где между словом border и наименованием свойства указывается нужная сторона — left, right, top или bottom. Так, например, для описания цвета верхней границы рамки используется атрибут border-top-color.

Наконец, внутренняя зона имеет всего два свойства — ширину и высоту, описываемые соответственно атрибутами width и height.

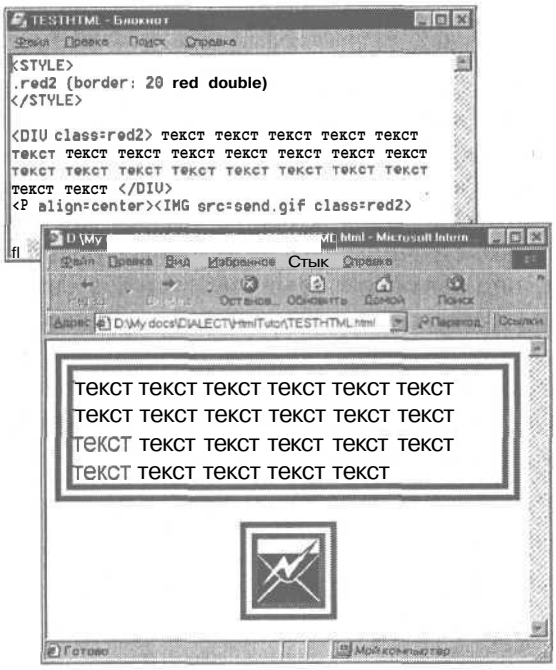


Рис. 15.21. Атрибут border позволяет описать свойства рамки различных элементов Web-страницы

Рис. 15.22. Для того чтобы описать свойства только одной из сторон рамки, к имени border добавляется название этой стороны

Резюме

Каскадные таблицы стилей — мощное средство, значительно расширяющее возможности стандартного HTML. Стили, создаваемые посредством CSS, позволяют создать единый дизайн страниц, экономят труд и размер кода.

С помощью каскадных таблиц стилей можно как уточнять форматирование, определяемое дескриптором в данном месте документа, так и создавать новые стили, которые можно использовать многократно. В первом случае стиль описывается внутри дескриптора с помощью параметра style, во втором — стилю присваивается имя и он размещается внутри дескриптора <STYLE>. При этом стиль может создаваться как подкласс определенного дескриптора (и тогда впереди имени стиля ставится имя этого дескриптора), так как самостоятельный стиль.

Описание стиля состоит из перечня аргументов и их значений, разделенных точками с запятой и заключенных в фигурные скобки. Между аргументом и его значением ставится двоеточие.

Благодаря каскадным таблицам стилей можно описывать такие отсутствующие в стандартном HTML элементы форматирования, как фон отдельных символов, точный размер шрифта, разрядка букв, "красная строка", подсвечивание указателя мыши при наведении на гиперссылку и многое другое.

Тесты

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а) <I class="font-size: 18; color: white; background-color: black">
   Это текст, к которому применен стиль.</I>

   б) <I style="font-size: 18; color: white; background-color: black">
   Это текст, к которому применен стиль.</I>

   в) <I.style="font-size: 18; color: white; background-color: black">
   Это текст, к которому применен стиль.</I>

   г) <I class=inverse>
   Это текст, к которому применен стиль.</I>

2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а)
   ```
   <STYLE>
   I.inverse="font-size: 18; color: white; background-color: black"
   </STYLE>
   ```

   б)
   ```
   <STYLE>
   I.inverse:"font-size: 18; color: white; background-color: black"
   </STYLE>
   ```

   в)
   ```
   <STYLE>
   inverse="font-size: 18; color: white; background-color: black"
   </STYLE>
   ```

г) <STYLE>
I.inverse={font-size: 18; color: white; background-color: black"}
</STYLE>

д) <STYLE>
inverse "font-size: 18; color: white; background-color: black"
</STYLE>

е) <STYLE>
.inverse {font-size: 18; color: white; background-color: black"}
</STYLE>

ж) <STYLE>
I.inverse {font-size: 18; color: white; background-color: black"}
</STYLE>

3. Требуется, чтобы весь полужирный курсив в тексте выводился красным цветом. Каким из предлагаемых ниже стилей это можно сделать?

а) U I {color=red}

б) U {font-style=italic; color=red}

в) I {font-style=bold; color=red}

г) I {font-weight=bold; color=red}

д) I.bold {color=red}

е) U.italic {color=red}

ж) I.bold {font-weight=bold; color=red}

з) DIV {font-style=italic; font-weight=bold; color=red}

Глава 16

Понятие событий

В этой главе...

- Понятие динамического HTML
- Виртуальные события
- События мыши
- События клавиатуры
- События форм
- События страницы

Понятие динамического HTML

Как вы могли убедиться, ознакомившись с предыдущими главами этой книги, "классический" HTML — это средство *форматирования* электронных документов. Язык разметки гипертекста изначально не рассчитан на выполнение каких-либо активных *действий*. Этим HTML радикально отличается от языков программирования, выполняющих последовательность команд.

Но жизнь требует от Web-страниц более широких возможностей. В первую очередь они заключаются в реакции на определенные действия пользователя — переход к тому или иному элементу окна, манипуляции мышью и т.д. Так появился динамический HTML (DHTML), представляющий собой сочетание обычного HTML и языка сценариев JavaScript.

Более сложные вещи, такие как обработка и хранение информации, получаемой от посетителя страницы, осуществляются с помощью других языков программирования, таких как PHP и Perl. См. также Зандстра М. *Освой самостоятельно PHP4 за 24 часа*. К.: Вильяме, 2001; Пирс К. *Освой самостоятельно Perl за 24 часа*. К.: Вильяме, 2000.

Виртуальные события

Событием называют некое действие, произошедшее на странице, в ответ на которое требуется выполнить какие-то операции — изменить форматирование страницы, передать данные на сервер и т.п. Обычно причиной этого действия является посетитель, который что-то ввел, выбрал или на чем-то щелкнул.

Любое событие имеет "хозяина" - элемент Web-страницы, с которым это событие происходит. Таким "хозяином" может быть, например гиперссылка, а событиями — наведение на нее указателя мыши или щелчок на ней. Кроме того, каждое событие имеет зарезервированное имя, одновременно являющееся параметром дескриптора. Этот параметр определяет, какое именно событие нужно обрабатывать — движение мыши, ввод текста, загрузку страницы и т.п. Значением параметра является действие, которое нужно выполнить:

<ДЕСКРИПТОР при_событии_N="описание действия"> содержимое дескриптора </ДЕСКРИПТОР>

Например, на рис. 16.1 показан код, реализующий "виртуальный светофор": при наведении указателя мыши ячейки таблицы "загораются" красным, желтым и зеленым цветом.

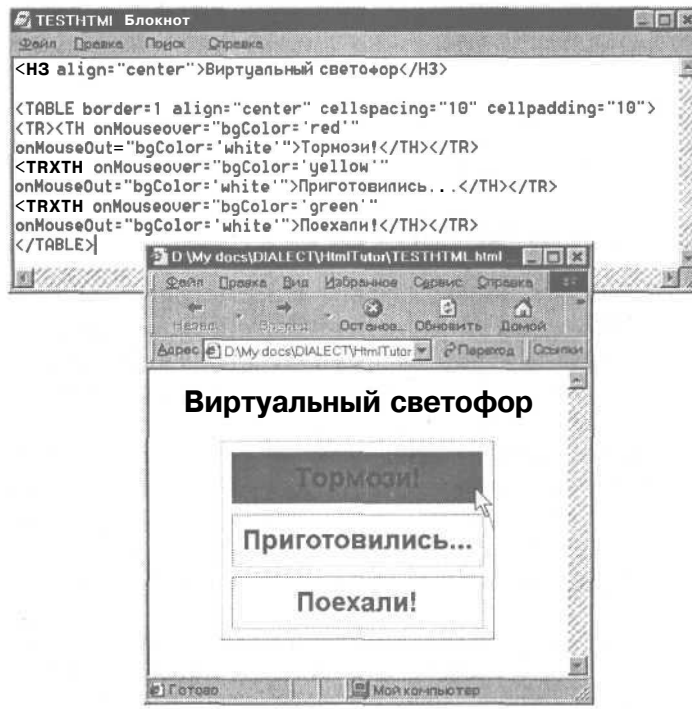


Рис. 16.1. "Виртуальный светофор": когда указатель мыши попадает на ячейку таблицы, "срабатывает" код JavaScript, и ячейка окрашивается в соответствующий цвет

В примере на рис. 16.1 использован код на языке JavaScript. Он вам незнаком. Однако, зная HTML, вы вполне можете понять логику использованных выражений и догадаться, какие действия должны выполняться.

Все события, обрабатываемые броузером, можно разделить на группы, в зависимости от "виновника": мыши, клавиатуры, элемента формы или всей HTML-страницы.

События мыши

Эта группа событий, пожалуй, самая многочисленная. Вот парадокс: простейшее устройство из шарика и двух кнопок приводит к появлению такого множества событий: наведение указателя мыши на объект страницы, нажатие кнопки, одинарный и двойной щелчок, отпускание кнопки, движение указателя по объекту и смещение указателя за пределы объекта. Рассмотрим эти события более подробно на примерах.

Самое первое событие, связанное с мышью, которое приходит в голову, — это простой щелчок. Такое событие в HTML носит имя onclick (от английского *on click* — "при щелчке"). Например, при щелчке на гиперссылке происходит переход на другую HTML-страницу. Однако для обработки такого события, как мы уже знаем, достаточно обычного HTML. Другое дело, если мы захотим, чтобы одновременно с переходом по ссылке изменялся вид самой ссылки, скажем одна картинка заменялась другой. Для этого можно использовать код, показанный на рис. 16.2.

Названия событий — обычные параметры дескрипторов HTML. Поэтому использовать в них строчные и прописные буквы можно произвольно. В частности, удобно отмечать прописными буквами начало английских слов. Однако значениями этих параметров часто является код JavaScript, где регистр букв имеет значение. Поэтому будьте внимательны: строго соблюдайте регистр букв в коде, который стоит после знака равенства.

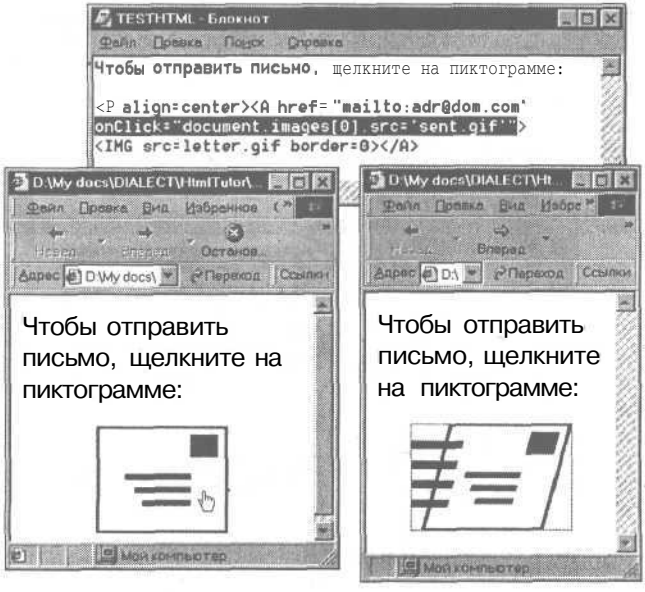


Рис. 16.2. При щелчке на ссылке происходят две вещи: открывается бланк письма и меняется картинка

В примере на рис. 16.2 для замены картинки использован код JavaScript. В JavaScript все однородные объекты на странице — картинки, таблицы, формы, гиперссылки — объединены в группы, именуемые *массивами*.

Массив изображений называется images. Каждому элементу массива присвоен собственный порядковый номер, начиная от нуля: первая по порядку картинка на странице имеет номер 0, вторая — 1 и т.д. Для обращения к определенному изображению на странице используется запись images [N], где N — порядковый номер этого изображения. Для обращения к определенному свойству данного объекта используется имя этого свойства. Между именем свойства и объекта ставится разделитель — точка. Например, для того чтобы изменить источник первого изображения на странице, используется запись наподобие `images[0].src="source.gif"`. Наконец, объекты страницы могут "вкладываться" друг в друга, как матрешки. Например, кнопка является элементом формы, а форма — элементом всего электронного документа. Для обращения к таким "подобъектам" используется та же запись имен через точку. Поэтому полный код, позволяющий изменить источник первого изображения на странице, выглядит так: `document.images[0].src='sent.gif'`. В JavaScript много стандартных объектов, описывающих основные элементы HTML-страницы.

Событие, которое должно происходить после двойного щелчка, описывается с помощью параметра onDblClick (от английского *on double click* — "по двойному щелчку"). На рис. 16.3 предыдущий пример дополнен еще одним действием: после двойного щелчка содержимое окна исчезает, и появляется забавная надпись.

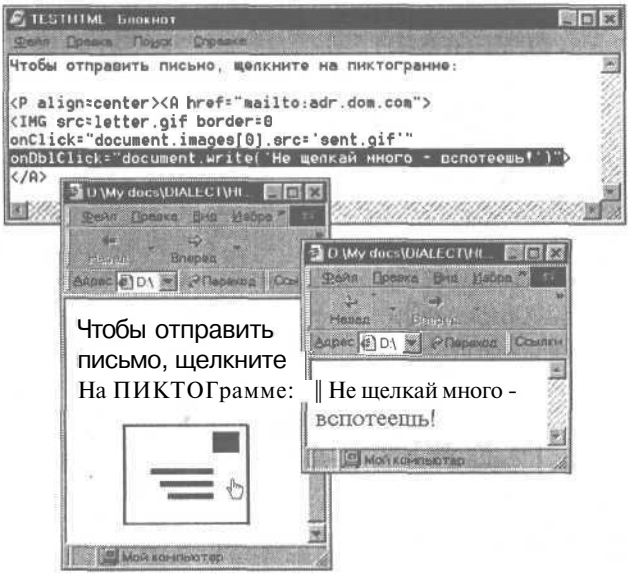


Рис. 16.3. После двойного щелчка содержимое окна меняется

В примере, представленном на рис. 16.3, использован код JavaScript с функцией write. Эта функция позволяет выводить на экран произвольный текст, в том числе и с дескрипторами HTML.

Обратите внимание: события можно использовать не только в дескрипторе гиперссылки <A>, но и в других дескрипторах HTML.

В Internet часто встречаются страницы, на которых кнопки в виде пиктограмм меняют вид в момент щелчка, а после него возвращаются в исходное состояние. Для этого используются события onMouseDown и onMouseUp. Первое из них описывает состояние, когда кнопка мыши нажата, но еще не отпущена, так что щелчок еще не состоялся (ее название происходит от английского *on mouse down* — "при нажатии кнопки мыши"). Второе событие описывает состояние, когда кнопка мыши уже отпущена (ее название происходит от английского *on mouse up* — "при отпускании кнопки мыши"). Пример использования этих событий приведен на рис. 16.4: при нажатии кнопки мыши изображение закрытого замка заменяется на изображение открытого; при отпускании кнопки замок снова "закрывается".

Рис. 16.4. События onMouseDown и onMouseUp позволяют "открывать" и "запирать" замок путем нажатия и отпускания кнопки мыши

Наконец, для того чтобы страница "сработала", не обязательно щелкать кнопкой мыши. Можно написать код так, чтобы было достаточно простого движения указателя. Для этого используются события onMouseOver, onMouseMove и onMouseOut. Событие onMouseOver описывает ситуацию, когда указатель мыши "наползает" на объект, onMouseMove — когда движется по объекту и onMouseOut — когда "сползает" с объекта. Пример использования этих событий показан на рис. 16.1.

Все это касается только левой кнопки мыши. Правая кнопка не обрабатывается, она используется броузером для вывода контекстного меню.

События клавиатуры

События, связанные с клавиатурой и распознаваемые динамическим HTML, аналогичны событиям мыши. Однако их меньше: только нажатие клавиши onKeyPress, движение клавиши вниз `onKeyDown` и вверх `onKeyUp`. На рис. 16.5 представлен пример страницы, позволяющей определить код нажатой клавиши.

Код нажатой клавиши сохраняется в объекте JavaScript `window.event.keyCode`. Функция `window.alert` используется для вывода стандартного окна предупреждения. Значением этой функции является текстовая строка. В JavaScript можно составлять строки из фрагментов с помощью выражений вида `строка1+строка2+строка3`.

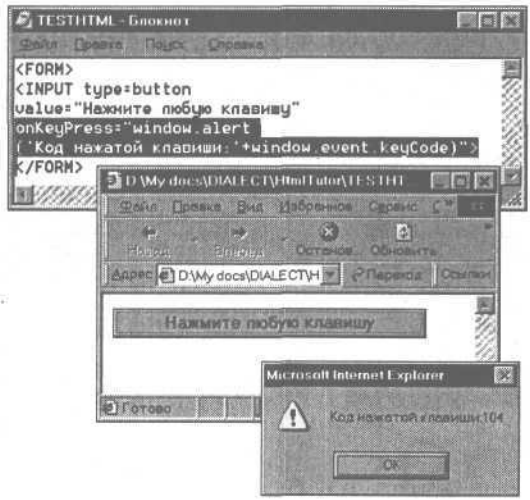


Рис. 16.5. Полезная страничка: позволяет узнать код нажатой клавиши

События форм

Если вы сами попробовали, как "работает" предыдущий пример, то должны были заметить, что он "срабатывает" не всегда, а только тогда, когда кнопка выделена. Такое выделение еще называют *фокусом ввода*. Для того чтобы элемент формы стал активен (получил фокус), щелкните на нем кнопкой мыши. Если форма состоит из нескольких элементов, то для перехода между ними (передачи фокуса) можно использовать клавишу <Tab>. Разумеется, вводить что-то в элемент формы можно только тогда, когда он активен, т.е. когда он имеет фокус ввода.

Событие получения фокуса элементом формы называется onFocus, потери фокуса — `onBlur`. На рис. 16.6 показан улучшенный вариант рассмотренного выше примера: когда кнопка не активна, надпись приглашает щелкнуть на ней мышью; когда же кнопка получает фокус, надпись меняется.

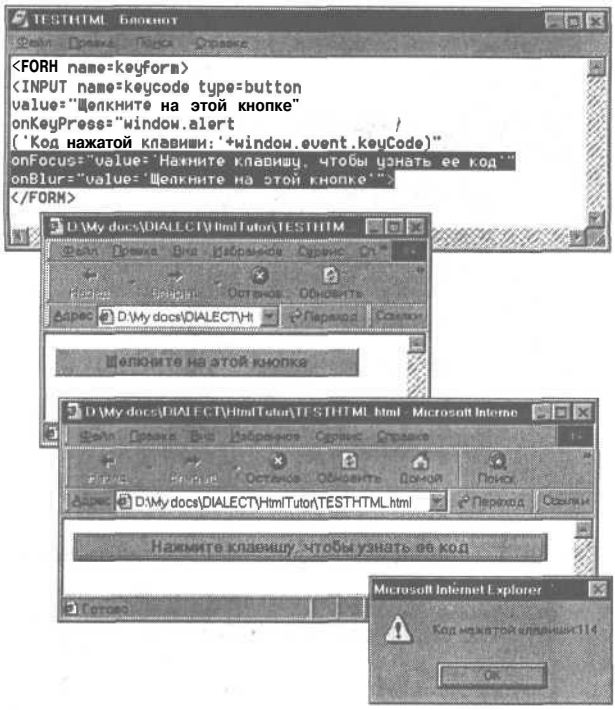


Рис. 16.6. Улучшенный вариант предыдущего примера с учетом активности и неактивности кнопки

Событие onChange означает, что данный элемент формы не просто потерял фокус, а при этом еще и претерпел изменения. Пример использования такого события показан на рис. 16.7: в ответ на неправильный пароль выводится окно с предупреждением.

Пример, показанный на рис. 16.7, является только иллюстрацией применения события onChange. Его нельзя рассматривать как хороший способ парольной защиты. Имейте в виду, что код Web-страницы является открытым, поэтому любой, прочитавший его, сможет узнать истинный пароль.

В примере на рис. 16.7 проиллюстрировано еще одно свойство языка JavaScript: условный оператор if. Выражение, стоящее в скобках, является условием, в случае истинности которого выполняется выражение, следующее за скобками.

Событие onSelect "срабатывает" тогда, когда пользователь выделяет в поле ввода некий текст. При этом переходить к другому элементу формы вовсе не обязательно. На рис. 16.8 показана страница, где, в зависимости от того, что пользователь намерен делать с текстом (вводить или редактировать выделенный фрагмент), меняется пиктограмма.

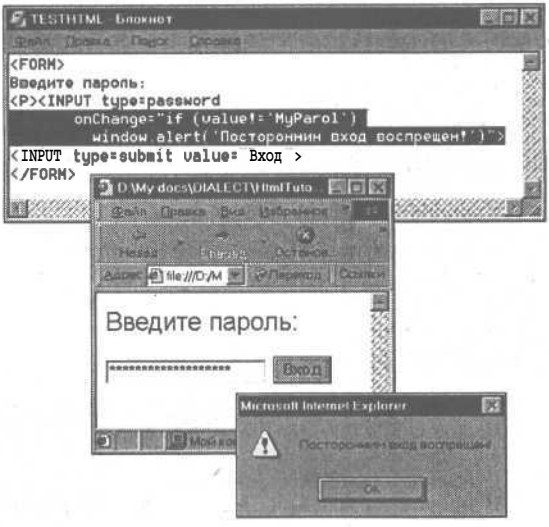


Рис. 16.7. Пример использования события onChange

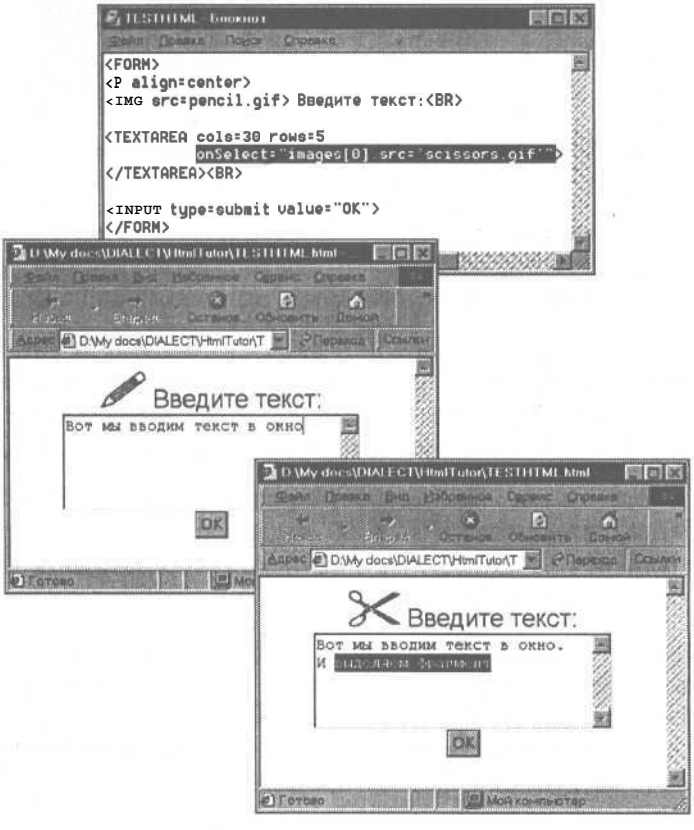


Рис. 16.8. Событие onSelect позволяет определить, выделен ли фрагмент вводимого текста

Наконец, момент, когда пользователь завершил работу с формой и щелкнул на кнопке ее отправки, соответствует событию onSubmit. Отправленные данные могут обрабатываться по-разному. Мы же, чтобы продемонстрировать "работу" этого события, приведем простой пример: по окончании ввода данных выводится информационное окно (рис. 16.9).

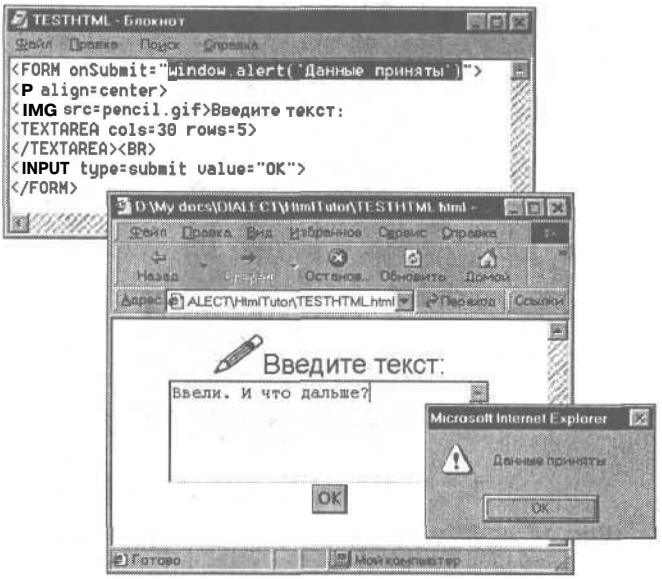


Рис. 16.9. По окончании ввода данных появляется информационное окно

События страницы

С функционированием страницы в целом связаны следующие события: загрузка и выгрузка страницы, а также прерванная загрузка и ошибка загрузки. Событие загрузки называется onLoad и часто используется для временной замены больших и долго загружающихся объектов меньшими. Таким образом, например, посетитель страницы, вместо того чтобы ожидать окончания загрузки большого изображения, может первое время довольствоваться его уменьшенным вариантом (рис. 16.10)

В коде на рис. 16.10 использованы последовательно две команды JavaScript. Разделителем между ними служит точка с запятой.

О том, что посетитель перешел с данной страницы на другую — по ссылке, ввел новый URL в строке адреса или перезагрузил эту же страницу с помощью кнопки Обновить (Refresh), — свидетельствует событие onUnload. Использовать эту информацию можно самыми различными способами. Самый простой из них — вывод предупреждающего сообщения (рис. 16.11).

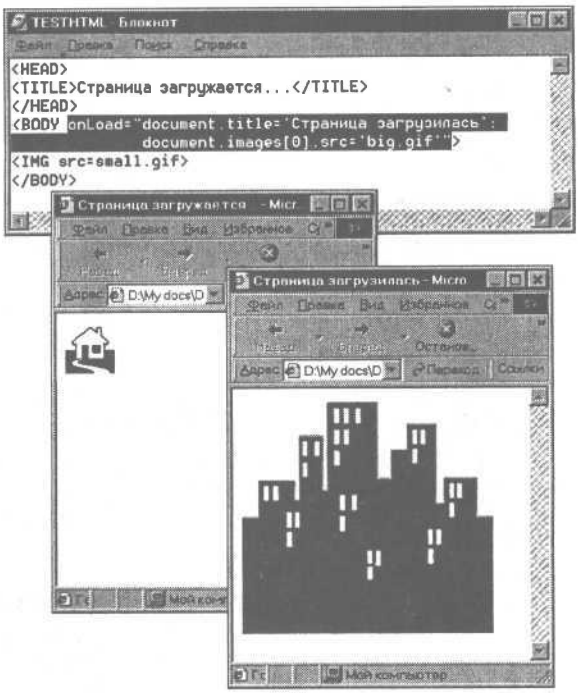


Рис. 16.10. До окончания загрузки страницы большое изображение можно временно заменить маленьким

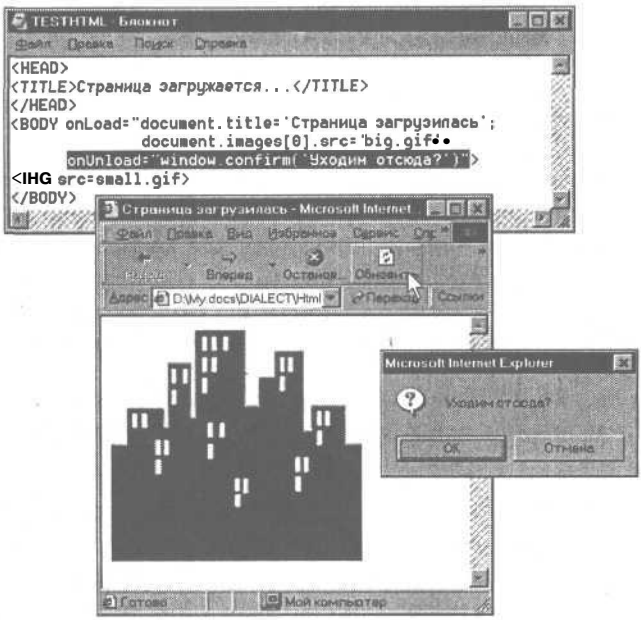


Рис. 16.11. Событие `onUnload` "срабатывает" при выгрузке страницы

Аналогично "работают" событие ошибки загрузки onError и прерывания загрузки onAbort. Оба они означают несостоявшуюся загрузку, только в первом случае загрузка прерывается программно, во втором — посетителем страницы.

Эта последняя глава книги — только краткое введение в другую интереснейшую область — JavaScript и динамический HTML. Если же вы захотите глубже изучить этот язык, рекомендую книгу Монкур М. "Освой самостоятельно JavaScript за 24 часа" М.: Вильяме, 2001.

Резюме

HTML рассчитан только на статическое отображение гипертекста. Для создания динамично меняющихся страниц используются другие средства, дополняющие и расширяющие этот язык. Одним из таких средств является, в частности язык сценариев JavaScript.

"Сигналами" к действиям служат *события* — одинарный или двойной щелчок, движение указателя мыши, нажатие клавиши, передача фокуса элементу формы и т.д. Каждому событию соответствует свой параметр. Этот параметр указывается в дескрипторе, описывающем элемент страницы, с которым происходит данное событие. Значением параметра является строка, представляющая собой код на языке сценариев.

Тесты

1. В каком случае вывод сообщения произойдет по щелчку мыши?
   а) <P onMouseDown="window.alert('Щелк!')"> Щелкните здесь
   б) <P onMouseClick="window.alert('Щелк!')"> Щелкните здесь
   в) <P onMouseUp="window.alert('Щелк!')"> Щелкните здесь
   г) <P onMouseOver="window.alert('Щелк!')"> Щелкните здесь
2. В каком случае вывод сообщения произойдет при наведении указателя мыши на изображение?
   а) <IMG src=big.gif onMouseDown="window.alert('Щелк!')">
   б) <IMG src=big.gif onMouseOut="window.alert('Щелк!')">
   в) <IMG src=big.gif onMouseUp="window.alert('Щелк!')">
   г) <IMG src=big.gif onMouseOver="window.alert('Щелк!')">
3. В каком случае вывод сообщения произойдет при удалении указателя мыши с изображения?
   а) <IMG src=big.gif onMouseDown="window.alert('Щелк!')">
   б) <IMG src=big.gif onMouseOut="window.alert('Щелк!')">
   в) <IMG src=big.gif onMouseUp="window.alert('Щелк!')">
   г) <IMG src=big.gif onMouseOver="window.alert('Щелк!')">

4. В состав страницы входит следующий код:

   <BODY onUpload="window.alert('Страница выгружена')">.

   В каком или в каких случаях будет появляться предупреждение?

   а) Когда посетитель щелкнет на кнопке Назад (Back).

   б) Когда посетитель щелкнет на кнопке Вперед (Forward).

   в) Когда посетитель щелкнет на кнопке Обновить (Refresh).

   г) Когда посетитель щелкнет на кнопке Остановить (Stop).

5. В состав страницы входит следующий код:

   <BODY onLoad="window.alert('Страница выгружена')">.

   В каком или в каких случаях будет появляться предупреждение?

   а) Когда посетитель щелкнет на кнопке Назад (Back).

   б) Когда посетитель щелкнет на кнопке Вперед (Forward).

   в) Когда посетитель щелкнет на кнопке Обновить (Refresh).

   г) Когда посетитель щелкнет на кнопке Остановить (Stop).

Приложение А

Ответы на вопросы тестов

Введение

1. Что из нижеперечисленного является броузером?

   а) Microsoft Word — неправильно. Это текстовый процессор. Хотя он и позволяет просматривать Web-страницы, но это только одна из функций данного приложения.

   б) Microsoft Internet Explorer — правильно.

   в) Microsoft FrontPage Express — правильно.

   г) HomeSite — неправильно. Это редактор HTML-страниц.

   д) Netscape Navigator — правильно.

   е) Opera — правильно.

   ж) Mozilla — правильно.

   з) Notepad — неправильно. Это простейший текстовый редактор, с помощью которого можно написать код HTML-страницы.

   и) PhotoShop — неправильно. Это графический процессор, с помощью которого можно обрабатывать изображения для HTML-страниц.

   к) Google.com — неправильно. Это поисковый сервер Internet.

2. Что из нижеперечисленного используется для просмотра HTML-страницы?

   а) Microsoft Word — правильно (см. тест 1 этой главы).

   б) Microsoft Internet Explorer — правильно.

   в) Microsoft FrontPage Express — правильно. Это визуальный редактор HTML-страниц, позволяющий в том числе увидеть, как будет выглядеть страница в броузере.

   г) HomeSite — правильно. Это редактор HTML-страниц, позволяющий в том числе увидеть, как будет выглядеть страница в броузере.

   д) Netscape Navigator — правильно.

   е) Opera — правильно.

   ж) Mozilla — правильно.

   з) Notepad — неправильно. В этом редакторе можно просмотреть только *код* HTML-страницы, но не то, как она должна выглядеть для посетителя.

   и) Windows - неправильно. Windows — это семейство операционных систем, под управлением которых работают различные приложения, в том числе и средства для просмотра HTML-страниц.

к) Linux — неправильно (см. ответ (и) этого теста).

л) PhotoShop — неправильно (см. тест 1 этой главы).

м) Google.com — неправильно. С помощью этого поискового сервера можно *найти* нужные страницы в Internet, но отображаются эти страницы, как впрочем и страницы Google.com, в броузере.

3. Что из нижеперечисленного используется для просмотра кода HTML-страницы?

а) Microsoft Word — правильно.

б) Microsoft Internet Explorer — неправильно. По команде Вид ⇨ В виде HTML из Internet Explorer открывается Notepad или другой HTML-редактор, который используется по умолчанию.

в) Microsoft FrontPage Express — правильно.

г) HomeSite — правильно.

д) Netscape Navigator — неправильно (см. пункт (б) этого теста).

е) Opera — неправильно (см. пункт (б) этого теста).

ж) Mozilla — неправильно (см. пункт (б) этого теста).

з) Notepad — правильно.

и) Windows — неправильно (см. тест 2 этой главы).

к) Linux — неправильно (см. тест 2 этой главы).

л) PhotoShop — неправильно.

м) Google.com — неправильно.

Глава 1

1. Какие из следующих фрагментов HTML-кода содержат ошибку?

а) Зачем КРИЧАТЬ, когда можно мягко <I>расставить акценты</I>? — ошибок нет. Этот код будет выглядеть так: Зачем КРИЧАТЬ, когда можно мягко *расставить акценты*?

б) Текст курсивом: <I></I> — формально ошибок нет. Но конструкция <I></I> бессмысленна, так как внутри нее ничего нет.

в) Многие дескрипторы принадлежат к <\I>контейнерному типу<I> — ошибка. Для того чтобы слова *контейнерному типу* выводились курсивом, нужно вначале поставить открывающий тег <I>, а в конце — закрывающий <\I>. Иначе дескриптор <\I> будет проигнорирован, а весь текст, начиная от <I> и до конца документа (или до ближайшего <\I>), будет выводиться курсивом.

2. Какие из следующих фрагментов HTML-кода не содержат ошибки?

а) Увлекаться дескриптором <U> не рекомендуется — ошибка. Дескриптор <U> — контейнерный и должен иметь закрывающую пару </U>. Иначе весь текст, начиная от <U> и до конца (или до ближайшего дескриптора </U>), будет выводиться с подчеркиванием.

б) С помощью вложенных дескрипторов «I» и «B» достигается эффект полужирного курсива — ошибки нет. Впрочем, форматирования тоже нет. Для того

чтобы оно было, дескрипторы должны заключаться не в угловые кавычки, а между знаками < и >.

в) Контейнерные дескрипторы бывают вложенными, например: <I><U></U></I> — формально ошибки нет. Однако ставить в код контейнерные дескрипторы, внутри которых отсутствует текст, бессмысленно.

г) Разметка гипертекста средствами HTML производится путем вставки в текст <I><U>дескрипторов</I></U> — ошибки нет. Текст в окне броузера будет выглядеть так: Разметка гипертекста средствами HTML производится путем вставки в текст *дескрипторов*.

3. Как будет выглядеть в окне броузера следующий фрагмент HTML-страницы: Для того чтобы броузер воспринимал содержимое файла как HTML-код, этот файл должен иметь расширение <U><I>.htm</I> или <B>.html</B></U>

а) Для того чтобы броузер воспринимал содержимое файла как HTML-код, этот файл должен иметь расширение *.htm* или **.html** — неправильно.

б) Для того чтобы броузер воспринимал содержимое файла как HTML-код, этот файл должен иметь расширение *.htm* или **.html** — правильно.

в) Для того чтобы броузер воспринимал содержимое файла как HTML-код, этот файл должен иметь расширение ***.htm или .html*** — неправильно.

Глава 2

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <P> Для того чтобы перейти на новую строку, используется тег <P> — формально ошибок нет. Второй дескриптор <P> приведет к тому, что следующий текст будет выводиться с новой строки.

б) <P> Для того чтобы перейти на новую строку, используется тег <P></P> — формально ошибок нет. Второй дескриптор <P> приведет к переходу на новую строку. Закрывающий дескриптор </P> не обязателен.

в) <P> Для того чтобы перейти на новую строку, используется тег «P»</P> — ошибок нет.

г) <R> Для того чтобы перейти на новую строку, используется тег «P»</R> — ошибка: дескриптора <R> не существует.

д) «P» Для того чтобы перейти на новую строку, используется тег P — формально ошибки нет. Но если под «P» подразумевался дескриптор абзаца, его нужно заключать не в кавычки, а в угловые скобки: <P>

2. В каком случае абзац будет выровнен по левому краю:

а) <P align = center> Этот абзац выровнен по левому краю? — нет, этот абзац выровнен по центру.

б) <P align = left> Этот абзац выровнен по левому краю? — да, этот абзац выровнен по левому краю. *

в) <P align = right> Этот абзац выровнен по левому краю? — нет, этот абзац выровнен по правому краю.

г) <P align = justify> Этот абзац выровнен по левому краю? — нет, этот абзац выровнен по ширине.

д) <P> Этот абзац выровнен по левому краю? — да. Когда параметр align не указан, то по умолчанию абзац выравнивается по левому краю.

е) <P align> Этот абзац выровнен по левому краю? — да. Поскольку значение параметра align не указано, броузер его игнорирует и по умолчанию устанавливает выравнивание по левому краю.

3. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <P align=center> Этот текст выровнен по центру — ошибок нет.

б) <P ALIGN=CENTER> Этот текст выровнен по центру — и здесь ошибок нет: броузеру безразлично, в каком регистре набраны дескрипторы и их параметры.

в) <P_align=center> Этот текст выровнен по центру — ошибка: между дескриптором и параметром должен стоять пробел, а не знак подчеркивания.

г) <P
align = center> Этот текст выровнен по центру———ошибок нет. Между дескриптором и параметрами может стоять как пробел, так и знак перехода на новую строку.

д) <P aLiGn=cEnTeR> Этот текст выровнен по центру — ошибок нет. Дескрипторы и параметры могут быть набраны в любом регистре.

е) <P center = align> Этот текст выровнен по центру — ошибка: параметр должен стоять слева от знака равенства, значение — справа, а не наоборот. Броузер проигнорирует эту ошибку, абзац будет выровнен по левому краю — так, как предусмотрено по умолчанию.

4. Как будет выглядеть в окне броузера следующий фрагмент HTML-страницы?
<p align = right> Где переход на новую строку:
здесь?
 Или здесь?
<P> А может быть, и здесь тоже?

Правильный вариант — (б).

Варианту (а) соответствует код

```
<p align = right> Где переход на новую строку:
здесь? <P> Или здесь?
<P> А может быть, и здесь тоже?
```

Варианту (в) соответствует код

```
<p align = right> Где переход на новую строку:
здесь? <P align = right> Или здесь?
<P align = right> А может быть, и здесь тоже?
```

Глава 3

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) Есть три типа кавычек: «елочки», "верхние лапки" и „нижние лапки". — Ошибка. Символы, не входящие в стандартный набор ASCII, не всегда

правильно интерпретируются броузером. Для того чтобы они отображались правильно, необходимо использовать esc-последовательности или мнемонические имена.

б) Есть три типа кавычек: <<елочки>>, "верхние лапки" и ,нижние лапки'. — Ошибка. Эти символы, за исключением нижней одиночной кавычки, вообще не являются кавычками. Это знаки "больше" и "меньше", символы дюйма и минут. Их использование в данном случае противоречит правилам орфографии и полиграфическим канонам.

в) Есть три типа кавычек: «елочки», “верхние лапки” и „нижние лапки”. — Правильно. Для отображения кавычек в коде использованы их мнемонические имена.

г) Есть три типа кавычек: &171;елочки&187;, &8220;верхние лапки&8221; и &8222;нижние лапки&8221;. — Правильно. Для отображения кавычек в коде использованы соответствующие esc-последовательности.

д) Есть три типа кавычек: «елочки», “верхние лапки” и „нижние лапки”. -- Правильно. Для отображения кавычек в коде использованы соответствующие esc-последовательности и мнемоническое имя. Их можно комбинировать.

е) Есть три типа кавычек: »елочки», “верхние лапки” и „нижние лапки”. -- Неправильно. » — Мнемоническое имя правой кавычки, а на этом месте должна стоять левая кавычка. Ее мнемоническое имя — «.

2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) HTML-страница не должна содержать ошибок - в том числе орфографических. И ее объем не должен превышать 5-10 Кбайт. — С точки зрения кода — правильно, с точки зрения орфографии — нет. Кроме того, следует позаботиться о правильной расстановке переносов.

б) HTML-страница не должна содержать ошибок ‐ в том числе орфографических. И ее объем не должен превышать 5‐10 Кбайт. — Неправильно. Не существует символа с мнемоническим именем ‐.

в) HTML-страница не должна содержать ошибок – в том числе орфографических. И ее объем не должен превышать 5—10 Кбайт. — С точки зрения кода — правильно, с точки зрения орфографии — все должно быть наоборот: между цифрами нужно использовать короткое тире, а между словами — длинное.

г) HTML-страница не должна содержать ошибок — в том числе орфографических. И ее объем не должен превышать 5–10 Кбайт. В принципе все правильно. Но, кроме тире, следует также позаботиться о неразрывных пробелах в тех местах, где, по правилам орфографии, переход на следующую строку запрещен.

д) HTML-страница не должна содержать ошибок—в том числе орфографических. И ее объем не должен превышать 5 Sndash; 10 Кбайт. — С точки зре-

ния кода — правильно, с точки зрения орфографии — нет: между длинным тире и словами должны стоять пробелы, а между коротким тире и цифрами их не должно быть.

е) HTML-страница не должна содержать ошибок — в том числе орфографических. И ее объем не должен превышать 5–10 Кбайт. — Все тире и неразрывные пробелы расставлены правильно.

3. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <P align=center> По-моему, весь этот текст должен выводиться в одной строке</P> — код правильный, но утверждение неверно: если окно броузера окажется *уже*, чем эта строка, она будет разорвана.

б) <NOBR align=center> По-моему, весь этот текст должен выводиться в одной строке</NOBR> — неправильно: у дескриптора NOBR нет параметров. Код align=center будет проигнорирован.

в) По-моему, весь этот текст должен выводиться в одной строке. — Код правильный, но утверждение неверно: конечно, неразрывные пробелы сделают свое дело, но строка может быть разорвана после дефиса.

г) <NBSP> По-моему, весь этот текст должен выводиться в одной строке</NBSP> — правильно: действительно, весь этот текст будет выводиться в одной строке.

4. В каких точках (A, B, C, D, E) следующего фрагмента возможен переход на новую строку?

<NOBR> Где пере<WBR>(а)ход на <WBR>(б) новую стро-(в)ку: здесь?<WBR>(г)Или здесь? <P>(д)А может быть, и здесь тоже?</NOBR>

Переход возможен во всех точках, кроме *а:* в середине слова дескриптор <WBR> не "срабатывает".

Глава 4

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <FONT> Этот текст набран другим шрифтом</FONT> — формально ошибок нет. Но дескриптор <FONT> без параметров, сам по себе, не меняет вида текста.

б) <FONT face=Arial Cyr> Этот текст набран другим шрифтом</FONT> — ошибка: название шрифта, состоящее из двух слов, нужно заключать в кавычки. Иначе броузер воспримет как значение параметра face только первое слово — в данном случае Arial.

в) <FONT face="Arial Cyr"> Этот текст набран другим шрифтом</FONT> — ошибок нет.

г) <FONT face="ARIAL CYR"> Этот текст набран другим шрифтом</FONT> — и здесь ошибок нет. HTML нечувствителен к регистру, поэтому название шрифта можно набирать как прописными, так и строчными буквами.

д) <FONT face="Arial Bold"> Этот текст набран другим шрифтом</FONT> — ошибка: такого шрифта нет. Для того чтобы текст выводился полужир-

ным Arial, нужно, кроме дескриптора <FONT>, где задается шрифт Arial, использовать дескриптор <B>.

2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а) <FONT size=3> Этот текст набран другим размером </FONT> — ошибок нет. Но эффект от применения такой конструкции будет только в том случае, если размер окружающего текста не равен 3.

   б) <FONT size=8 pt> Этот текст набран другим размером </FONT> — ошибка: дескриптор <FONT> позволяет задать только 7 размеров шрифта, и не в пунктах, а в собственных единицах.

   в) <FONT size=-3> Этот текст набран другим размером </FONT> — ошибок нет. Но эффект от применения такой конструкции будет только в том случае, если размер окружающего текста не меньше 4.

   г) <FONT size=+7> Этот текст набран другим размером </FONT> — ошибка: каким бы ни был окружающий текст, его размер нельзя увеличить больше, чем на 6 единиц.

3. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а) <FONT color=#123456> Этот текст набран другим цветом</FONT> — ошибок нет. Но эффект от применения такой конструкции будет только в том случае, если цвет окружающего текста отличается от #123456.

   б) <FONT color=#&123456> Этот текст набран другим цветом</FONT> — ошибка: цвет в HTML задается по шаблону #RRGGBB, и символ & здесь не нужен.

   в) <FONT color=#ABCDEF> Этот текст набран другим цветом</FONT> — ошибок нет.

   г) <FONT color=fGHIKLM> Этот текст набран другим цветом</FONT> — ошибка: цвет в HTML задается в шестнадцатеричном формате, а в нем используются только цифры от 0 до 9 и буквы от A до F.

   д) <FONT color=lred> Этот текст набран другим цветом</FONT> — неправильно. Мнемонические имена цветов не нуждаются в предваряющих их символах.

   е) <FONT color=#&red> Этот текст набран другим цветом</FONT> — ошибка: мнемонические имена цветов не нуждаются в предваряющих их символах.

   ж) <FONT color=red> Этот текст набран другим цветом</FONT> — ошибок нет.

   з) <FONT color-red> Этот текст набран другим цветом</FONT> — ошибка: между параметром и его значением должен стоять знак "=".

4. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а) Этот текст набран мелким шрифтом и красным цветом — ошибок нет.

   б) Этот текст набран красным цветом и мелким шрифтом — и здесь ошибок нет: параметры дескриптора можно задавать в любом порядке.

в) <color=red font size=l> Этот текст набран красным цветом и мелким шрифтом - ошибка: вначале определяется дескриптор, затем — параметры.

г) <font="red, 1"> Этот текст набран красным цветом и мелким шрифтом — ошибка: параметры дескриптора задаются по отдельности.

д) Этот текст набран красным цветом и мелким шрифтом — ошибка: каждый из вложенных дескрипторов нуждается в собственном закрывающем дескрипторе, даже если они одинаковые.

е) Этот текст набран красным цветом и мелким шрифтом — ошибок нет.

ж) <font color=red> Этот текст набран красным цветом и <font size=1>мелким шрифтом — ошибка: в закрывающем дескрипторе не нужно перечислять параметры.

5. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <BASEFONT color=blue> Синий текст <FONT color=yellow> Зеленый текст </FONT> - в коде ошибок нет. Но предположение, что цвета смешаются, как при наложении цветных стекол, и синий с желтым даст зеленый, — ошибка.

б) <BASEFONT color=blue> Синий текст <FONT color=yellow> Желтый текст </FONT> — ошибок нет.

в) <BASEFONT color=blue> Синий текст </BASEFONT> — ошибка: дескриптор <BASEFONT> — непарный и в закрывающем дескрипторе не нуждается.

Глава 5

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <H1> Это заголовок первого уровня — ошибка: дескрипторы заголовка нужно закрывать.

б) <H1> Это заголовок первого уровня </H1> — ошибок нет.

в) <H1> Это заголовок первого уровня </H2> — ошибка: дескриптор заголовка первого уровня <H1> нужно закрыть дескриптором заголовка этого же уровня <H1>.

2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <H1> Это заголовок
 из двух строк </H1> — ошибок нет.

б) <H1> Это заголовок <P> из двух строк </H1> — и здесь ошибок нет. Только расстояние между строками будет больше, чем в случае (а).

в) <H1> Это заголовок <H2> из двух строк </H2></H1> — формально ошибок нет. Но на самом деле этот составной заголовок состоит из заголовков двух уровней: первого и второго.

г) <H1> Это заголовок </H1><H2> из двух строк </H2> - формально ошибок нет. Эффект от этого кода такой же, как в случае (в).

3. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а) <H1>Это заголовок, выровненный по левому краю</H1> — ошибок нет. По умолчанию заголовки выравниваются по левому краю.

   б) <H1 align=left>Это заголовок, выровненный по левому краю</H1> — ошибок нет.

   в) <H1 align=right>Это заголовок, выровненный по правому краю</H1> — ошибок нет.

   г) <H1 align=center>Это заголовок, выровненный по центру</H1> — ошибок нет.

   д) <H1 align=justify>Это заголовок, выровненный по ширине</H1> — ошибка: в заголовках параметр align не может принимать значение justify.

Глава 6

1. Дескриптор **<S>** является аналогом:

   а) дескриптора <STRONG> — неправильно. Дескриптором <STRONG> выделяют фрагменты, на которые посетителю страницы следует обратить особое внимание, а дескриптор **<S>** означает зачеркивание текста.

   б) дескриптора <EM> — неправильно. Дескриптором <EM> выделяют фрагменты, на которые посетителю страницы следует обратить внимание, а дескриптор <S> означает зачеркивание текста.

   в) дескриптора <STRIKE> — правильно.

   г) дескриптора <DEL> — неправильно. Такой текст на экране выглядит зачеркнутым, так же как и размеченный дескриптором **<S>**, но код с дескриптором <DEL> несет также дополнительную информацию — выделенные им данные, как правило, являются устаревшими.

2. Какой или какие из следующих фрагментов кода содержат ошибки?

   а) <BLOCK QUOTE>Тише едешь — дальше будешь</BLOCK QUOTE>. Народная мудрость — ошибка: название дескриптора <BLOCKQUOTE> не содержит пробелов.

   б) <BLOCKQUOTE>Тише едешь — дальше будешь</BLOCKQUOTE>. Народная мудрость — ошибок нет.

   в) <BLOCK quote="Тише едешь — дальше будешь">. Народная мудрость —, ошибка: этот дескриптор называется <BLOCKQUOTE> и не имеет параметров.

   г) <ACKRONYM title="язык разметки гипертекста">HTML</ACKRONYM> — ошибок нет.

   д) <ACKRONYM>HTML</ACKRONYM> — язык разметки гипертекста — формально ошибок нет. Но дескриптор <ACKRONYM> обычно используют для того, чтобы указать в нем расшифровку сокращения.

   е) Формула этилового спирта — C₂H₅OH — ошибок нет.

   ж) Формула этилового спирта — C²H⁵OH — ошибка: здесь нужно использовать дескриптор нижнего индекса <SUB>, а не верхнего <SUP>.

3. Какой из следующих фрагментов состоит из двух абзацев?

   а) `&laquo;<CITE>Тише едешь &mdash; дальше будешь</CITE>&raquo;  &mdash;` народная мудрость — один абзац.

   б) `<CITE>Тише едешь &mdash; дальше будешь</CITE>. <ADDRESS>Народная мудрость</ADDRESS>` — два абзаца: содержимое дескриптора <ADDRESS> выводится с новой строки.

   в) `<BLOCKQUOTE>Тише едешь &mdash;` дальше `будешь</BLOCKQUOTE>. <ADDRESS>` Народная `мудрость</ADDRESS>` - два абзаца: содержимое дескриптора `<ADDRESS>` выводится с новой строки.

   г) `<BLOCKQUOTE>Тише едешь &mdash;` дальше `будешь</BLOCKQUOTE>.` Народная мудрость — два абзаца: содержимое дескриптора <BLOCKQUOTE> выводится в отдельном абзаце.

   д) `<PRE>Тише едешь &mdash;` дальше `будешь</PRE>.` Народная мудрость — два абзаца: содержимое дескриптора <PRE> выводится в отдельном абзаце.

4. Какие из следующих пар фрагментов отображаются в броузере одинаково?

   а) `<DFN>HTML</DFN> &mdash;` язык разметки гипертекста `<DEL>HTML</DEL> &mdash;` язык разметки гипертекста

   Фрагменты отображаются по-разному: содержимое дескриптора `<DFN>` выводится полужирным шрифтом, содержимое дескриптора `<DEL>` — зачеркнутым.

   б) `<DFN>HTML</DFN> &mdash;` язык разметки гипертекста `<ACKRONYM title="язык разметки гипертекста">HTML</ACKRONYM>`

   Фрагменты отображаются по-разному: содержимое дескриптора `<DFN>` выводится полужирным шрифтом, содержимое дескриптора `<ACKRONYM>` — обычным, а расшифровка сокращения — при наведении на него указателя мыши.

   в) `<DFN>HTML</DFN> &mdash;` язык разметки гипертекста `<B>HTML</B> &mdash;` язык разметки гипертекста

   Фрагменты отображаются одинаково: в обоих случаях слово HTML выводится полужирным шрифтом.

   г) `<DFN>HTML</DFN> &mdash;` язык разметки гипертекста `<I>HTML</I> &mdash;` язык разметки гипертекста

5. Какие из следующих пар фрагментов отображаются в броузере одинаково?

   а) Саша + `<DEL>Маша</DEL><INS>Саша` + `<B>Маша</B><I>Наташа</I>` = Дружба `Наташа</INS>` = Дружба

   Фрагменты отображаются по-разному: содержимое дескриптора <DEL> выводится зачеркнутым, дескриптора <INS> — подчеркнутым, дескриптора <B> — полужирным, а дескриптора <I> — курсивом.

   б) Саша + `<DEL>Маша</DEL><INS>Саша` + `<S>Маша</S><U>Наташа</U>` = Дружба `Наташа</INS>` = Дружба

   Фрагменты отображаются одинаково: содержимое дескрипторов <DEL> и <S> выводится зачеркнутым, а дескрипторов <INS> и <U> — подчеркнутым.

в) Саша + <DEL>Маша</DEL> <INS>Наташа</INS> = Дружба

Саша + <STRIKE>Маша</STRIKE> <STRONG>Наташа</STRONG> = Дружба

Фрагменты отображаются по-разному: содержимое дескрипторов <DEL> и <STRIKE> выводится зачеркнутым, дескриптора <INS> — подчеркнутым, а дескриптора <STRONG> — полужирным.

6. Какие из следующих пар фрагментов отображаются в броузере одинаково?

а) Вот фрагмент HTML-кода:
```
<PRE>
<I>курсив<I>
<U>подчеркнутый<U>
</PRE>
```
Вот фрагмент HTML-кода:
```
<PRE>
курсив
подчеркнутый
</PRE>
```

Фрагменты отображаются по-разному: содержимое дескриптора <PRE> выводится с сохранением разделителей, но не форматирования.

б) Вот фрагмент HTML-кода:
```
<PRE>
<I>курсив</I>
<U>подчеркнутый</U>
</PRE>
```
Вот фрагмент HTML-кода:
```
<CODE>
<I>курсив</I>
<U>подчеркнутый</U>
</CODE>
```

Фрагменты отображаются по-разному: содержимое дескриптора <PRE> выводится с сохранением разделителей, а содержимое дескриптора <CODE> — без их сохранения.

в) Вот фрагмент HTML-кода:
```
<PRE>
<I>курсив</I>
<U>подчеркнутый</U>
</PRE>
```
Вот фрагмент HTML-кода:
```
<CODE>
&lt;I&gt;курсив&lt;/I&gt;
&lt;U&gt;подчеркнутый&lt;/U&gt;
</CODE>
```

Фрагменты отображаются по-разному: дескрипторы, вложенные в дескриптор <PRE>, обрабатываются броузером как обычно.

г) Вот фрагмент HTML-кода:
```
<PRE>
<I>курсив</I>
<U>подчеркнутый</U>
</PRE>
```
Вот фрагмент HTML-кода:
```
<CODE>
<BR>   <I>курсив</I>
<BR>   <U>подчеркнутый</U>
</CODE>
```

Фрагменты отображаются примерно одинаково (если считать, что три неразрывных пробела примерно равны одной табуляции).

д) Вот фрагмент HTML-кода:
```
<PRE>
<I>курсив</I>
<U>подчеркнутый</U>
</PRE>
```
Вот фрагмент HTML-кода:
```
<FONT
face="Courier New">
<BR>   <I>курсив</I>
<BR>   <U>подчеркнутый</U>
</FONT>
```

Фрагменты отображаются примерно одинаково (если считать, что три неразрывных пробела примерно равны одной табуляции).

7. Какие из следующих пар фрагментов отображаются в броузере одинаково?

а) Дескриптор <VAR><SUP></VAR> предназначен для отображения верхнего индекса

Дескриптор <I><SUP></I> предназначен для отображения верхнего индекса

Фрагменты отображаются одинаково.

б) Дескриптор `<VAR>&lt;SUP&gt;</VAR>` предназначен для отображения верхнего индекса

Дескриптор <FONT face="Courier New"><SUP></FONT> предназначен для отображения верхнего индекса

Фрагменты отображаются по-разному: содержимое дескриптора <VAR> — курсивом, содержимое дескриптора <FONT face="Courier New"> — шрифтом *Courier New*.

в) Дескриптор <KBD><SUP></KBD> предназначен для отображения верхнего индекса

Дескриптор <FONT face="Courier New"><SUP></FONT> предназначен для отображения верхнего индекса

Фрагменты отображаются одинаково.

г) Дескриптор <SAMP><SUP></SAMP> предназначен для отображения верхнего индекса

Дескриптор <FONT face="Courier New"><SUP></FONT> предназначен для отображения верхнего индекса

Фрагменты отображаются одинаково.

Глава 7

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?
   а) <UL start=1>
   <LI> Первый элемент списка
   <LI> Второй элемент списка
   </UL> — ошибка: у дескриптора <UL> нет параметра start.
   б) <UL start=circle>
   <LI> Первый элемент списка
   <LI> Второй элемент списка
   </UL> — ошибка: во-первых, у дескриптора <UL> нет параметра start, а во-вторых, этот параметр не принимает значения circle.
   в) <UL type=1>
   <LI> Первый элемент списка
   <LI> Второй элемент списка
   </UL> — ошибка: параметр type не принимает значения 1.
   г) <UL type=circle>
   <LI> Первый элемент списка
   <LI> Второй элемент списка
   </UL> — правильно.
2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?
   а) <OL start=1>
   <LI> Первый элемент списка
   <LI> Второй элемент списка
   </OL> — правильно.
   б) <OL start=circle>
   <LI> Первый элемент списка
   <LI> Второй элемент списка
   </OL> — ошибка: параметр start не принимает значения circle.

в) <OL type=1>
<LI> Первый элемент списка
<LI> Второй элемент списка
</OL> — правильно.

г) <OL type=circle>
<LI> Первый элемент списка
<LI> Второй элемент списка
</OL> — ошибка: параметр type дескриптора <OL> не принимает значения circle.

3. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <DL>
<DT> Термин 1</DT>
<DD> Определение 1</DD>
<DT> Термин 2</DT>
<DD> Определение 2</DD>
</DL> — правильно.

б) <DT> Термин 1</DT>
<DD> Определение 1</DD>
<DT> Термин 2</DT>
<DD> Определение 2</DD> — и это правильно: можно создать список определений и без дескриптора <DL>.

в) <DL type=circle>
<DT> Термин 1</DL>
<DD> Определение 1</DD>
<DT> Термин 2</DL>
<DD> Определение 2</DD>
</DL> — ошибка: у дескриптора <DL> нет параметра type.

г) <DL start=1>
<DT> Термин 1</DL>
<DD> Определение 1</DD>
<DT> Термин 2</DL>
<DD> Определение 2</DD>
</DL> — ошибка: у дескриптора <DL> нет параметра start

Глава 8

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <IMG src="Мои документы\vasya.gif"> — ошибка: в именах файлов и путях к ним не допускается использование кириллицы и пробелов; в качестве разделителей используются не обратные, а прямые косые.

б) <IMG src="Mydoc\vasya.gif"> — ошибка: в качестве разделителей используются не обратные, а прямые косые.

в) <IMG src="Mydoc/vasya.gif"> — правильно.

г) <IMG src=Mydoc/vasya.gif> — правильно.

д) <IMG src="Mydoc/vasya.tif"> — ошибка: формат TIFF не относится к форматам графики для Internet, изображения в этом формате не открываются броузером.

е) <IMG src="Mydoc/vasya.GIF"> — формально ошибки нет. Но поскольку в пути файла различаются строчные и прописные буквы, рекомендуется всегда использовать строчные, чтобы не запутаться.

2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <IMG src=vasya.gif border=l bordercolor=red> — ошибка: в дескрипторе <IMG> отсутствует параметр bordercolor.

б) <IMG src=vasya.gif border=l> — правильно.

в) <IMG src=vasya.gif border=1,5> — ошибка: параметр border может принимать только целочисленные значения, так как определяет ширину рамки в пикселях.

г) <IMG src=vasya.gif border=1.5> — ошибка: параметр border может принимать только целочисленные значения, так как определяет ширину рамки в пикселях.

д) <IMG src=vasya.gif border=red> — ошибка: параметр border не может принимать никакие значения, кроме целочисленных, так как определяет ширину рамки в пикселях.

е) <IMG src=vasya.gif border=0> — правильно.

3. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <IMG src=vasya.gif alt=vasya.jpg> — формально ошибки нет. Но при наведении указателя мыши на изображение vasya.gif будет выводиться комментарий vasya.jpg (а вовсе не изображение из файла vasya.jpg, как можно было бы подумать).

б) <IMG src=vasya.gif alt=vasya> — правильно. Но при наведении указателя мыши на изображение vasya.gif будет выводиться комментарий vasya.

в) <IMG src=vasya.gif alt="Это Вася"> — правильно. Но при наведении указателя мыши на изображение vasya.gif будет выводиться комментарий Это Вася.

г) <IMG src=vasya.gif alt=Это Вася> — ошибка: если комментарий содержит пробелы, его нужно заключать в кавычки. Иначе будет выводиться только первое слово.

д) <IMG src=vasya.gif alt=Eto Vasya> — ошибка: если комментарий содержит пробелы, его нужно заключать в кавычки. Иначе будет выводиться только первое слово, и то, что комментарий написан "транслитом", не поможет.

4. Для того чтобы текст обтекал изображение справа, используется следующий код:

а) <IMG src=pict.gif align=right> текст — неправильно. Здесь изображение разместится справа, а текст будет обтекать его слева.

б) <IMG src=pict.gif align=left> текст — правильно.

в) <IMG src=pict.gif> текст — неправильно. Текст разместится в той же строке, что и изображение.

г) <IMG src=pict.gif align=left><BR clear=right> текст — формально все правильно. Но дескриптор <BR clear=right> здесь не нужен.

д) <IMG src=pict.gif align=left><BR clear=left> текст — ошибка. Дескриптор <BR clear=left> отменит обтекание.

5. Для того чтобы текст обтекал изображение слева, используется следующий код:

а) <IMG src=pict.gif align=right> текст — правильно.

б) <IMG src=pict.gif align=left> текст — неправильно. Здесь изображение разместится слева, а текст будет обтекать его справа.

в) <IMG src=pict.gif> текст — неправильно. Текст разместится в той же строке, что и изображение.

г) <IMG src=pict.gif align=right><BR clear=right> текст — ошибка. Дескриптор <BR clear=right> отменит обтекание.

д) <IMG src=pict.gif align=right><BR clear=left> текст — формально все правильно. Но дескриптор <BR clear=left> здесь не нужен.

6. Для того чтобы изображение разместилось по середине строки, напишите следующий код:

а) <IMG src=pict.gif align=center> — неправильно. Параметр align в дескрипторе <IMG> не может принимать значения center.

б) <BR clear=center><IMG src=pict.gif> — неправильно. Параметр clear в дескрипторе
 не может принимать значения center.

в) <IMG src=pict.gif><BR clear=center> — неправильно. Параметр clear в дескрипторе
 не может принимать значения center.

г) <P align=center><IMG src=pict.gif> — правильно.

Глава 9

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <A> Гиперссылка </A> — ошибка: ссылка без указания адреса работать не будет.

б) <HREF> page2.htm </HREF> — ошибка: дескриптора <HREF> не существует.

в) <A href=page2.htm> Гиперссылка </A> — правильно.

г) <A href="page2.htm"> Гиперссылка </A> — правильно.

д) <A target=page2.htm> Гиперссылка </A> — ошибка: адрес присваивается параметру href, а не target.

2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <A href=page2.htm target=blank> Гиперссылка </A> — ошибка: предопределенный параметр называется _blank.

б) <A href=page2.htm target= _blank> Гиперссылка </A> — правильно.

в) <A href=page2.htm target=blank_> Гиперссылка </A> — ошибка: предопределенный параметр называется _blank.

3. Какой или какие из следующих фрагментов HTML-кода, содержащие ссылку на метку label, которая находится на странице page2.htm, содержат ошибки?

   а) <A href=page2.htm name=label> Гиперссыпка </A> — ошибка: параметр name используется для создания метки, а не для ссылки на нее.

   б) <A href="page2.htm\label"> Гиперссылка </A> — ошибка: разделителем между именем файла и меткой служит символ #.

   в) <A href="page2.htm/label"> Гиперссылка </A> — ошибка: разделителем между именем файла и меткой служит символ I.

   г) <A href="page2.htm#label"> Гиперссылка </A> — правильно.

   д) <A href="page2.htm, label"> Гиперссылка </A> — ошибка: разделителем между именем файла и меткой служит символ #.

4. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а) <A href="mailto:tome@mail.dom"> Написать мне письмо </A> — правильно.

   б) <A href="mailto:tome@mail.dom; subject=Привет;body=Как дела?"> Написать мне письмо </A> — ошибка: разделителем между адресом и словом subject служит символ ?; пробелы в шаблоне письма недопустимы.

   в) <A href="mailto:tome@mail.dom? subject=Привет;body=Как дела?"> Написать мне письмо </A> — ошибка: пробелы в шаблоне письма недопустимы.

   г) <A href="mailto:tome@mail.dom? subject=Привет;body=Как%20дела?"> Написать мне письмо </A> — правильно.

   д) <A href="mailto:tome@mail.dom?subject=Привет"> Как дела? </A> — формально ошибок нет. Но фраза "Как дела?" не попадет в письмо, а будет служить гиперссылкой.

   е) <A href="mailto:tome@mail.dom?subject=Привет;"> Как дела? </A> - • формально ошибок нет. Но фраза "Как дела?" не попадет в письмо, а будет служить гиперссылкой. Кроме того, в теме письма будет написано Привет;.

5. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а)
   ```
   <MAP name=image>
   <AREA shape=triangle coords="10,20,30,40" href=pag2.htm>
   </MAP>
   ```
   — ошибка: параметр shape не может принимать значения triangle.

   б)
   ```
   <MAP name=image>
   <AREA shape=rect coords="10,20,30,40" href=pag2.htm>
   </MAP>
   ```
   — правильно.

   в)
   ```
   <MAP name=image>
   <AREA shape=poly coords="10,20,30,40" href=pag2.htm>
   </MAP>
   ```
   — ошибка: при описании многоугольника последней указывается та же пара координат, что и в начале описания, чтобы фигура была замкнутой.

6. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <MAP name=image>
...
</MAP>
...
<IMG usemap=image> — ошибка: при ссылке на карту изображения перед ее именем ставится символ #. Кроме того, в дескрипторе <IMG> отсутствует ссылка на файл с самим изображением.

б) <MAP name=image>
...
</MAP>
...
<IMG usemap=#image> — ошибка: в дескрипторе <IMG> отсутствует ссылка на файл с самим изображением.

в) <MAP name=image>
...
</MAP>
...
<IMG usemap=#image src=image.jpg> — правильно.

г) <MAP name=image>
...
</MAP>
...
<IMG src=image.jpg#image> — ошибка: для ссылки на карту изображения используется параметр usemap.

д) <MAP name=image>
...
</MAP>
...
<A href=image> <IMG src=image.jpg> </A> — ошибка: для ссылки на карту изображения используется параметр usemap.

Глава 10

1. Напишите код для верхней страницы на рис. 10.1.

Код может выглядеть, например, так.

```
<table align=center cellpadding=0>
<tr>
<td colspan="2" width=50% height=25%> </td>
<td bgcolor="lightblue" rowspan=2 align=center height=250 width=25%>
      <img src="up.gif" width=50><b><BR>B<BR>B<BR>e<BR>p<BR>x</bx/td>
<td rowspan=2 width=25%> </td>
</tr>
```

```
<tr>
<td colspan="2" bgcolor=darkyellow width=250 align=center height=20>
      <img src="left.gif" align="absmiddle" width=50>
      <b> B п e в o</b></td>
</tr>
<tr>
<td width=25% height=25%> </td>
<td rowspan=2 bgcolor=lightgreen align=center height=250 width=25%>
      <b>B<BR>и<BR>и<BR>з<BR></b><img src="down.gif" width=50></td>
<td colspan=2 bgcolor=pink align=center width=250>
      <b>B п p a в o </b>
      <img src="right.gif" align="absmiddle" width=50></td>
</tr>
<tr>
<td height=25%> </td>
<td colspan=2> </td>
</tr>
</table>
```

2. Напишите код для нижней страницы на рис. 10.1.

Код может выглядеть, например, так.

```
<TABLE cellpadding=5 align=center>
<TR>
      <TD>Кто в центре?</TD>
      <TDrowspan=2><b>Я!</b></TD>
      <TD>Кто в центре?</TD>
</TR>
<TR>
      <TD>КТО в центре?</TD>
      <TD>Кто в центре?</TD>
</TR>
</TABLE>
```

Глава 11

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а) `<FRAMESET rows=70,*> ... </FRAMESET>` – правильно.

   б) `<FRAMESET rows=70%,*,70>... </FRAMESET>` – правильно.

   в) `<FRAMESET cols=70,*%> ... </FRAMESET>` — ошибка: символ * означает неопределенный размер, так что знак процента после него не нужен.

   г) `<FRAMESET cols=70000,*> ... </FRAMESET>` — С точки зрения HTML все правильно. Но часто ли вам встречались мониторы с разрешением 7000С пикселей по горизонтали?

   д) `<FRAMESET rows=*,*> ... </FRAMESET>` — правильно.

е) `<FRAMESET rows=2*,*> ... </FRAMESET>` — правильно.

ж) `<FRAMESET rows=**,*> ... </FRAMESET>` — ошибка: двойной символ * не допускается.

з) `<FRAMESET rows=*;*> ... </FRAMESET>` — ошибка: разделителем между размерами областей служит запятая, а не точка с запятой.

и) `<FRAMESET rows=*> ... </FRAMESET>` — формально все правильно. Но какой смысл во фреймовой структуре, состоящей из одного фрейма неопределенного размера?

2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а)
```
<FRAMESET cols=*,*>
<FRAME src="source1.htm">
</FRAMESET>
```
— ошибка: отсутствует второй фрейм.

б)
```
<FRAMESET cols=*,*>
<FRAME src="source1.htm">
<FRAME src="source2.htm">
</FRAMESET>
```
— правильно.

в)
```
<FRAMESET cols=*,* rows=*,*>
<FRAME src="source1.htm">
<FRAME src="source2.htm">
<FRAME src="source3.htm">
<FRAME src="source4.htm">
</FRAMESET>
```
— ошибка: нельзя использовать в одном дескрипторе `<FRAMESET>` параметры cols и rows одновременно.

г)
```
<FRAMESET cols=*,*,*,*>
<FRAME src="source1.htm">
<FRAME src="source2.htm">
<FRAME src="source3.htm">
<FRAME src="source4.htm">
</FRAMESET>
```
— правильно.

д)
```
<FRAMESET cols=*,*>
<FRAME src="source1.htm">
<FRAME src="source1.htm">
</FRAMESET>
```
— правильно. Правда, в обоих фреймах будет отображен один и тот же файл. Но, возможно, именно таков был замысел дизайнера...

3. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а)
```
<FRAMESET cols=*,* frameborder=no>
<FRAME src="source1.htm">
<FRAME src="source2.htm">
</FRAMESET>
```
— правильно.

б) <FRAMESET cols=*,*>
<FRAME src="source1.htm" frameborder=no>
<FRAME src="source2.htm">
</FRAMESET> — правильно.

в) <FRAMESET cols=*,* border=no>
<FRAME src="source1.htm">
<FRAME src="source2.htm">
</FRAMESET> — ошибка: значением параметра border является число пикселей.

г) <FRAMESET cols=*,* border=10>
<FRAME src="source1.htm">
<FRAME src="source2.htm">
</FRAMESET> — правильно.

д) <FRAMESET cols=*,*>
<FRAME src="source1.htm" border=10>
<FRAME src="source2.htm">
</FRAMESET> — ошибка: параметр border используется в дескрипторе <FRAMESET>, но не в дескрипторе <FRAME>.

4. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <FRAMESET cols=*,*>
<IFRAME src="source1.htm" border=10>
<IFRAME src="source2.htm">
</FRAMESET> — ошибка: дескриптор <IFRAME> не нуждается во фреймовой структуре.

б) <TABLE>
<TR><TD><IFRAME src="source1.htm" border=10></TD>
<TD><IFRAME src="source2.htm"></TD></TR>
</TABLE> — правильно.

в) <IFRAME> Содержимое встроенного фрейма</IFRAME> — ошибка: содержимое встроенного фрейма находится в отдельном файле.

Глава 12

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <FORM action=mailto:mymail@mail.dom method=post enctype=application /x-www-form-urlencoded>
...
</FORM> — ошибка: при передаче данных электронной почтой используется тип кодирования text/plain.

б) <FORM action=mailto:mymail@mail.dom method=get enctype=text/plain>
...
</FORM> — ошибка: при передаче данных электронной почтой используется метод post.

в) <FORM action=mailto:mymail@mail.dom method=post enctype=text/plain>
...
</FORM> — правильно.

г) <FORM action=mailto:mymail@mail.dom method=post enctype=plain>
...
</FORM> — ошибка: при передаче данных электронной почтой используется тип кодирования text/plain.

д) <FORM action=mailto:mymail@mail.dom method=post enctype=text>
...
</FORM> — ошибка: при передаче данных электронной почтой используется тип кодирования text/plain.

2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <INPUT type=text value="Введите текст:"> — ошибка: необходимо присвоить имя элементу формы.

б) <INPUT type=text name=itext value="Введите текст:"> — правильно.

в) <INPUT type=text name=itext> Введите текст </INPUT> — ошибка: дескриптор <INPUT> — непарный.

г) Введите текст: <INPUT type=text name=itext> — правильно.

3. Сколько символов можно ввести в следующей строке и какова видимая длина этой строки?

<INPUT type=text name=itext size=10 maxlength=20>

а) 10 и 20, соответственно — ошибка.

б) 20 и 10, соответственно — правильно.

4. Какой или какие из следующих фрагментов HTML-кода кнопки отправки данных содержат ошибки?

а) <INPUT type=button value=submit> — ошибка: не указано, какую именно операцию выполняет кнопка.

б) <INPUT type=submit value=OK> — правильно.

в) <INPUT type=submit name=OK>OK</INPUT> — ошибка дескриптор <INPUT> — непарный.

г) <INPUT type=submit name=OK> — правильно.

д) <INPUT type=image src=ok.gif value=OK> — правильно.

5. Какой или какие из следующих фрагментов HTML-кода кода обнуления данных формы содержат ошибки?

а) <INPUT type=button value=reset> — ошибка: не указано, какую именно операцию выполняет кнопка.

б) <INPUT type=reset value=Cancel> — правильно.

в) <INPUT type=reset name=Cancel>Cancel</INPUT> — ошибка дескриптор <INPUT> — непарный.

г) <INPUT type=reset name=Cancel> — правильно.

д) <INPUT type=image src=cancel.gif value=Cancel> — ошибка: тип image соответствует только кнопке передачи данных.

6. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

а) <TEXTAREA>Введите длинный текст</TEXTAREA> — ошибка: для передачи данных элемент формы должен иметь имя.

б) <TEXTAREA name=ltext>Введите длинный текст</TEXTAREA> — правильно.

в) <TEXTAREA name=ltext value=Введите длинный текст >Введите длинный текст </TEXTAREA> — формально правильно, но значение value — лишнее.

г) <TEXTAREA name=ltext value=Введите длинный текст></TEXTAREA> — формально правильно, но значение value не появится в поле ввода.

д) <TEXTAREA name=ltext value=Введите длинный текст> — ошибка: дескриптор <TEXTAREA> — парный.

1. Какой или какие из следующих списков позволяют выбрать несколько вариантов?

а)
```
<INPUT type=checkbox name=v1>Вариант 1
<INPUT type=checkbox name=v2>Вариант 2
<INPUT type=checkbox name=v3>Вариант 3
```
— многовариантный.

б)
```
<INPUT type=checkbox name=v1 checked>Вариант 1
<INPUT type=checkbox name=v2 checked>Вариант 2
<INPUT type=checkbox name=v3 checked>Вариант 3
```
— многовариантный.

в)
```
<INPUT type=radio name=v1 checked>Вариант 1
<INPUT type=radio name=v1 checked>Вариант 2
<INPUT type=radio name=v1 checked>Вариант 3
```
— одновариантный.

г)
```
<INPUT type=radio name=v1>Вариант 1
<INPUT type=radio name=v1>Вариант 2
<INPUT type=radio name=v1>Вариант 3
```
— одновариантный.

д)
```
<SELECT name=v1>
<OPTION value="Вариант 1">
<OPTION value="Вариант 2">
<OPTION value="Вариант 3">
</SELECT>
```
— одновариантный.

е)
```
<SELECT name=v1>
<OPTION value="Вариант 1" checked>
<OPTION value="Вариант 2" checked>
<OPTION value="Вариант 3" checked>
</SELECT>
```
— одновариантный.

ж) <SELECT name=v1 checked>
<OPTION value="Вариант 1">
<OPTION value="Вариант 2">
<OPTION value="Вариант 3">
</SELECT> — ошибка: дескриптор <SELECT> не имеет параметра checked.

з) <SELECT name=v1 multiple>
<OPTION value="Вариант 1">
<OPTION value="Вариант 2">
<OPTION value="Вариант 3">
</SELECT> — многовариантный.

Глава 13

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?
   а) <BODY margin=10> — ошибка: параметра margin не существует.
   б) <BODY marginwidth=10> — правильно.
   в) <BODY leftmargin=10> — правильно.
   г) <BODY leftmargin=10%> — правильно.
2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?
   а) <BODY text=4> — ошибка: цвет текста задается в формате #RRGGBB.
   б) <BODY text=Times> — ошибка: параметр text определяет не гарнитуру, а цвет.
   в) <BODY text=black> — правильно.
   г) <BODY text=0> — ошибка: цвет текста задается в формате IRRGGBB.
   д) <BODY text=#1234> — правильно.
   е) <BODY text=1234> — ошибка: цвет текста задается в формате IRRGGBB.
3. Как должен выглядеть код, чтобы гиперссылки в тексте были зеленого цвета?
   а) <BODY link=green> — правильно.
   б) <BODY alink=green> — ошибка: параметр alink определяет цвет только активной ссылки.
   в) <BODY blink=green> — ошибка: параметра blink не существует.
   г) <BODY xlink=green> — ошибка: параметра blink не существует.
   д) <BODY ylink=green> — ошибка: параметра blink не существует.
   е) <BODY vlink=green> — ошибка: параметр vlink определяет цвет использованных ссылок.
4. Как должен выглядеть код, чтобы фон страницы был желтого цвета?
   а) <BODY color=yellow> — ошибка: параметра color не существует.
   б) <BODY bgcolor=yellow> — правильно.
   в) <BODY background=yellow> — ошибка: параметр background определяет не цвет фона, а фоновое изображение.

Глава 14

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а) <HEAD>Заголовок страницы</HEAD> — ошибка: заголовок страницы помещается между дескрипторами <TITLE> и </TITLE>.

   б) <TITLE text="Заголовок страницы"> — ошибка: дескриптор <TITLE> не имеет параметров.

   в) <TITLE>Заголовок страницы</TITLE> — правильно.

   г) <TITLE><U>Заголовок страницы</U></TITLE> — ошибка: между дескрипторами <TITLE> и </TITLE> форматирование не допускается.

   д) <TITLE align=center>Заголовок страницы</TITLE> — ошибка: дескриптор <TITLE> не имеет параметров.

2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а) <META name=keywords>список ключевых слов</META> — ошибка: дескриптор <META> одинарный и не имеет закрывающего дескриптора </META>.

   б) <META name=keywords contents="список ключевых слов"> — ошибка: параметра contents не существует.

   в) <META name=keyword content="список ключевых слов"> — ошибка: мета-записи keyword не существует.

   г) <META name=keywords content="список ключевых слов"> — правильно.

   д) <META http-equiv=keywords content="список ключевых слов"> — ошибка: мета-запись keywords требует параметра name.

3. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а) <META name=description content="аннотация к странице" name=copyright content="А.Б. Иванов"> — ошибка: дескриптор <META> может содержать только одну мета-запись.

   б) <META namel=description content="аннотация к странице" name2=copyright content="А.Б. Иванов"> — ошибка: дескриптор <META> может содержать только одну мета-запись.

   в) <META name=description content="аннотация к странице" http-equiv=copyright content="А.Б. Иванов"> — ошибка: дескриптор <META> может содержать только одну мета-запись.

   г) <META name=description content="аннотация к странице">
   <META name=copyright content="А.Б. Иванов"> — правильно.

   д) <META>
   name=description content="аннотация к странице"
   name=copyright content="А.Б. Иванов"
   </META> — ошибка: дескриптор <META> — одиночный.

Глава 15

1. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а) <I class="font-size: 18; color: white; background-color: black">
   Это текст, к которому применен стиль.</I> — ошибка. Для описания стиля внутри дескриптора используется параметр style.

   б) <I style="font-size: 18; color: white; background-color: black">
   Это текст, к которому применен стиль.</I> — правильно.

   в) <I.style="font-size: 18; color: white; background-color: black">
   Это текст, к которому применен стиль.</I> — ошибка. Описание стиля в формате дескриптор.подкласс используется только в таблице стилей.

   г) <I class=inverse>
   Это текст, к которому применен стиль.</I> — правильно, при условии, что класс inverse описан в таблице стилей.

2. Какой или какие из следующих фрагментов HTML-кода содержат ошибки?

   а) <STYLE>
   I.inverse="font-size: 18; color: white; background-color: black"
   </STYLE> — ошибка: описание стиля в таблице стилей заключается в фигурные скобки; между именем и описанием стиля ставится не знак равенства, а пробел.

   б) <STYLE>
   I.inverse:"font-size: 18; color: white; background-color: black"
   </STYLE> — ошибка: описание стиля в таблице стилей заключается в фигурные скобки; между именем и описанием стиля ставится не двоеточие, а пробел.

   в) <STYLE>
   inverse="font-size: 18; color: white; background-color: black"
   </STYLE> — ошибка: описание стиля в таблице стилей заключается в фигурные скобки; между именем и описанием стиля ставится не знак равенства, а пробел; перед именем стиля ставится точка.

   г) <STYLE>
   I.inverse={font-size: 18; color: white; background-color: black"}
   </STYLE> — ошибка: между именем и описанием стиля ставится не знак равенства, а пробел.

   д) <STYLE>
   inverse "font-size: 18; color: white; background-color: black"
   </STYLE> — ошибка: описание стиля в таблице стилей заключается в фигурные скобки; перед именем стиля ставится точка.

е) <STYLE>
.inverse{font-size: 18; color: white; background-color: black"}
</STYLE> — правильно.

ж) <STYLE>
I.inverse {font-size: 18; color: white; background-color: black"}
</STYLE> — правильно.

3. Требуется, чтобы весь полужирный курсив в тексте выводился красным цветом. Какие из предлагаемых ниже стилей позволяют это сделать?

а) U I {color=red} — правильно.

б) U {font-style=italic; color=red} — правильно.

в) I {font-style=bold; color=red} — ошибка: атрибут font-style не имеет значения bold.

г) I {font-weight=bold; color=red} — правильно.

д) I.bold {color=red} — ошибка: такой стиль описывает только красный курсив, без полужирного.

е) U.italic {color=red} - ошибка: такой стиль описывает только красный полужирный шрифт, без курсива.

ж) I.bold {font-weight=bold; color=red} — правильно.

з) DIV {font-style=italic; font-weight=bold; color=red} — ошибка: красным полужирным курсивом будут выводиться только абзацы, заключенные внутри дескриптора <DIV>.

Глава 16

1. В каком случае вывод сообщения произойдет по щелчку мыши?

а) <P onMouseDown="window.alert('Щелк!')"> Щелкните здесь — ошибка: для появления сообщения щелчок необязателен. Достаточно нажать кнопку и удерживать ее.

б) <P onMouseClick="window.alert('Щелк!')"> Щелкните здесь — правильно.

в) <P onMouseUp="window.alert('Щелк!')"> Щелкните здесь — ошибка. Событие произойдет после отпускания кнопки мыши.

г) <P onMouseOver="window.alert('Щелк!')"> Щелкните здесь — ошибка: щелкать не обязательно. Сообщение появится уже при попадании указателя мыши на абзац.

2. В каком случае вывод сообщения произойдет при наведении указателя мыши на изображение?

а) <IMG src=big.gif onMouseDown="window.alert('Щелк!')"> — ошибка: сообщение появится только по щелчку.

б) <IMG src=big.gif onMouseOut="window.alert('Щелк!')"> — ошибка: сообщение появится при удалении указателя мыши с изображения.

в) <IMG src=big.gif onMouseUp="window.alert('Щелк!')"> — ошибка: сообщение появится при отпускании нажатой кнопки мыши.

г) <IMG src=big.gif onMouseOver="window.alert('Щелк!')"> — правильно.

3. В каком случае вывод сообщения произойдет при удалении указателя мыши с изображения?

а) <IMG src=big.gif onMouseDown="window.alert('Щелк!')"> — ошибка: сообщение появится только по щелчку.

б) <IMG src=big.gif onMouseOut="window.alert('Щелк!')"> — правильно.

в) <IMG src=big.gif onMouseUp="window.alert('Щелк!')"> — ошибка: сообщение появится при отпускании нажатой кнопки мыши.

г) <IMG src=big.gif onMouseOver="window.alert('Щелк!')"> — ошибка: сообщение появится при попадании указателя мыши на изображение.

4. В состав страницы входит следующий код:

```
<BODY onUpload="window.alert('Страница выгружена')">.
```

В каком или в каких случаях будет появляться предупреждение?

а) Когда посетитель щелкнет на кнопке Назад (Back) — правильно.

б) Когда посетитель щелкнет на кнопке Вперед (Forward) — правильно.

в) Когда посетитель щелкнет на кнопке Обновить (Refresh) — правильно.

г) Когда посетитель щелкнет на кнопке Остановить (Stop) — ошибка: страница не выгружается, и сообщение не появляется.

5. В состав страницы входит следующий код:

```
<BODY onLoad="window.alert('Страница выгружена')">.
```

В каком или в каких случаях будет появляться предупреждение?

а) Когда посетитель щелкнет на кнопке Назад (Back) — ошибка: сообщение появится только при загрузке данной страницы.

б) Когда посетитель щелкнет на кнопке Вперед (Forward) — ошибка: сообщение появится только при загрузке данной страницы.

в) Когда посетитель щелкнет на кнопке Обновить (Refresh) — правильно.

г) Когда посетитель щелкнет на кнопке Остановить (Stop) — ошибка: сообщение появится только при загрузке данной страницы.

Приложение Б

Таблица дескрипторов HTML

Дескриптор	Описание дескриптора	Параметры	Описание параметров
<A>	Гиперссылка	href	Адрес объекта, на который указывает ссылка
		name*	Метка
		target	имя окна, в котором Будет открыт объект
<ACRONYM>	Сокращение	title	Расшифровка (отображается при наведении указателя мыши на сокращение)
<ADDRESS>	Адрес (обычно отображается курсивом в отдельном абзаце)	—	—
<AREA>***	Область карты изображения	alt	См. <img>
		coords	Координаты области (в пикселях, отсчет от верхнего левого угла рисунка)
		href	См. <a>
		NOHREF	Ссылка отсутствует
		shape	Форма области
		target	см. <a>
<B>	Полужирный	—	—
<BASEFONT>***	Параметры шрифта по умолчанию	См. <FONT>	
<BLOCKQUOTE>	Длинная цитата (обычно отображается в отдельном абзаце с отступом)	-	—
<BODY>		alink	Цвет активной гиперссылки по умолчанию
		background	Фоновый рисунок
		bgcolor	цвет фона
		bgproperties	Режим прокрутки фонового изображения
		bottommargin	Нижний отступ
		leftmargin	Левый отступ

Значения параметров	Описание значений
URL	Для генерации электронного письма используется такой шаблон: `mailto:адрес?subject=тема`
Имя	Имя метки в документе, на которую можно сослаться
имя	или одно из предопределенных значений: _blank (новое окно), _self (окно текущего документа), _parent (окно текущей фреймовой структуры), _top (окно всей фреймовой структуры)
Текст	—
—	—
Список координат через запятую	Круг: Хцентра, Уцентра, радиус; прямоугольник: Хлев, Yверх, Хправ, Униз; многоугольник: X1, Y1, X2, Y2, ... Xn, Yn, X1, Y1
—	—
circ	Круг
rect	Прямоугольник
poly	многоугольник
—	—
—	—
RRGGBB	Или мнемоническое имя
URL	
RRGGBB	Или мнемоническое имя
fixed	Прокрутка фона запрещена
Целое число	В пикселях или процентах****
Целое число	В пикселях или процентах****

Дескриптор	Описание дескриптора	Параметры	Описание параметров
<BODY> продолжение		link	Цвет гиперссылок по умолчанию
	Тело Web-страницы	marginheight	Вертикальные отступы
		marginwidth	Горизонтальные отступы
		rightmargin	Правый отступ
		scroll	Режим прокрутки
		text	Цвет текста по умолчанию
		topmargin	Верхний отступ
		vlink	Цвет использованных гиперссылок по умолчанию
 ***	Разрыв строки (переход на новую строку с сохранением параметров текущего абзаца)	clear	Местоположение следующей строки
<CITE>	Короткая цитата (обычно отображается курсивом)	—	—
<CODE>	Фрагмент программного кода (обычно отображается моноширинным шрифтом)	—	—
<COL>***	Параметры подгруппы столбцов	См. <COLGROUP>	
<COLSPAN>*	Параметры группы ячеек	align	См. <td>
		bgcolor	См. <td>
		height	См. <td>
		span	Количество столбцов в группе
		valign	См. <td>
		width	См. <td>
*	Определение	—	—
<DEL>	Удаленный текст (обычно отображается зачеркнутым)	—	—
<DFN>	Определение (обычно отображается курсивом)	—	—
<DIR>	Каталог	См. <UL>	
	Раздел текста с общими свойствами	align	Выравнивание

Значения параметров	Описание значений
#RRGGBB	Или мнемоническое имя
Целое число	В пикселях или процентах****
Целое число	В пикселях или процентах****
Целое число	В пикселях или процентах****
yes**	Прокрутка разрешена
no	Прокрутка запрещена
#RRGGBB	Или мнемоническое имя
Целое число	В пикселях или процентах****
#RRGGBB	Или мнемоническое имя
none**	Такое же, как у предыдущей
left	Под той строкой пикселей, левый край которой не занят рисунком
right	Под той строкой пикселей, правый край которой не занят рисунком
all	под той строкой пикселей, оба края которой не заняты рисунком
—	—
—	—
Целое число	По умолчанию -1
—	—
—	—
-	—
;м. <P>	

Дескриптор	Описание дескриптора	Параметры	Описание параметров
<DL>	Список определений	—	—
<DT>	Определяемый термин	—	—
<EM>	Выделение важного (обычно отображается курсивом)	—	—
<FONT>		color	Цвет
	Параметры шрифта	face	Гарнитура
		size	Размер
<FORM>	Форма для ввода данных	action	Ссылка на обработчик данных
		enctype	Тип кодирования
		method	Метод передачи
		name	Имя формы
		target	См. <a>
<FRAME>***	Фрейм	bordercolor	Цвет рамки
		frameborder	Наличие рамок у фреймов
			frameborder
		marginheight	Отступ содержимого от верхнего края
		marginwidth	Отступ содержимого от правого и левого краев
		name	Имя фрейма
		noresize	Запрет на изменение размеров
		scrolling	Режим прокрутки
		src	Файл с исходным содержимым фрейма
<FRAMESET>	Фреймовая структура	border	Толщина рамки
		bordercolor	См. <frame>
		cols	Размеры вертикальных областей
		frameborder	См. <frame>
		framespacing	Ширина области между фреймами
		rows	Размеры горизонтальных областей
<H1>...<H6>	Заголовок	align	Выравнивание

Значения параметров	Описание значений
—	—
—	—
—	—
#RRGGBB	Или мнемоническое имя
	Имя шрифта или список таких имен, разделенных запятой
..7 или -6...+6	Абсолютный или относительныйразмер. По умолчанию size=3
URL	
application/x-www-form-urlencoded**	Для обработки программой
text/plain	для передачи по электронной почте
get**	По ссылке
post	По значению
Имя	
#RRGGBB	Или мнемоническое имя
yes или Г*	Отображать рамку
по или 0	Не отображать рамку
Целое число	В пикселях
Целое число	В пикселях
Имя	Используется в качестве значения параметра target дескриптора <A>
—	—
ves	Прокрутка разрешена, полоса прокрутки присутствует независимо от потребности в ней
по	Прокрутка запрещена, полоса прокрутки отсутствует
auto**	Прокрутка разрешена, полоса прокрутки появляется по мере необходимости
URL	
Целое число	В пикселях
елое число или символ *	В пикселях или процентах****
елое число	В пикселях
елое число или символ *	В пикселях или процентах****
left**	По левому краю

Дескриптор	Описание дескриптора	Параметры	Описание параметров
<H1>...<H6>	(продолжение)		
<HEAD>	Область заголовка	—	—
<HTML>	Html-код	—	—
	Курсив	—	—
<IFRAME>*	Встроенный фрейм	frameborder	См. <frame>
		height	Высота
		marginheight	См. <frame>
		marginwidth	См. <frame>
		name	См. <frame>
		scrolling	См. <frame>
		src	См. <frame>
		width	Ширина
<IMG>***	Вставка изображения	align	Выравнивание
		alt	Комментарий
		border	Черная рамка
		height	Высота изображения
		hspace	Отступ до текста по горизонтали
		src	URL графического файла
		usemap	Карта, связанная с изображением
		vspace	Отступ до текста по вертикали
		width	Ширина изображения
<INPUT>***	Универсальный элемент формы	checked	Элемент списка, выбранный по умолчанию
		maxlength	Максимальная длина вводимой строки
		name	См. <form>
		size	Видимая длина строки ввода
		type	Тип элемента формы

Значения параметров	Описание значений
right	По правому краю
center	По центру
—	—
—	—
—	—
Целое число	В пикселях
Целое число	В пикселях
left	По левому краю окна
right	По правому краю окна
absbottom	По нижним выступающим элементам строки
baseline, bottom**	По нижнему краю строки
absmiddle, middle	По середине строки
texttop	По верхнему краю строки
top	По верхним выступающим элементам строки
текст	Текст контекстной подсказки
Целое число	Толщина в пикселях (по умолчанию 0, для гиперссылок -2)
Целое число	В пикселях или процентах****
Целое число	В пикселях
URL	
имя	Перед именем карты ставится символ #
Целое число	В пикселях
Целое число	В пикселях или процентах****
	Для type=checkbox и type=radio
Целое число	Число символов
Целое число	Число символов
text**	Строка для ввода текста

Дескриптор	Описание дескриптора	Параметры	Описание параметров
<INPUT>***	(продолжение)		
		value	Передаваемое значение
<INS>	Вставленный текст (обычно отображается подчеркнутым)	—	—
<KBD>	Текст, введенный с клавиатуры (обычно отображается моноширинным шрифтом)	—	—
*	Элемент списка	type	Тип маркера или нумерации
		value	Номер элемента (для нумерованных списков)
<LINK>	Подключение внешнего файла	href, src	Адрес файла
		rel	Тип файла
		type	Тип таблицы стилей (для rel=stylesheet)
<MAP>	Карта изображения	name	Имя карты изображения
<MENU>	Меню	—	—
<META>***	Мета-информация о странице	content	Содержание мета-записи или http-заголовка
		http-equiv	http-заголовок
		name	Мета-запись
<NOBR>	Запрет перехода на другую строку	—	—
<OL>	Нумерованный список	type	Тип нумерации

Значения параметров	Описание значений
password	Строка для ввода пароля
checkbox	Список вариантов
radio	Список-переключатель
submit	Кнопка приема данных
image	Кнопка приема данных в виде изображения (дополнительные параметры см. <IMG>)
reset	Кнопка отмены ввода
file	Поле выбора файла
hidden	Скрытая информация
Текст	Используется как значение элемента по умолчанию
—	—
—	—
I, i, A, a или disc, square, circle	В зависимости от типа списка
Целое число или латинская буква	
URL	—
stylesheet	Таблица стилей*****
См. <STYLE>	
имя	Указывается в дескрипторе <IMG>, которому соответствует карта, с помощью параметра usemap
—	—
Текст	Точное содержание зависит от значения параметра name или http-equiv
content-type	Информация о кодировке MIME
expires	Дата следующего обновления страницы
refresh	Вместо страницы через определенный интервал должна загрузиться другая
author	Автор страницы
lescription	Описание страницы (аннотация)
eywords	Список ключевых слов
enerator	HTML-редактор, с помощью которого была создана страница
	—
	Арабские цифры
	Большие латинские цифры

Дескриптор	Описание дескриптора	Параметры	Описание параметров
<OL> (продолжение)			
			start
<OPTION>***	Элемент раскрывающегося списка	name	См. <form>
		selected или checked	Выбран по умолчанию
		value	См. <input>
<P>*	Абзац	align	Выравнивание
<PRE>	Сохранение форматирования (обычно отображается моноширинным шрифтом с сохранением разделителей)	—	—
<SAMP>	Образец вывода на экран (обычно отображается моноширинным шрифтом)	—	—
<SELECT>	Раскрывающийся список	multiple	Возможность выбора нескольких пунктов
		name	См. <form>
		size	Высота видимой части списка
<STRONG>	Выделение особо важного (обычно отображается полужирным)	—	—
<STYLE>	Таблица стилей	type	Тип таблицы
<SUB>	Нижний индекс	—	—
<SUP>	Верхний индекс	—	—
<TABLE>	Таблица	align	Горизонтальное выравнивание таблицы относительно окна
		background	Фоновый рисунок
		bgcolor	Цвет фона

Значения параметров	Описание значений
i	Малые латинские цифры
A	Большие латинские буквы
a	Малые латинские буквы
Целое число или латинская буква	Символ, с которого начинается нумерация
—	—
left"	По левому краю
right	По правому краю
center	По центру
justify	По обоим краям
—	—
—	—
—	—
Целое число	В строках
—	—
ext/css	Таблица CSS
ext/javascript	таблица стилей JavaScript
	—
	—
ft"	По левому краю
ght	По правому краю
ter	По центру
GGBB	Или мнемоническое имя

Дескриптор	Описание дескриптора	Параметры	Описание параметров
<TABLE> (продолжение)		border	Толщина рамки (по умолчанию равна 0)
		bordercolor	Цвет рамки
		bordercolordark	Цвет нижней и правой границ рамки
		bordercolorlight	Цвет верхней и левой фаниц рамки
		cellpadding	Расстояние между рамкой и содержимым ячейки
		cellspacing	Расстояние между рамками ячеек
		frame	Частичное отображение рамок ячеек
		height	Минимальная высота таблицы
		rules	Частичное отображение рамки таблицы
		width	Минимальная ширина таблицы
<TD>*	Ячейка таблицы	align	Горизонтальное выравнивание содержимого относительно границ ячейки
		background	См. <table>
		bgcolor	См. <table>
		bordercolor	См. <table>
		bordercolordark	См. <table>
		bordercolorlight	См. <table>

Значения параметров	Описание значений
Целое число	8 пикселях
#RRGGBB	Или мнемоническое имя
#RRGGBB	Или мнемоническое имя
#RRGGBB	Или мнемоническое имя
Целое число	В пикселях
Целое число	В пикселях
above	Над ячейками
below	Под ячейками
hsides	Над и под ячейками
lhs	Слева от ячеек
rhs	справа от ячеек
vsides	слева и справа от ячеек
border или box"	Вокруг ячейки (то же самое, что и просто border=1)
void	Только границы между ячейками, но не рамка вокруг всей таблицы (если указан параметр border)
Целое число	В пикселях или процентах****
all	Вокруг всех ячеек, независимо от наличия параметра border
cols	Между столбцами
groups	Между группами столбцов (см. <COLGROUP>)
юле	Вокруг таблицы
ows	Между строками
елое число	В пикселях или процентах****
eft**	По левому краю
ght	По правому краю
nter	По центру

Дескриптор	Описание дескриптора	Параметры	Описание параметров
<TD>* (продолжение)		colspan	Объединение столбцов
		height	См. <table>
		nowrap	Запрет автоматического перехода на новую строку
		rowspan	Объединение строк
		valign	Вертикальное выравнивание содержимого относительно границ ячейки
		width	См. <table>
<TEXTAREA>	Поле ввода текста	cols	Ширина
		name	См, <form>
		rows	Высота
		wrap	Переход на следующую строку
<TH>*	Ячейка заголовка	См. <TD>	
<TITLE>	Заголовок страницы	—	—
<TR>	Строка таблицы	align	См. <td>
		bgcolor	См. <td>
		bordercolor	См. <td>
		bordercolordark	См. <td>
		bordercolorlight	CM. <td>
		height	CM. <td>
		valign	CM. <td>
<U>	Подчеркивание	—	—
<UL>	Маркированный список	type	Тип маркера
<VAR>	Переменная (обычно отображается курсивом)	—	—
<WBR>***	Мягкий перенос	—	—

** Закрывающий дескриптор не обязателен*

*** Значение по умолчанию*

Значения параметров	Описание значений
Целое число	Количество объединенных столбцов
—	—
Целое число	Количество объединенных строк
top	По верхнему краю
middle	По середине
bottom	По нижнему краю
Целое число	Количество символов
Целое число	Количество строк
off	Отсутствует
virtual	Отображается, но не передается
hard	отображается и передается
-	—
	—
c**	Кружки
cle	Окружности
are	Квадратики
	—
	—

* *Непарный дескриптор*

** *При указании значения в процентах рядом с числом ставится символ %*

*** *Параметр не стандартизован. Приведено одно из значений.*

Приложение B

Коды и мнемонические имена спецсимволов

Символ	Мнемоническое имя	Код	Описание
			Неразрывный пробел
¡	¡	¡	Перевернутый восклицательный знак
$	¢	¢	Цент
£	bpound;	£	Фунт стерлингов
и	¤	¤	Денежная единица
¥	¥	¥	Иена или юань
¦	¦	¦	Разорванная вертикальная черта
§	§	§	Параграф
¨	¨	¨	Знак диерезиса
©	©	©	Знак авторского права
а	ª	ª	Признак порядкового числительного женского рода
«	«	«	Левая двойная угловая кавычка
¬	¬	¬	Знак отрицания
			"Мягкий" перенос
®	®	®	Знак зарегистрированной торговой марки
¯	¯	¯	Знак долготы над гласным
о	°	°	Градус
±	±	±	Плюс-минус
2	²	²	Цифра "2" в верхнем индексе — "квадрат"
3	³	³	Цифра "3" в верхнем индексе — "куб"
´	´	´	Апостроф
µ	µ	µ	Микро
¶	¶	¶	Знак абзаца
·	·	·	Знак умножения — "точка"
¸	¸	¸	Седиль

Продолжение табл.

Символ	Мнемоническое имя	Код	Описание
1	&supl;	¹	Цифра "1" в верхнем индексе
o	º	º	Признак порядкового числительного мужского рода
»	»	»	Правая двойная угловая кавычка
¼	&fracl4;	¼	Дробь "одна четвертая"
½	&fracl2;	½	Дробь "одна вторая"
%	Sfrac34;	¾	Дробь "три четверти"
Г	¿	¿	Перевернутый вопросительный знак
À	À	À	Латинская прописная буква А с грависом
Á	Á	Á	Латинская прописная буква А с акутом
Â	Â	Â	Латинская прописная буква А с циркумфлексом
Ã	SAtilde;	Ã	Латинская прописная буква А с тильдой
Ä	Ä	Ä	Латинская прописная буква А с диарезисом
Д	SAring;	Å	Латинская прописная буква А с кольцом
Æ	SAElig;	Æ	Латинская прописная лигатура АЕ
C	SCcedil;	Ç	Латинская прописная буква С с седилем
Ё	SEgrave;	È	Латинская прописная буква Е с грависом
Ё	SEacute;	É	Латинская прописная буква Е с акутом
	Ê	Ê	латинская прописная буква Е с циркумфлексом
	Ë	Ë	Латинская прописная буква Е с диарезисом
	Ì	Ì	Латинская прописная буква I с грависом
	Í	Í	Латинская прописная буква I с акутом
	SIcirc;	Î	Латинская прописная буква I с циркумфлексом
	Ï	Ï	Латинская прописная буква I с диарезисом
	Ð	Ð	Латинская прописная буква ETH
	SNtilde;	Ñ	Латинская прописная буква N с тильдой
	Ò	Ò	Латинская прописная буква 0 с грависом
	Ó	Ó	латинская прописная буква 0 с акутом
	Ô	Ô	Латинская прописная буква 0 с циркумфлексом
	Õ	Õ	Латинская прописная буква 0 с тильдой
	Ö	Ö	Латинская прописная буква 0 с диарезисом
	& times;	×	Знак умножения

Символ	**Мнемоническое** имя	Код	Описание
Ø	Ø	Ø	Латинская прописная буква 0 с перечеркиванием
Ù	Ù	Ù	Латинская прописная буква U с грависом
Ú	Ú	Ú	Латинская прописная буква U с акутом
Û	Û	Û	Латинская прописная буква U с циркумфлексом
Ü	«Ml;	Ü	Латинская прописная буква U с диарезисом
Ý	Ý	Ý	Латинская прописная буква Y с акутом
Þ	Þ	Þ	Латинская прописная буква THORN
ß	ß	ß	Латинское строчное двойное s
à	à	à	Латинская строчная буква а с грависом
á	á	á	Латинская строчная буква а с акутом
â	fcacirc;	â	Латинская строчная буква а с циркумфлексом
ã	ã	ã	Латинская строчная буква а с тильдой
ä	& auml;	ä	Латинская строчная буква а с диарезисом
§	å	å	Латинская строчная буква а с кольцом
æ	æ	æ	Латинская строчная лигатура ае
ç	Siccedil;	ç	Латинская строчная буква с с седилем
ë	è	è	Латинская строчная буква е с грависом
ë	é	é	Латинская строчная буква е с акутом
ë	ê	ê	Латинская строчная буква е с циркумфлексом
ë	& euml;	ë	Латинская строчная буква е с диарезисом
ì	ì	ì	Латинская строчная буква i с грависом
i	í	í	Латинская строчная буква i с акутом
î	î	î	Латинская строчная буква i с циркумфлексом
ï	ï	ï	Латинская строчная буква i с диарезисом
б	ð	ð	Латинская строчная буква eth
п	ñ	ñ	Латинская строчная буква п с тильдой
б	ò	ò	Латинская строчная буква о с грависом
б	ó	ó	Латинская строчная буква о с акутом
ô	ô	ô	Латинская строчная буква о с циркумфлексом
õ	õ	õ	Латинская строчная буква о с тильдой
б	ö	ö	Латинская строчная буква о с диарезисом
÷	÷	÷	Знак деления

Продолжение табл.

Символ	Мнемоническое имя	Код	Описание
o	soslash;	ø	Латинская строчная буква о с перечеркиванием
ù	ù	ù	Латинская строчная буква u с грависом
ú	suacute;	ú	Латинская строчная буква u с акутом
û	û	û	Латинская строчная буква u с циркумфлексом
ü	ü	ü	Латинская строчная буква и с диарезисом
ý	ý	ý	Латинская строчная буква у с акутом
þ	þ	þ	Латинская строчная буква thorn
ÿ	ÿ	ÿ	Латинская строчная буква у с диарезисом
/	ƒ	ƒ	Знак функции
A	& Alpha;	Α	Греческая заглавная буква альфа
B	SBeta;	Β	Греческая заглавная буква бета
Г	Γ	Γ	Греческая заглавная буква гамма
A	Δ	Δ	Греческая заглавная буква дельта
E	Ε	Ε	Греческая заглавная буква эпсилон
Z	SZeta;	Ζ	Греческая заглавная буква дзета
H	Η	Η	Греческая заглавная буква эта
Э	Θ	Θ	Греческая заглавная буква тета
[	Ι	Ι	Греческая заглавная буква иота
C	& Kappa;	Κ	Греческая заглавная буква каппа
ʌ	Λ	Λ	Греческая заглавная буква лямбда
1	Μ	Μ	Греческая заглавная буква мю
	Ν	Ν	Греческая заглавная буква ню
	Ξ	Ξ	Греческая заглавная буква кси
	Ο	Ο	Греческая заглавная буква омикрон
	Π	Π	Греческая заглавная буква пи
	SRho;	Ρ	Греческая заглавная буква ро
	Σ	Σ	Греческая заглавная буква сигма
	Τ	Τ	Греческая заглавная буква тау
	Υ	Υ	Греческая заглавная буква ипсилон
	SPhi;	Φ	Греческая заглавная буква фи
	Χ	Χ	Греческая заглавная буква хи
	Ψ	Ψ	Греческая заглавная буква пси

Символ	Мнемоническое имя	Код	Описание
Ω	&Omega;	&#937;	Греческая заглавная буква омега
α	& alpha;	&#945;	Греческая строчная буква альфа
β	&beta;	&#946;	Греческая строчная буква бета
γ	&gamma;	&#947;	Греческая строчная буква гамма
δ	&delta;	&#948;	Греческая строчная буква дельта
ε	&epsilon;	&#949;	Греческая строчная буква эпсилон
ζ	&zeta;	&#950;	Греческая строчная буква дзета
η	&eta;	&#951;	Греческая строчная буква эта
θ	&theta;	&#952;	Греческая строчная буква тета
ι	&iota;	&#953;	Греческая строчная буква иота
κ	&kappa;	&#954;	Греческая строчная буква каппа
λ	&lambda;	&#955;	Греческая строчная буква лямбда
μ	&mu;	&#956;	Греческая строчная буква мю
ν	&nu;	&#957;	Греческая строчная буква ню
ξ	&xi;	&#958;	Греческая строчная буква кси
ο	&omicron;	&#959;	Греческая строчная буква омикрон
π	&pi;	&#960;	Греческая строчная буква пи
ρ	&rho;	&#961;	Греческая строчная буква ро
ς	&sigmaf;	&#962;	Греческая строчная буква сигма (конечная)
σ	&sigma;	&#963;	Греческая строчная буква сигма
τ	&tau;	&#964;	Греческая строчная буква тау
υ	&upsilon;	&#965;	Греческая строчная буква ипсилон
φ	&phi;	&#966;	Греческая строчная буква фи
χ	&chi;	&#967;	Греческая строчная буква хи
ψ	&psi;	&#968;	Греческая строчная буква пси
ω	& omega;	&#969;	Греческая строчная буква омега
•	&bull;	&#8226;	Маленький черный кружок
…	&hellip;	&#8230;	Многоточие
′	&prime;	&#8242;	Одиночный штрих — минуты и футы
″	&Prime;	&#8243;	Двойной штрих — секунды и дюймы
‾	&oline;	&#8254;	надчеркивание
⁄	&frasl;	&#8260;	Косая дробная черта

Продолжение табл.

Символ	Мнемоническое имя	Код	Описание
™	™	™	Знак торговой марки
←	←	←	Стрелка влево
T	↑	↑	Стрелка вверх
→	→	→	Стрелка вправо
↓	↓	↓	Стрелка вниз
↔	↔	↔	Стрелка влево-вправо
J	↵	↵	Стрелка вниз и влево — знак возврата каретки
⇐	⇐	⇐	Двойная стрелка влево
⇑	⇑	⇑	Двойная стрелка вверх
⇒	⇒	⇒	Двойная стрелка вправо
⇓	⇓	⇓	Двойная стрелка вниз
⇔	⇔	⇔	Двойная стрелка влево-вправо
V	∀	∀	Математический знак "для всех"
∂	Spart;	∂	Частный дифференциал
∃	∃	∃	Математический знак "существует"
∅	∅	∅	Пустое множество; диаметр
7	∇	∇	Знак "набла"
	∈	∈	Знак принадлежности
	∉	∉	Знак "не принадлежит"
	Sni;	∋	Знак "содержит"
	∏	∏	Произведение последовательности
	∑	∑	Сумма последовательности
	−	−	Минус
	∗	∗	"Звездочка"
	√	√	Квадратный корень
	∝	∝	Пропорционально
	∞	∞	Бесконечность
	∠	∠	Угол
	Sand;	∧	Логическое И
	Sor;	∨	Логическое ИЛИ
	∩	∩	Пересечение
	∪	∪	Объединение

Символ	Мнемоническое имя	Код	Описание
∫	∫	∫	Интеграл
∴	∴	∴	Следовательно
∼	∼	∼	Знак тильда - 'изменяется с' — знак подобия
≅	≅	≅	Знак "примерно равно"
≈	≈	≈	Знак "почти равно — асимптотически стремится"
≠	≠	≠	Не равно
≡	≡	≡	Тождественно равно
≤	≤	≤	Меньше либо равно
≥	≥	≥	Больше либо равно
⊂	⊂	⊂	Является подмножеством
⊃	⊃	⊃	Является надмножеством
⊄	⊄	⊄	Не является подмножеством
⊆	⊆	⊆	Является подмножеством либо равно
⊇	⊇	⊇	Является надмножеством либо равно
⊕	⊕	⊕	Плюс в кружке — прямая сумма
⊗	⊗	⊗	Знак умножения в кружке — векторное произведение
⊥	⊥	⊥	Ортогонально, перпендикулярно
⋅	⋅	⋅	Оператор 'точка'
⟨	⟨	〈	Левая угловая скобка
⟩	⟩	〉	Правая угловая скобка
◊	◊	◊	Ромб
♠	♠	♠	Знак масти "пики"
♣	♣	♣	Знак масти "трефы"
♥	♥	♥	Знак масти "червы"
♦	♦	♦	Знак масти "бубны"
"	"	"	Двойная кавычка
&	&	&	Амперсанд
<	<	<	Знак "меньше"
>	>	>	Знак "больше"
Œ	Œ	Œ	Латинская заглавная лигатура OE
œ	œ	œ	Латинская строчная лигатура oe
Š	Š	Š	Латинская прописная буква S с "короной"

Окончание табл.

Символ	Мнемоническое имя	Код	Описание
Š	š	š	Латинская строчная буква s с "короной"
Ÿ	Ÿ	Ÿ	Латинская прописная буква У с диарезисом
ˆ	ˆ	ˆ	Циркумфлекс
˜	˜	˜	Малая тильда
			Узкий пробел
			Широкий пробел
			Минимальный пробел между словами
	‌	‌	Разделитель нулевой ширины
	‍	‍	Соединитель нулевой ширины
–	–	–	Узкое тире
—	—	—	Широкое тире
‘	‘	‘	Левая одиночная кавычка
’	’	’	Правая одиночная кавычка
‚	‚	‚	Нижняя одиночная кавычка
“	ildquo;	“	Левая двойная кавычка
”	Srdquo;	”	Правая двойная кавычка
	„	„	Нижняя двойная кавычка
	†	†	Крест
	‡	‡	Двойной крест
‰	‰	‰	Промилле
	‹	‹	Левая угловая одиночная кавычка
	›	›	Правая угловая одиночная кавычка
	€	€	Евро

Предметный указатель

Научно-популярное издание

Елена Леонидовна Полонская

Язык HTML
Самоучитель

Литературный редактор *Е.П. Перестюк*
Верстка *К.В. Самоцветов*
Художественный редактор *В.Г. Павлютин*
Корректоры *Л.А. Гордиенко, О.В. Мишутина*

Издательский дом "Вильямс".
101509, Москва, ул. Лесная, д. 43, стр. 1.
Изд. лиц. ЛР № 090230 от 23.06.99
Госкомитета РФ по печати.

Подписано в печать 19.05.2003. Формат 70x100/16.
Гарнитура Times. Печать офсетная.
Усл. печ. л. 25,8. Уч.-изд. л. 16,0.
Тираж 4000 экз. Заказ № 105.

Отпечатано с диапозитивов в ФГУП "Печатный двор"
Министерства РФ по делам печати,
телерадиовещания и средств массовых коммуникаций.
197110, Санкт-Петербург, Чкаловский пр., 15.

САМОУЧИТЕЛЬ

Язык HTML

Советы — это действительно мудрые и полезные сведения

Замечания — просто и доходчиво объясняют понятия и процедуры

Предостережения — помогают избежать наиболее частых недоразумений

Технические подробности — разъясняют особенности определенной темы

Книга предназначена для самостоятельного изучения базового языка создания документов в World Wide Web - языка HTML. Ее главная задача состоит в том, чтобы помочь новичкам освоиться в сложной среде современных интерактивных документов, изучить правила их оформления и имеющиеся инструментальные средства, приобреаи навыки, необходимые для успешной публикации своих материалов в Web. В книге рассматриваются основы языка HTML, особенности структуры интерактивных документов, а также самые современные Web-технологии.

Наличие у читателя каких-либо знаний из области программирования не предполагается. Что действительно нужно, так это компьютер, на котором будут создаваться HTML-документы, примеры которых приведены в книге, - только в этом случае можно будет освоить этот язык профессионально.

Благодаря лаконичному, понятному изложению материал книги будет доступен широкому кругу читателей. Книга снабжена многочисленными врезками, а также тематическими контрольными вопросами, способствующими усвоению прочитанного.

ISBN 5-8459-0466-8

домашний КОМПЬЮТЕР

ЖУРНАЛ СОВРЕМЕННОЙ СЕМЬИ

www.homepc.ru