Ваша домашняя страничка в Интернете.

Homepage, или просто «ХОМЯК»

Андрей Калиновский

- О чем писать и как начать
- Разбираемся в устройстве сайта
- Делаем HTML-страницы быстро и качественно
- Украшаем сайт, используя графику и таблицы стилей CSS
- Размещаем домашнюю страничку в Интернете и «раскручиваем» ее

bhv

Андрей Калиновский

Ваша домашняя страничка в Интернете. Нотераде, или просто «хомяк»

Санкт-Петербург
«БХВ-Петербург»
2005

УДК 681.3.06
ББК 32.973.202
К17

Калиновский А. И.

К17 Ваша домашняя страничка в Интернете. Homepage, или просто "хомяк". — СПб.: БХВ-Петербург, 2005. — 224 с.: ил.

ISBN 5-94157-581-5

Рассмотрен процесс создания персональной домашней страницы на конкретных примерах и с пояснениями. Описано, какие темы и какая информация являются в Интернете наиболее популярными и востребованными, какие программы необходимо иметь на своем компьютере начинающему web-мастеру, как правильно указывать имена файлов, как просматривать и оценивать изготовленные HTML-страницы. Рассмотрены работа с текстом, в том числе форматирование и оформление, использование различных шрифтов и внешнего вида текста, гиперссылки всех существующих типов, добавление графики и таблиц стилей. Приведены примеры полезных и простых сценариев JavaScript. Описаны процессы быстрого и продуктивного тестирования сайта для выявления недостатков и ошибок, возникающих в ходе разработки. Даются рекомендации по размещению сайта в Интернете и набор приемов раскрутки и продвижения сайта.

Для широкого круга пользователей ПК

УДК 681.3.06
ББК 32.973.202

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. гл. редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Римма Смоляк*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Татьяна Кошелева*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 08.04.05.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 18.
Тираж 3000 экз. Заказ № 983.
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

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

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

ISBN 5-94157-581-5

Оглавление

Андрей Калиновский

Ваша домашняя страничка в Интернете. Нотераде, или просто «ХОМЯК»

Санкт-Петербург
«БХВ-Петербург»
2005

УДК 681.3.06
ББК 32.973.202
К17

Калиновский А. И.

К17 Ваша домашняя страничка в Интернете. Homepage, или просто "хомяк". — СПб.: БХВ-Петербург, 2005. — 224 с.: ил.

ISBN 5-94157-581-5

Рассмотрен процесс создания персональной домашней страницы на конкретных примерах и с пояснениями. Описано, какие темы и какая информация являются в Интернете наиболее популярными и востребованными, какие программы необходимо иметь на своем компьютере начинающему web-мастеру, как правильно указывать имена файлов, как просматривать и оценивать изготовленные HTML-страницы. Рассмотрены работа с текстом, в том числе форматирование и оформление, использование различных шрифтов и внешнего вида текста, гиперссылки всех существующих типов, добавление графики и таблиц стилей. Приведены примеры полезных и простых сценариев JavaScript. Описаны процессы быстрого и продуктивного тестирования сайта для выявления недостатков и ошибок, возникающих в ходе разработки. Даются рекомендации по размещению сайта в Интернете и набор приемов раскрутки и продвижения сайта.

Для широкого круга пользователей ПК

УДК 681.3.06
ББК 32.973.202

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. гл. редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Римма Смоляк*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Татьяна Кошелева*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 08.04.05.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 18.
Тираж 3000 экз. Заказ № 983.
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

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

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

ISBN 5-94157-581-5

Оглавление

Введение

Эта книга предназначена для тех, кто хочет за относительно короткий промежуток времени создать с нуля свою домашнюю страничку в Интернете, быстро и без больших усилий освоить основы языка HTML, при помощи которого и создаются интернет-сайты.

В книге, написанной понятным языком, рассмотрен процесс создания реального web-сайта "от и до" со всеми необходимыми пояснениями. Вовсе не обязательно обладать глубокими теоретическими знаниями или богатым опытом работы. Взяв книгу в руки, вы, используя приведенные примеры, сможете быстро создать собственную web-страничку, не особо вникая в теорию.

Книга включает 11 глав, каждая из которых отражает какую-то часть из жизни сайта, начиная с его замысла, заканчивая загрузкой в Интернет и сбором статистики посещения сайта. Примеры, приводимые в книге, построены таким образом, чтобы их можно было без труда переработать и использовать в своих целях. Кроме того, примеры предназначены для более легкого понимания того, о чем читаете, на их основе вы можете делать собственные сайты, а не просто копировать предлагаемое решение.

В настоящее время большинство сайтов начинающих пользователей Интернета выглядит не очень стильно, загружается медленно, работает не оптимально и с ошибками, потому что люди, которые их создают, пользуются не теми "инструментами". Они делают сайты при помощи автоматизированных средств, не особо в них разбираясь, не представляя принципов работы в Интернете, не зная тех способов, которые позволяют сделать собственный сайт легким, красивым и удобным. Прочитав эту книгу, вы вряд ли допустите распространенные ошибки в построении своей домашней странички, поскольку теперь будете понимать, как работает ваш сайт изнутри. Наверняка вы легко справитесь с возникшими в процессе работы трудностями, будете чувствовать себя уверенно в сложном мире Интернета. Переворачивайте эту страницу и добро пожаловать в новый мир!

Глава 1

Что такое домашняя страница? Разведение "хомяков"

Что такое "хомяк" и кто такой Вася Пупкин

Разрешите представиться, меня зовут Вася Пупкин. Вы наверняка слышали про меня или читали. Я самый известный в "русском" Интернете (или, как еще говорят, Рунете) персонаж. Часто мое имя упоминают с долей сарказма, мол, Вася Пупкин — это дилетант, "чайник", который только и знает, как включается компьютер, да как выходить в Интернет. Однако это не совсем так. Вася Пупкин — просто типичный пользователь во Всемирной паутине (буквы www, с которых обычно начинается любой адрес в Интернете, расшифровываются как world wide web, т. е. "всемирная широкая паутина").

Большинство обитателей Интернета — это простые люди, достаточно далекие от компьютерного мира, для них Интернет — это развлечение, а не способ зарабатывать деньги или развивать свой бизнес. Такие люди наравне со всеми остальными могут обсуждать в форумах достоинства той или иной модели мобильного телефона, обмениваться рецептами или советами, скачивать музыку и фильмы, читать, играть и т. д. Словом, заниматься тем, для чего предназначена учебно-развлекательная часть Всемирной паутины. Вот таких пользователей и называют — Васи Пупкин.

Часто в разговорах тех, кто считает себя специалистом в компьютерной области, можно услышать нечто подобное: — "А ты представь себе, что на другом конце земного шара сидит за компьютером какой-нибудь Вася Пупкин и ломает голову о том, как же ему сделать заказ на книгу в твоем интернет-магазине" или "А ты думаешь какой-нибудь Вася Пупкин поймет, что такое отрицательное сальдо?" или "Зачем тебе нужны домашние страницы Васи Пупкина? Может лучше ориентироваться на корпоративных клиентов?". Вот так.

А таким Васям, как я, все равно, что про них говорят. Я хочу сделать свою домашнюю страничку и делаю ее. Я хочу найти информацию и нахожу ее.

И что самое интересное, все эти профессионалы работают для таких людей, как мы. Они знают, что нас много и создают свои корпоративные и поисковые порталы, интернет-магазины и игровые сайты, понимая, что просто обязаны следить за нашими предпочтениями, чтобы не потерять значительную часть своих посетителей.

Вопреки мнению подобных компьютерных специалистов, я знаю и умею достаточно много, чтобы чувствовать себя уверенно в "великом и могучем" Интернете. У меня есть собственная домашняя страничка, которую я сделал своими руками, без чьей-либо помощи, чем и горжусь. А потом друг попросил помочь сделать домашнюю страницу для него и я ему помог. Это я называю "разведением хомяков". Почему "хомяков"? В переводе с английского home — это дом, а домашняя страница — home page. Произносить "домашняя страница", а уж тем более "хоум-пэйдж" — долго и неудобно. Вот компьютерщики и придумали такое слово — "хомяк".

Вообще говоря, у "хомяка" много других названий. Его еще называют: "личный сайт", "персональный сайт", гораздо реже — "приватный сайт". Некоторые люди (вроде знаменитого российского web-дизайнера Артемия Лебедева) предпочитают писать что-нибудь особенное, например, "домстраница" (а e-mail у него — это "электропочта"). Но мне больше нравится "хомяк".

Примечание

Вообще в компьютерной отрасли так много английских терминов и названий, что их произношение на русский манер почти узаконилось, хотя и является жаргоном. Например, когда говорят "полуось", то подразумевают операционную систему OS/2, клавиатуру часто называют "клавой", а винчестер "винтом" и т. д.

В этой книге я расскажу о том, как делал "хомяка". Надеюсь, вам будет интересно, и вы легко сможете на моих примерах создать своего собственного "хомяка".

Что такое сайт, web-страница и зачем делают сайты в Интернете

Итак, давайте поговорим о том, что такое сайт и из чего он состоит. Слово *сайт (site)* в переводе с английского означает место, участок. То есть, это как бы то место во Всемирной паутине, где расположена ваша информация. А любой сайт состоит уже из отдельных web-страниц. Кстати, очень многие термины, относящиеся ко Всемирной паутине, начинаются со слова web. Например, web-дизайн, web-страница, web-программирование. Если сравнивать привычные вещи с теми, что есть в Интернете, то сайт — это как бы книга или брошюра, только в электронном виде. Как и книга, сайт состоит из отдельных страниц, на каждой из которых своя, особая информация.

Точно так же, как у книги есть оглавление, у сайта чаще всего имеется список основных разделов, иллюстрации в виде файлов с изображениями, информация об авторах и т. д.

Отличие web-сайта, например, от обычной книги в том, что он "живет" и "работает" по другим законам и принципам. Помню, когда я первый раз узнал, что такое web-сайт, то долго ходил под впечатлением. Ведь, в самом деле, нельзя же в обычной книге, нажав пальцем на какое-нибудь слово, сразу перенестись на двадцать страниц вперед, нет, придется методично перелистывать их одну за другой, отыскивая нужное место. А в Интернете этого делать не надо — там достаточно нажать кнопку мыши, наведя ее курсором на определенную часть экрана, и вы уже получаете другую информацию. В обычной книге нельзя увидеть оглавление на каждой странице, а на web-сайте это просто. Вы никогда не обнаружите внутри книги надпись о том, что показать данную страницу невозможно. Никогда не появится в книге пустой прямоугольник с крестиком в углу вместо иллюстрации. Вам не придется ждать, перевернув страницу книги, пока на ней появится текст, а в Интернете все не так. Зато в обычной жизни вы никогда не попадете за несколько секунд из Урюпинска в Токио или в Библиотеку Конгресса США. А во Всемирной паутине — запросто.

Таким образом, web-сайт — это набор web-страниц, содержащих определенную информацию, связанных между собой в основном при помощи специальных элементов — гиперссылок (подробнее эти понятия будут рассмотрены чуть позже). Каждая web-страница — это отдельный файл с расширением .html (или, реже, .htm).

Чтобы увидеть сам сайт и все страницы на нем, нужно воспользоваться специальной программой, которая называется *web-браузер* — от английских слов web (паутина) и browse (просматривать, пролистывать), т. е. это Обозреватель Всемирной паутины. Далее для простоты я буду говорить просто "браузер". Именно им чаще всего пользуются при работе в Интернете. Существует большое количество браузеров, их выпускают разные компании, очень часто в условиях жесткой конкуренции, однако смысл у всех один и тот же — показывать web-страницы. Браузеры разных производителей поразному отображают один и тот же HTML-код, из-за чего у создателей сайтов возникают определенные проблемы, поскольку одно и тоже содержимое необходимо подстраивать под наиболее распространенные модели и версии браузеров.

Каждый компьютер, с которым можно связаться во Всемирной паутине, имеет свой собственный адрес. На самом деле имеется два адреса. Один из них состоит из цифр — это четыре числа, разделенные точкой, каждое из которых может иметь значение от 0 до 255 (например, **245.24.132.54**). Однако очевидно, что такое сочетание цифр сложнее запомнить, чем какое-либо название (например, **Yahoo.com** или **Rambler.ru**). Поэтому стали использовать так называемую доменную систему имен — *DNS (Domain Name System)*.

Это способ составления адреса компьютера в Интернете при помощи так называемых доменов (в переводе с английского — сфер, владений).

Что же такое эти сферы? Домены указываются в адресе компьютера через точку. Минимальное количество таких сочетаний в интернет-адресе — два. Вот например, **yahoo.com.** — это двухуровневый домен. В этом адресе есть две сферы, о которых я говорил. Одна сфера — это **com**, говорящая о том, что данный сайт принадлежит сфере коммерческих организаций. Другая сфера — **yahoo** — говорит о том, что сайт принадлежит владельцу под условным названием "yahoo" (в данном случае это компания "Yahoo"). Уровень домена играет некоторую роль даже в смысле престижности компании или самого сайта. Если какая-либо компания имеет домен третьего уровня (например, **russian-oil.boom.ru**), говорящий о том, что она разместила свой сайт на бесплатном сервере **boom.ru**, то отношение к такой компании будет заведомо хуже — они, мол, жалеют денег на домен второго уровня (который почти никогда не бывает бесплатным). А вот если в Интернете адрес будет **russian-oil.ru**, совсем другое дело — видно, что это серьезная компания.

Большинство людей знают и умеют многое: кто-то хорошо рисует, кто-то здорово делает ремонт, кто-то пишет стихи и рассказы. У кого-то есть родственники в другой стране, и они мечтают посмотреть фотографии своего годовалого внучатого племянника. Некоторым людям просто интересно передать имеющуюся у них информацию кому-то еще. В самом деле, если ты живешь в Витебске, а читаешь в Интернете рассказ жителя Новосибирска о том, как правильно отремонтировать замок, то в этом есть что-то магическое. А как приятно человеку, когда он узнает, что его совет помог кому-то, находящемуся за тысячи километров от него.

Но вернемся к "хомякам". Личный сайт — это набор информации, которая интересна в первую очередь вам, вашим друзьям и знакомым. Это информация, которая вам близка и часто тесно связана с вашей жизнью, работой или хобби. Например, вы создали со своими ровесниками, соседями по подъезду, компьютерную сеть и вам необходимо каким-то образом общаться, рассказывать о новостях в округе, иметь "под рукой" список фильмов и музыки, существующей в вашей сети. Для всего этого можно создать web-страницы, на которых будут представлены необходимые сведения (единственное отличие такого сайта от расположенного в Интернете в том, что он находится в небольшой домашней сети).

Или другой пример: кто-то увлекается морскими свинками и хочет общаться с такими же любителями этих существ. Тогда можно создать свой сайт и назвать его "Все о морских свинках", расположить на нем истории своих питомцев, их фотографии, рассказы о возможных болезнях и о том, как их лечить. Со временем, когда вы разместите свой сайт в Интернете и зарегистрируете его на поисковых серверах, кто-нибудь на другом конце света

будет искать информацию о том, как спасти свою морскую свинку от неизвестной болезни, наткнется на ваш сайт и воспользуется вашим советом. А возможно у вас завяжется переписка, вы подружитесь, будете делиться друг с другом своими знаниями, наблюдениями.

Если вам хочется рассказать про свою семью и про то, как вы живете, поместить на своей домашней странице какие-либо фотографии, то можете сообщить адрес своего сайта тем, кому хотите. И для этого совсем не обязательно рассылать в конвертах семейные фотографии или часами рассказывать по телефону о том, как ваш сын сказал первое слово или сделал первый шаг — вы можете поместить электронный фотоальбом на вашей домашней странице, да к тому же что-то описать, а все ваши знакомые смогут все это увидеть и прочитать в любое время.

Существует еще множество причин, по которым люди пожелают создать своего "хомяка". Наверняка у вас найдется своя причина, но в одном я уверен — ваш хомяк будет очень важным и нужным. Он обязательно принесет пользу.

Однако имеются некоторые важные правила, от соблюдения которых зависит жизнь "хомяков".

На популярность вашего интернет-ресурса влияет периодичность его обновления. Обновляемая информация привлекает больше. Ведь, в самом деле, если кто-то прочел интересную статью, а через некоторое время снова попал на тот же сайт и обнаружил там еще много нового, то он непременно будет и в дальнейшем часто заходить на такой сайт, да еще порекомендует его всем знакомым. Если же там располагаются пусть даже интересные сведения, но трехлетней давности, то такой сайт никого не привлечет.

Особенная и редкая информация — бесценна. Если вы обладаете такой информацией, то существенно выигрываете при создании собственной домашней странички. В качестве такой информации может служить что угодно, вплоть до самостоятельно переведенной инструкции к стиральной машине. Люди, которые ищут такую инструкцию, будут радоваться как дети, обнаружив ее на вашем сайте.

Третья составляющая успеха — доступность сайта. Дело в том, что найти вашего "хомяка" в Интернете, где количество сайтов уже много лет назад перевалило за полмиллиона, практически невозможно. И вот здесь на помощь приходят поисковые серверы, на которых работают программы, называемые "роботы-пауки" (кто, как не пауки, могут ползать по паутине). Эти "пауки" постоянно рыскают по Интернету, отыскивая новые и измененные страницы, а также анализируя их. На основе содержимого ваших страниц на поисковом сервере ("поисковике") записывается определенная информация, которая и выдается пользователю, ищущему что-либо, связанное с темой вашего сайта.

Примечание

Естественно, что ссылка на ваш ресурс выдается вместе с другими похожими сайтами. Вот вам и преимущества редкой информации. Если информация редкая, а тема не очень обширная, то конкурентов у вас на поисковом сервере будет немного.

Чтобы адрес вашего сайта мог попасть в поле зрения различных "пауков", его необходимо зарегистрировать на большом количестве "поисковиков", необходимо "раскручивать", рекламировать, т. е. делать доступным. Вот в этом и состоят три составляющих успеха "хомяка": ценная информация, постоянное обновление и доступность.

Что такое HTML и гипертекст

Web-страница — это информация, которая оформляется при помощи специального языка — языка разметки. Этот язык не является языком программирования, он просто описывает, как следует показывать содержимое страницы. Называется этот язык HTML (Hyper Text Markup Language — язык разметки гипертекста). Языком разметки его называют потому, что он размечает текст, т. е. описывает, какие части текста и каким образом должны выглядеть.

В общем случае, если просмотреть обычный текст каким-нибудь редактором на компьютере, то будет виден лишь черный текст на белом фоне. Такую информацию можно только читать. Очень похоже на обычную книгу. Но чтобы сделать содержимое страницы гипертекстом (если переводить дословно, гипертекст — это сверх-текст), необходимо дать компьютеру определенные инструкции по обработке. Причем такие инструкции не должны быть видны при просмотре текста. Подобный эффект достигается при помощи так называемых тегов.

Тег — это и есть такая инструкция компьютеру, которая говорит, как должен выглядеть и что должен делать тот или иной элемент web-страницы. Тег представляет собой название этой инструкции и состоит из двух частей — открывающей и закрывающей. Это вроде кавычек — есть сами кавычки и есть какое-то содержимое внутри них.

Открывающий тег выглядит как угловые скобки с названием тега внутри них, например: `<i>`.

Закрывающий тег представляет собой то же самое, но только со знаком дроби перед названием тега, например `</i>`.

Я привел в качестве примера тег `<i>`, название которого образовано от слова italic (курсив). Программа знает, что любой текст, находящийся между открывающим и закрывающим тегами `<i>`, должен быть написан курсивом.

Если бы предыдущий абзац был текстом, написанным при помощи языка разметки HTML, то в браузере он бы выглядел таким образом:

названием тега внутри них, например: *Закрывающий тег представляет собой то же самое, но только со знаком дроби перед названием тега, например.* Я привел в качестве...

Смотрите, что произошло бы: браузер, увидев в языке разметки открывающую часть тега <i>, а потом найдя закрывающую часть, сам тег спрятал бы от пользователя, как служебную информацию, а то, что размещено между ними, показал бы курсивом.

Как вы понимаете, тег — это своего рода контейнер для части содержимого web-страницы. Естественно, что одни теги могут находиться внутри других, наподобие вложенных друг в друга коробок.

Текст, оформленный при помощи языка разметки HTML, называют HTML-кодом (или просто кодом, если понятно, что речь идет только о языке разметки).

Пример части кода, который описывает таблицу, приведен в листинге 1.1, из которого видно, как могут быть вложены друг в друга теги.

Листинг 1.1

```
<table>
  <tr>
   <td>
       Содержимое и текст <i>курсивом</i> внутри таблицы
   </td>
  </tr>
</table>
```

Кроме названия, у тега еще есть атрибуты — это элементы, имеющие собственное наименование и расположенные через пробел после названия тега. В отличие от названия тега, атрибуты имеют значение.

Записываются атрибуты так: имя_атрибута = "значение_атрибута" (например, цвет="красный").

Пример тега с атрибутом:

 Здесь будет текст красного цвета

Примечание

Некоторые теги могут не иметь содержимого, тогда они не разделяются на две части, а как бы объединены в одну. Например, существует тег <hr>, согласно

которому браузер должен показать горизонтальную черту. Выглядит он как <hr>, т. е. только открывающая часть тега. Также некоторые атрибуты могут не иметь значения. Значением таких атрибутов является как бы само их присутствие, например, атрибут nowrap дает браузеру инструкцию о том, что содержимое тега не должно переноситься на следующую строчку, даже если оно не помещается в предоставленное ему пространство.

Как вы видели на примере листинга 1.1, HTML-код отформатирован, т. е. написан определенным образом при помощи отступов и переносов на следующую строчку. Это делается лишь для удобства человека, который будет смотреть этот код. На самом деле, даже если он будет написан в одну строчку без переносов, браузер также хорошо поймет его.

Уникальность языка HTML состоит еще и в том, что он кроссплатформенный. Это сложное слово означает, что в любой установленной операционной системе (как вы знаете, это может быть Microsoft Windows, Linux, UNIX, Macintosh, OS/2 и т. д.) HTML-код будет одинаково хорошо пониматься, независимо от строения файловой системы.

Кроме того, восприятие системой названий тегов и их атрибутов не зависит от того, какими буквами (заглавными или строчными) они набраны в HTML-коде. Это значит, что если вы напишите <TABLE border="1">, то написанное будет воспринято системой точно так же, как если бы вы написали <table BORDER="1">.

Теперь о том, что такое гипертекст. В книгах, газетах и тому подобном информация имеет линейную структуру, т. е. она сосредоточена в каком-то одном источнике, не связанном с другими подобными, и показывается чаще всего последовательно. Исключением является роман-игра Гарри Гаррисона о Стальной крысе, оформленная на основе принципа, заложенного в гипертексте. Я считаю, что это вообще первый пример гипертекста. В этой книге пронумерованы небольшие разделы, в каждом из них описана ситуация, в которой оказываетесь вы — главный герой, и даны варианты ваших действий с номером раздела, к которому необходимо перейти дальше. То есть вы читаете не от начала до конца ("по прямой"), а перескакиваете с раздела на раздел, с начала книги в середину, с середины в конец, а потом снова в середину или начало. Так же устроен и гипертекст, только вместо номеров разделов используются так называемые гиперссылки, которые сами переносят вас на другую web-страницу.

Как же устроена работа с гипертекстом? Все достаточно просто.

Во-первых, два слова об Интернете, да и о любом другом "нете". Net в переводе с английского означает сеть. Это может быть и Интернет и, как я говорил ранее, маленькая сеть для соседей по подъезду. В любом случае сеть — это некоторое количество соединенных между собой компьютеров. Многие из современных сетей работают по технологии клиент-сервер.

Сервер — это компьютер, который чаще всего бывает достаточно мощным, хранит большое количество информации и управляет взаимодействием нескольких компьютеров между собой.

Клиент (или так называемый клиентский компьютер) — это чаще более слабый компьютер, который лишь обращается к серверу за нужной информацией и показывает ее пользователю. Если вы работаете на компьютере дома, а не в сети, то ваш компьютер может быть и клиентом и сервером одновременно.

Существуют несколько способов связи компьютеров между собой. Эти способы называются протоколами связи. Например, FTP (File Transfer Protocol — протокол передачи файлов), IMAP (Internet Mail Access Protocol — протокол доступа к электронной почте) и т. д. Для работы с гипертекстом также имеется свой протокол — HTTP (Hyper Text Transfer Protocol — протокол передачи гипертекста). Нас интересует именно он, потому что HTTP является основой Всемирной паутины. При помощи этого протокола происходит работа с гипертекстом.

Допустим, вам необходимо найти информацию в электронной библиотеке. Ваш компьютер в данном случае является клиентом. Вы открываете браузер, набираете в нем электронный адрес библиотеки. Если ваш компьютер подключен к Интернету, то с него идет запрос на компьютер, хранящий эту библиотеку (в данном случае, как говорят, на сервер электронной библиотеки). На сервере нужная вам страница отыскивается и при помощи установленной связи по протоколу HTTP передается назад на ваш компьютер. Браузер принимает эту страницу, которая содержит HTML-код, и показывает ее вам. Если на странице есть гиперссылки, то, нажав на одну из них, вы снова обращаетесь к серверу, но уже за другой страницей — той, к которой ведет эта ссылка. Сервер снова возвращает вам соответствующую HTML-страницу и т. д. Чтобы работать по протоколу HTTP, на сервере должно быть установлено специальное программное обеспечение. Если же вы работаете с HTML и гипертекстом на своем компьютере, то вам нужен только браузер.

Что нужно знать и уметь, чтобы создать свой личный сайт

На самом деле для начала знать и уметь нужно не так уж и много, всему остальному можно научиться в процессе работы. Для создания простого личного сайта вам необходимо уметь пользоваться компьютером, т. е. уметь включать его, запускать приложения, находить и сохранять файлы, копировать их на дискету или записывать на CD-ROM. Также необходимо понимать, что такое файл и имя файла, технология клиент-сервер, язык разметки и гипертекст. Вот, пожалуй, и все. Всему остальному вы научитесь постепенно, в т. ч. и при помощи этой книги.

Глава 2

Работа с будущим содержимым сайта

О чем будет сайт — выбор темы

Первое, о чем хочется сказать, — это то, что выбор темы имеет большое значение для будущего "хомяка". В зависимости от темы сайта, его предназначения формируется аудитория ваших будущих посетителей. Очень часто web-дизайнеры критикуют домашние странички Васи Пупкина — они, дескать, некрасивые, сделаны непрофессионально, работают медленно. Но если на этих страничках важная и интересная информация, то никто не обращает внимания на внешнее оформление сайта. Когда вы торопитесь на вокзал, вам все равно, на чем добираться — на "Москвиче" тридцатилетней давности или на BMW — лишь бы успеть. Вот и в нашем случае прежде всего стоит подумать о тех целях, которые вы преследуете, создавая свой личный сайт, — хотите прославиться, рассказать о себе, заработать денег или что-либо еще.

Тема номер один — "о себе". Это сайт, содержащий информацию о вашей семье, жизни, о ваших интересах, увлечениях, мечтах и т. п. Так уж устроена жизнь, что кроме узкого круга родственников, знакомых и друзей ваши личные дела никому не интересны. Поэтому, если вы хотите создать своего "хомяка" с темой "о себе", то должны помнить, что кроме людей, которые вас знают, некому будет смотреть на вашу домашнюю страничку. Чаще всего такую тему и выбирают для того, чтобы можно было дать знакомым ссылку на свой сайт. Пусть бабушка из Нью-Йорка потешится, глядя на рисунки внука, или брат, который служит в армии и не смог попасть на вашу свадьбу, посмотрит электронный фотоальбом. Как я уже говорил ранее, такого рода сайты не надо регистрировать в поисковых системах или "раскручивать", добиваясь известности, поскольку у них нет конкуренции — такой информации, кроме как у вас, нет ни у кого.

Тема номер два — "мое творчество" (например, вы пишете стихи или рассказы, рисуете или фотографируете). Такая информация уже может быть интересна не только кругу ваших знакомых — в мире найдется достаточное

количество людей, которые могут прийти в восторг от ваших стихов или черно-белых фотографий старой мебели. Поэтому можете смело браться за изготовление такого "хомяка". Единственное, что вас может смутить, — это незащищенность авторских прав. Ваш сайт будет находиться в открытом доступе и вместе с теми, кого вы хотите заинтересовать своим творчеством, могут оказаться люди, нечистые на руку, которые постараются присвоить себе ваши заслуги и выдать их за свои. Когда-то я читал историю любителя рисовать карикатуры, который случайно обнаружил свои картинки на чужом сайте, под другими названиями и без ссылок на то, откуда они взяты.

Тема номер три — "развлекательный сайт" (анекдоты, картинки, рассказы, видео). Как и в большинстве других сфер жизни, во Всемирной паутине людям интересно то, что является необычным. Просто набор анекдотов — это, конечно же, любопытно, но страничка, на которой любой посетитель может рассказать и свой собственный анекдот, историю, случай из реальной жизни, афоризм и тому подобное будет гораздо привлекательнее. Это могут быть и так называемые "on-line" игры, и общение в "чате" (т. е. переписка по сети в режиме реального времени), и психологические тесты, и составление гороскопов — кому что интересно.

Тема номер четыре — "купите у меня что-нибудь". Со временем у некоторых людей может накопиться большое количество информации, которую они захотят продать. Наивно поступают люди, предлагающие купить то, что можно найти бесплатно. Не все в Интернете такие предприимчивые — некоторым просто интересно выставить перед всем миром свои сокровища, "нажитые непосильным трудом", тем более что электронные сокровища не уменьшаются. Не так давно я столкнулся с чудаком, предлагающим купить у него за два "web-доллара" (это такие электронные деньги) программу "Гороскоп", кстати, ему и не принадлежащую. Эту же программу совершенно бесплатно я нашел в Интернете буквально через минуту. Если вы хотите что-то продавать, подумайте, нужно ли это людям. Предмет продажи вовсе не обязательно должен быть чем-то осязаемым. Вы можете, например, сводить вместе тех, кому требуются услуги по разработке сайтов, и тех, кто их разрабатывает. Вы можете сканировать книги, которые есть у вас, а потом "обменивать" их электронную версию на какие-нибудь программы, которые имеются у других. Все зависит от вашей фантазии и результатов запроса "заработок в Интернете" на каком-нибудь поисковом сервере.

Тема номер пять — "информация" (это сайты, содержащие информационные и обучающие материалы). За ними "охотятся" многие и довольно-таки часто. Информация — это та основа, из которой вырос сам Интернет, изначально задуманный как военная разработка для обмена информацией и повышения надежности ее хранения. Найдя в Интернете интересный сайт, опытный пользователь непременно поставит на него "закладку" (т. е. запишет при помощи своего браузера адрес сайта со своей пометкой). Разработчики таких сайтов на своих web-страничках помещают самую разнообраз-

ную информацию — от статей по web-дизайну до правил этикета или способов создания воздушного змея.

Мой личный сайт как раз из подобного разряда. Я решил создать сайт (или, как иногда говорят, ресурс) о ремонте и домашних делах. Но, поскольку я не профессиональный строитель и не профессиональный повар, то не пишу обо всем и много. Создание такого глобального информационного ресурса потребовало бы значительного времени и сил. Я решил, что буду писать только о том, что узнал или сделал сам, своими руками. Ведь этому учат в первую очередь, например, на курсах писательского мастерства — надо писать только о том, в чем хорошо разбираешься и тогда успех обеспечен. Если вы сантехник и хотите создать фантастический роман, напишите про сантехника с Венеры, работающего на межгалактическом корабле. Я использую статьи и рассказы других авторов со ссылкой на их сайты и с их позволения, чтобы расширить тот объем информации, который предоставляю посетителям моего "хомяка". Обычно это получается просто — достаточно письма электронной почты, а иногда на сайтах пишут, что если хотите использовать мою информацию, то используйте, но дайте соответствующую ссылку. В отличие от "творческих" сайтов, тут с авторскими правами обычно проблем не возникает.

Собираем информацию для работы над сайтом

Ту информацию, которую предполагаете разместить на своем сайте, необходимо собрать вместе и привести к единому виду. Для этого существует масса способов. Вот у меня, например, часть информации имеется в виде рукописных записок и конспектов. Говоришь, бывает, со знакомым специалистом о том, как починить кран или покрасить подоконники, и записываешь в свою рабочую тетрадь. Часть информации находится в моей голове (я просто помню, как делал что-либо), часть — в виде напечатанных на принтере статей, рецептов, чертежей (а первоисточники этих распечаток давно канули в лету или изменились до неузнаваемости). Какие-то вещи имеются в виде файлов, созданных в Microsoft Word и Microsoft Visio. Как видите — весьма разнообразные носители информации. Что же мне делать, чтобы привести их все к единообразию?

Прежде чем начать работу с информацией для сайта, необходимо оценить полноту не только текстовой информации, но и файлов, программ, картинок, которые вы планируете разместить на нем. Нужно оценить объем существующей информации, и если она слишком велика, то выбрать самое важное и интересное, а если слишком мала — поискать еще или повременить, хотя все зависит от специфики сайта. Если материал интересный и ценный, то начинать можно всего лишь с нескольких статей, главное — не браться поначалу за крупные, крутые проекты и не кричать всему миру

о том, что у вас самый полный информационный ресурс по данной теме, дабы вас не засмеяли и чтобы вы не утратили рейтинг в будущем. И еще необходимо помнить — для информационных сайтов дизайн далеко не главное и не стоит на нем "зацикливаться". Есть набор простых правил, соблюдение которых позволит вам смело смотреть в глаза любому дизайнеру, но об этом позже.

Итак, вы для себя отобрали ту информацию, которая должна присутствовать на вашем сайте. Теперь необходимо переходить к ее обработке.

Для начала следует решить, как должна выглядеть в конечном итоге систематизированная информация. Во-первых, т. к. она впоследствии превратится в HTML-код, то будет потеряно всякое постороннее форматирование и все преобразуется в обыкновенный текст. Во-вторых, некоторые файлы содержат (или будут содержать) рисунки. Эти рисунки необходимо собрать вместе в отдельном каталоге. В-третьих, возможно будут использоваться таблицы, которые необходимо будет оставить нетронутыми до момента их "переселения" на web-страницы. Мне представляется целесообразным оставлять всю информацию в виде файлов текстового редактора Microsoft Word (за исключением отдельно находящихся картинок или схем, сделанных в каком-нибудь особенном программном средстве вроде Microsoft Visio, AutoCAD и др.). Конечно же, потом, когда сайт будет завершен, новый текст можно готовить уже в HTML-коде. В процессе работы нужно выработать для себя некоторые правила, по которым вы будете оформлять свои подготовленные doc-файлы. Например, я заранее придумываю заголовок (допустим, "Покраска подоконников") и название подзаголовков, если они есть (например, "Снятие старой краски", "Выравнивание поверхности подоконника", "Использование шпатлевки по дереву" и т. д.), а соответствующий файл позднее превращаю в отдельную web-страницу.

В первую очередь я, конечно, разбираюсь с той частью, которая уже существует в электронном виде. Для дальнейшей работы необходимо извлечь рисунки из файлов Microsoft Word (далее я буду говорить просто Word), а там, откуда я их убираю, надо оставить пометку, т. е. написать что-то вроде "вставить рисунок brick.gif". На всякий случай можно сохранить исходный документ в виде web-страницы. Я из меню **Файл** запускаю команду **Сохранить как**, выбираю нужный каталог (например, /sand.files) и сохраняю документ в виде файла с расширением .htm (например, sand.htm). Вот как раз внутри выбранного каталога и появляются рисунки из Word-овского документа (правда, иногда их качество оставляет желать лучшего). Еще необходимо помнить, что прежде всего надо проверить орфографию и синтаксис, потому что грамматические ошибки — плохой тон везде, а особенно во Всемирной паутине. Это необходимо делать даже с текстовыми файлами (расширение .txt). Откройте их при помощи Microsoft Word и проверьте на наличие ошибок.

Теперь мои записи. Их придется набирать вручную в текстовом редакторе Word. Если вы умеете печатать только одним пальцем, то возиться, к сожалению, придется долго. Это самая длительная и неинтересная процедура превращения информации в нужный вид. Зато это, чаще всего, самые ценные сведения и они того стоят. С чертежами и рисунками, сделанными от руки, можно поступать по-разному. Можно их аккуратно перечертить, а потом отсканировать. Можно, если вы обладаете таким умением, воссоздать в каком-нибудь графическом или текстовом редакторе и сохранить как картинку. Все промежуточные результаты такого рода (например, чертеж, сделанный в CorelDRAW) необходимо также сохранить, чтобы в случае обнаружения ошибки в рисунке, ее можно было быстро поправить. Почти так же, как и со своими записями, я поступаю с тем, что хранится "в моей голове". Все эти сведения я сначала просто записываю в виде списка того, что должен напечатать, а потом сажусь и описываю все, что помню.

Ту часть информации, которая хранится у меня в виде напечатанных на принтере статей, я, естественно, не собираюсь набирать повторно — для этого существуют иные средства. При помощи программы ABBYY FineReader я сканирую текст и рисунки с этих распечаток. При помощи того же ABBYY FineReader компьютер превращает отсканированное изображение символов, что были на бумаге, в нормальный текст, который можно сохранить как файл. Тут все, конечно же, зависит от качества принтера (или пишущей машинки), на котором была напечатана эта информация. Чем качественнее напечатаны листы, тем меньше ошибок окажется при распознавании текста. В любом случае такого рода тексты впоследствии необходимо внимательно проверить на наличие ошибок, потому что пятнышко на бумаге может быть воспринято программой распознавания как точка, буква "н" может быть понята как символ "и" и т. д. Но это все равно намного быстрее, чем набирать вручную. На сканирование и исправление одной страницы я трачу примерно десять минут, а чтобы набрать ее на клавиатуре необходимо минут сорок.

Делим информацию на части — будущие разделы сайта

Будем считать, что будущая информация для сайта собрана (ее еще называют контент сайта, от английского слова content — содержание). Теперь мы должны эту информацию систематизировать и "разложить по полочкам", равносильно тому, как в библиотеках расставляют книги: в одном шкафу фантастика, в другом — приключения, в третьем — любовные романы и т. д., только вместо шкафов будут каталоги. Каждый каталог условно считается разделом сайта. Каталоги могут быть верхнего уровня (т. е. относиться к главным разделам сайта), а могут быть и второго уровня (т. е. входить

внутрь уже самих разделов сайта). Это обычная иерархическая структура. Выглядит она примерно так, как представлено на рис. 2.1.

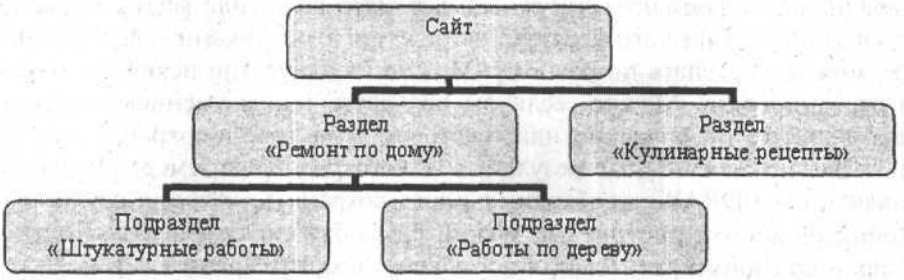


Рис. 2.1. Предварительная зарисовка иерархической структуры разделов на сайте

Важным свойством сайта является количество разделов — так называемая ширина и глубина иерархической структуры. Под шириной понимается количество разделов верхнего уровня, а под глубиной — количество уровней подчиненных подразделов. Близким примером такой структуры является файловая система. Например, у меня, помимо системных файлов и программ, на диске C располагаются каталоги Work, Install, Private, Games и Projects. Это значит, что, условно говоря, ширина моего корневого каталога — 5 элементов. Теперь возьмем, например, программу homesite45.exe (из каталога Install/Editors/HTML/HomeSite) — она, как видим, располагается в каталоге с глубиной 4 элемента.

Необходимо стараться поддерживать оптимальный баланс ширины и глубины, чтобы не случилось такой ситуации, когда на верхнем уровне сайта много разделов (ведь хочется показать сразу и все), а внутри каждого раздела почти ничего нет. Посетителям будет неудобно ориентироваться на таком сайте — у них "глаза разбегутся". Однако если вы сделаете всего лишь два-три раздела верхнего уровня, а все остальное "запрячете" внутрь, то с первого взгляда будет непонятно, о чем ваш сайт и где что искать.

Конечно же, нет формулы для вычисления нужной ширины и глубины, но, видимо, ложный стандарт лучше, чем его отсутствие. Еще раньше существовал миф о том, что есть такое магическое число 7 ± 2. Считалось, что количество разделов верхнего уровня должно находиться в пределах от 5 до 9, а 7 разделов — почти оптимально. Под это утверждение даже подводили теорию и исследования о том, что человек может одновременно воспринимать не более семи элементов, но постепенно время и специалисты опровергли нерушимость этого принципа. Однако самое смешное, что чаще всего на небольших сайтах, вроде "хомяков", именно такое магическое число разделов и получается.

Все разделы, которые образуются после такой систематизации информации, относятся к так называемой "навигации" или "навигационной модели" сайта.

Навигация — это те инструменты для перемещения по содержимому сайта (при помощи гипертекста), которые разработчик сайта предлагает посетителю. Основные разделы "хомяка" называются первичной или главной навигацией. Как вы, наверное, поняли, есть еще и вторичная навигация — это разделы, не имеющие прямого отношения к содержимому сайта, но присутствующие практически на каждом сайте. Ведь вы не просто так делаете своего "хомяка" — вы хотите, чтобы люди узнали про вас, могли с вами связаться, высказать свое мнение о вашем сайте и о том, что бы им хотелось на нем видеть еще и т. д. Именно для этого и существуют такие разделы, как "О себе", "Связаться со мной", "Гостевая книга".

Названия ваших разделов должны быть понятны всем. Не ленитесь их придумывать. Иногда на обдумывание названия может уйти от нескольких часов до нескольких дней. Ничего страшного — "как вы лодку назовете, так она и поплывет".

Вам же хорошая проработка структуры позволит легче поддерживать сайт в дальнейшем и тратить меньше времени на изменения, а особенно на пополнение.

То, что получилось у меня после проработки структуры моего сайта, представлено на рис. 2.2.

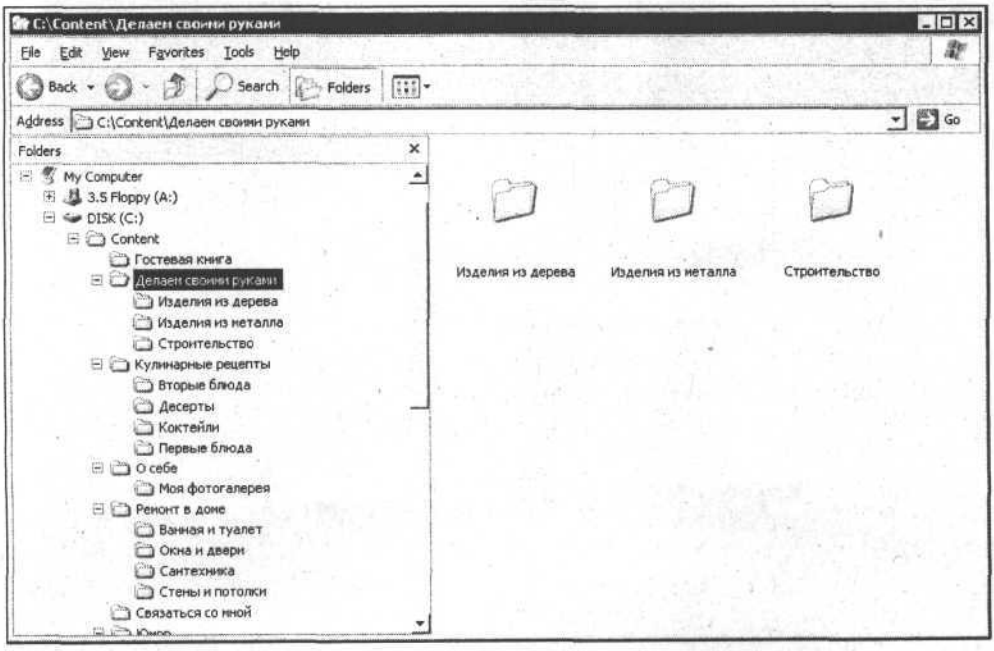


Рис. 2.2. Иерархическая структура разделов на сайте

Глава 3

Из чего же, из чего же сделаны наши "хомяки"

Подготовка к написанию HTML-страниц, редактор и рабочее место

Чтобы начать писать HTML-код, необходимо использовать HTML-редактор, т. е. программу, в которой вы будете создавать ваши web-странички. Вы уже знаете, что представляет собой язык разметки. Это по большому счету обычный текст. Для его создания может использоваться несколько разновидностей редакторов. Первая — Блокнот (он же Notepad) — это стандартное средство редактирования простых текстов, встроенное в операционную систему Microsoft Windows. Найти блокнот очень просто: **Пуск | Программы | Стандартные | Блокнот**. Многие считают, что Блокнот — это инструмент для фанатиков, которым нравятся трудности, или для начинающих, которые хотят написать несколько строчек кода, запустить браузер, увидеть, что у них получилось, а потом воскликнуть: "Ух! Кру-у-уто!". Для тех, кто всерьез решил заняться разведением "хомяков", возможностей Блокнота вскоре окажется мало, поскольку в нем минимум функций, весь текст одинакового цвета и поэтому работать крайне неудобно. HTML-код, написанный в Блокноте, представлен на рис. 3.1.

Лично мне гораздо больше нравится использование файлового менеджера Far. Мало того, что он сам по себе удобен в работе с файлами, когда вы возитесь со своим "хомяком", но там ко всему прочему есть простой текстовый редактор, в котором можно писать код. Far отличается от других, существующих ныне популярных файловых менеджеров, именно возможностью легко переходить в текстовый редактор и работать с содержимым текстовых файлов. На рис. 3.2 представлены рабочие панели в файловом менеджере Far.

Примечание

Far является свободно распространяемым программным продуктом, который можно скачать из Интернета с сайта **www.rarsoft.com**. Поскольку автор этого

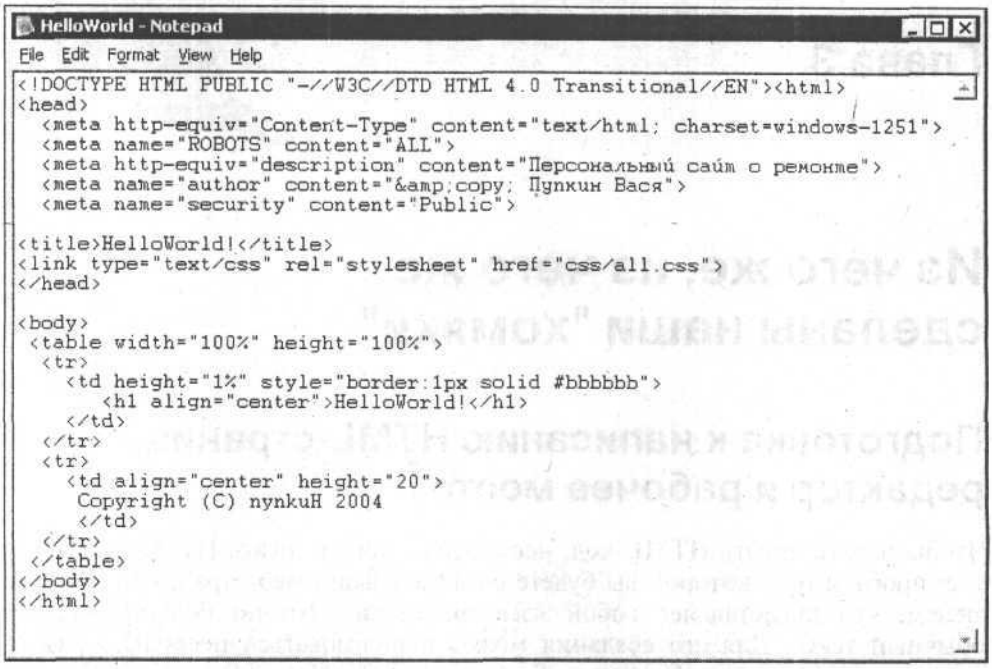


Рис. 3.1. HTML-код, написанный в Блокноте

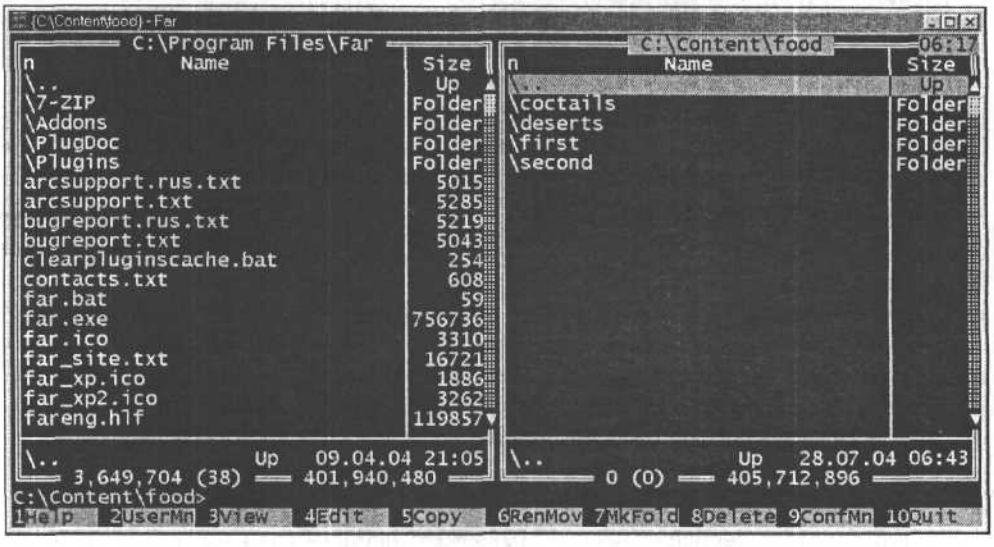


Рис. 3.2. Рабочие панели в файловом менеджере Far

продукта, Евгений Рошал, родом из Советского Союза, то он сделал своеобразный подарок всем жителям бывшего СССР, которые могут использовать его программу Far. Это бесплатная xUSSR регистрация. Чтобы зарегистрировать ваш Far, наберите в командной строке **far –r**, в появившемся диалоговом окне необходимо ввести в качестве имени **xUSSR регистрация**, а в качестве регистрационного кода название текущего дня недели маленькими буквами (например, **вторник**).

Кроме этого, для Far-а имеется большое количество так называемых подключаемых модулей (plugin), которые обеспечивают комфортную работу даже в таком простом редакторе. Главный такой модуль — это Colorer (Раскрашиватель). При его подключении код программы, написанной практически на любом современном языке программирования, подсвечивается разными цветами (например, на синем фоне белым цветом подсвечиваются открывающие и закрывающие части тегов, обычный текст можно сделать светло-голубым, атрибуты — желтым и т. д.), что позволяет очень быстро читать код и находить в нем нужное место. На рис. 3.3 представлен HTML-код, набранный в редакторе Far-а (к сожалению, на черно-белых картинках не видна подсветка текста).

Рис. 3.3. HTML-код, набранный в редакторе Far-а

Есть еще одна разновидность редакторов HTML-кода — так называемые визуальные редакторы (визуальные — потому что вы можете создавать в них web-страницы, не набирая на клавиатуре код, а при помощи специальных графических компонентов, т. е. визуально). Вы делаете что-либо (например, добавляете ячейки в таблицу), нажимая для этого на специальную кнопку, а в другом месте (чаще всего на соседней вкладке редактора) по ходу ваших

действий сам по себе создается HTML-код. Работают многие из таких редакторов по принципу *WYSIWYG What You See Is What You Get* (что ты видишь, то и получишь). Это значит, например, что нечто, сделанное внутри редактора в режиме визуального создания страницы, будет точно также выглядеть и в браузере. Главная проблема таких редакторов в том, что они лишь изредка умеют писать качественный код. Чаще всего они добавляют лишние теги, невообразимо форматируют, делают страницы более тяжелыми для браузера и т. д. Но, с другой стороны, такие редакторы ускоряют процесс разработки макетов и заготовок страниц, существенно упрощая создание HTML-кода для новичков. Для примера я буду использовать редактор Home Site (раньше он принадлежал компании Allaire, а потом его купила другая компания — Macromedia). Пример работы в визуальном редакторе представлен на рис. 3.4 (нажав на нужную кнопку, вы удалите ячейку таблицы, в которой расположена надпись **HelloWorld!**, даже не видя HTML-кода этой страницы, который сам изменится автоматически).

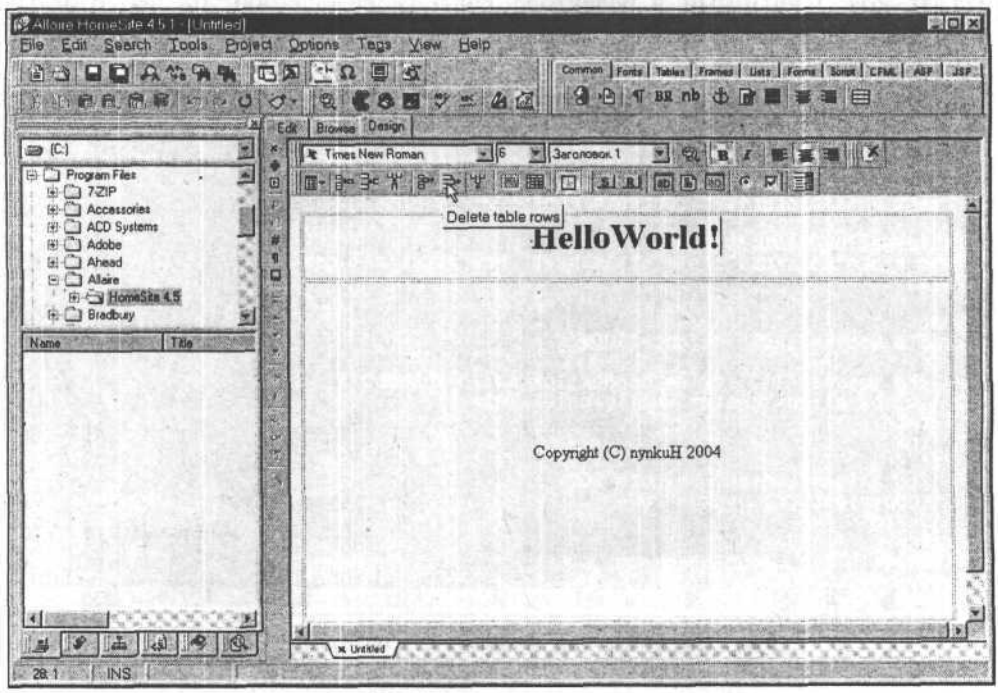


Рис. 3.4. Пример работы в визуальном редакторе

Кроме этого, визуальные редакторы обладают набором функций, весьма полезных на первых порах, — это и система помощи, и так называемая автоматическая вставка тега (т. е. вам достаточно начать набирать на клавиатуре открывающую часть тега, как редактор сразу же предложит варианты суще-

ствующих тегов). На рис. 3.5 представлена такая ситуация: только я набрал на клавиатуре угловую скобку, а передо мною уже появился выпадающий список, из которого можно выбрать нужное название тега. А когда я поставлю вторую угловую скобку, то автоматически появится закрывающая часть тега.

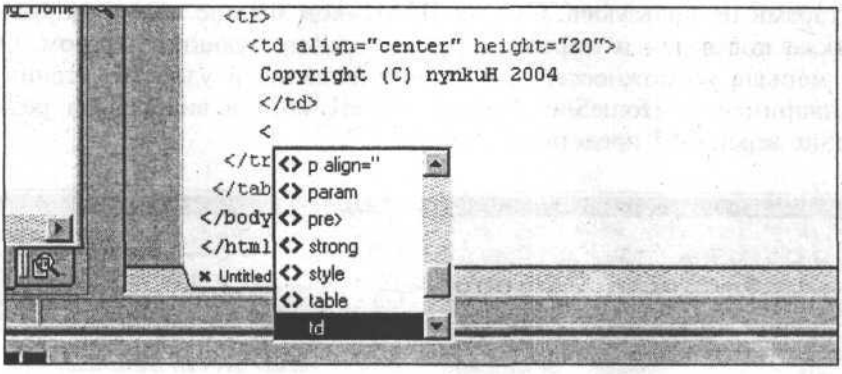


Рис. 3.5. Выпадающий список с набором тегов

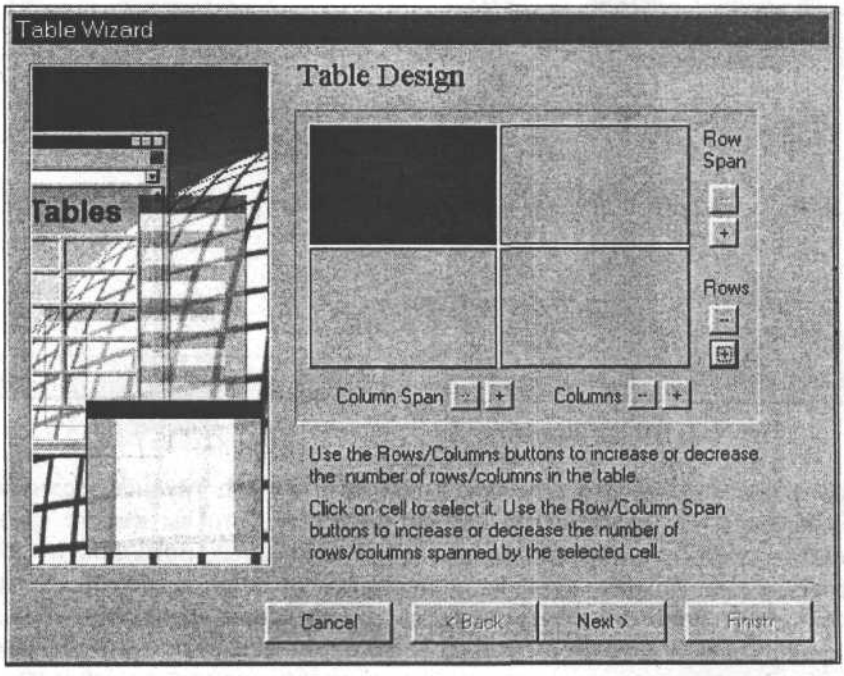


Рис. 3.6. Мастер создания таблиц в визуальном редакторе HomeSite 4

К сервисным возможностям этого редактора можно отнести и совокупность различных Мастеров (Wizard), которые позволяют опять же визуально создавать таблицы, рамки, кнопки и т. п. На рис. 3.6 представлен Мастер создания таблиц в визуальном редакторе HomeSite 4.5. Вы можете задавать конфигурацию таблицы так, как вам нужно, а HTML-код создастся автоматически.

Есть еще масса полезных функций в подобных редакторах, но лично я этими редакторами не пользуюсь. Сам же HTML-код внешне похож на код в Far. Он также подсвечен и отформатирован соответствующим образом. Однако у Far меньше возможностей для форматирования и удобства чтения кода, чем, например, в HomeSite. Пример HTML-кода в визуальном редакторе HomeSite версии 4.5 представлен на рис. 3.7.

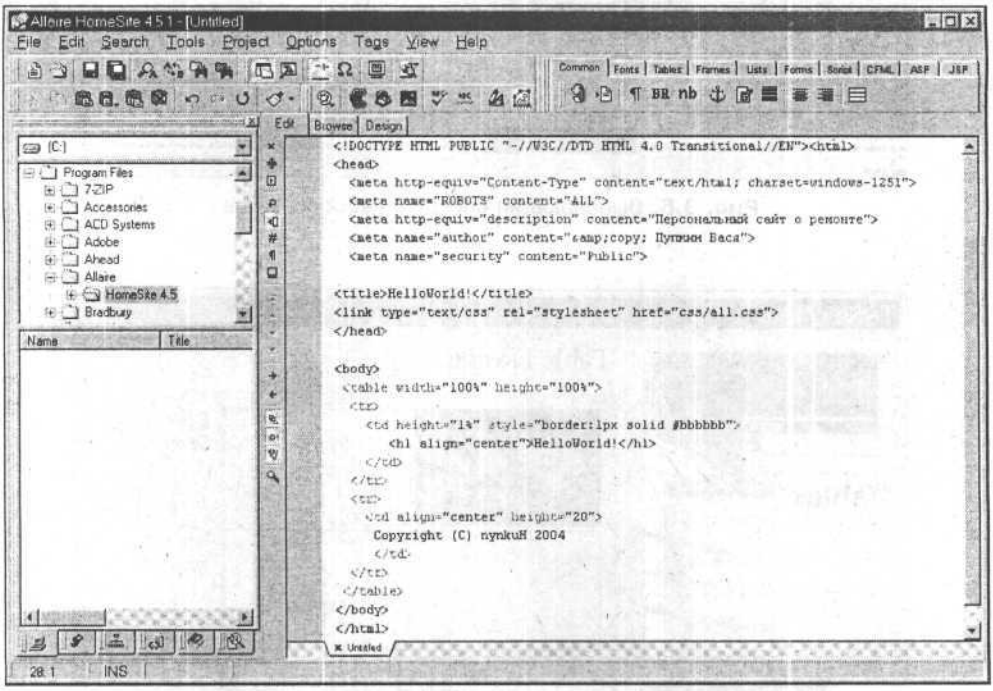


Рис. 3.7. HTML-код в визуальном редакторе HomeSite версии 4.5

Теперь я поясню, почему научиться работе с HTML труднее, если пользоваться визуальным редактором. Представьте себе, что вы имеете самые современные строительные материалы — от подвесных потолков и сайдинга до черепицы на крышу, которая защелкивается одна за другой. У вас есть превосходный дорогой инструмент со множеством функций — от дрели с перфоратором и шуруповертом до пистолета для забивания гвоздей. Все замечательно, но с этим инструментом вы толком не умеете работать. Пред-

ставьте себе такую ситуацию: вы нашли у кого-то на сайте интересную идею, хотите использовать ее у себя, но в несколько измененном виде. Что вам нужно сделать? Встать курсором на этой странице, нажать правую кнопку мыши и выбрать пункт **Показать в виде HTML**. Тогда откроется тот самый Блокнот, в котором и будет показан HTML-код данной страницы. Если вы привыкли пользоваться только визуальной частью своего редактора, то ничего не поймете из того, что увидите в Блокноте.

Через некоторое время после того, как создадите какую-то из своих страниц, вы можете совсем забыть, как она устроена, а при помощи визуального редактора практически невозможно будет с этим разобраться. У вас будут появляться какие-то лишние ячейки таблиц, границы разных элементов будут сливаться между собой, что-то спрячется от невооруженного взгляда. Уж поверьте, я сам не раз поначалу "наступал на эти грабли". Приходилось переходить на вкладку с исходным кодом страницы и вчитываться в код, с трудом вспоминая, что к чему. Я же сейчас предлагаю вам другой путь. Если вы сначала изучите основы HTML и будете работать именно с кодом, то потом для вас не составит труда разобраться в любой самой сложной странице. Может быть, позже вы попытаетесь использовать какой-нибудь визуальный редактор, но тогда он вам уже не понадобится и, возможно, будет несколько раздражать.

Теперь необходимо подготовить файловую систему компьютера к работе над "хомяком". Во-первых, нужно создать свой рабочий каталог (т. е. место, где будет вестись работа над сайтом). Это место должно быть удобным и структурированным, чтобы как можно меньше времени тратить на возню с файлами. Я назвал свой рабочий каталог /RepairSite. Внутри него создал такую структуру подкаталогов, чтобы потом не запутаться: каталог /doc (в нем будет храниться вся подготовленная ранее информация для будущих HTML-страниц), внутри каталога /doc будет еще один подкаталог со всеми рисунками, которые понадобятся, его я назову /img. Теперь необходимо создать подкаталог /html, в котором будут готовые HTML-страницы. Если мне понадобятся еще какие-нибудь подкаталоги, например /tools или /download, я сделаю их позже. Если какие-то файлы могут пригодиться, но пока непонятно, куда их отнести, то для таких файлов я использую подкаталог /Unsorted.

Теперь главная цель — каталог /html. В нем фактически отражена та же иерархическая структура сайта, которая описана в *гл. 2*, но только в файловой системе, с англоязычными названиями и в виде каталогов. Выглядит у меня это таким образом, как представлено на рис. 3.8.

Подобная структура создается для того чтобы вашим посетителям было легче ориентироваться на данном сайте, а вам легче им управлять. Только представьте себе, как сложно было бы что-то отыскать на жестком диске, если бы все находилось в одном каталоге: и фотографии, и документы, и фильмы, и системные файлы.

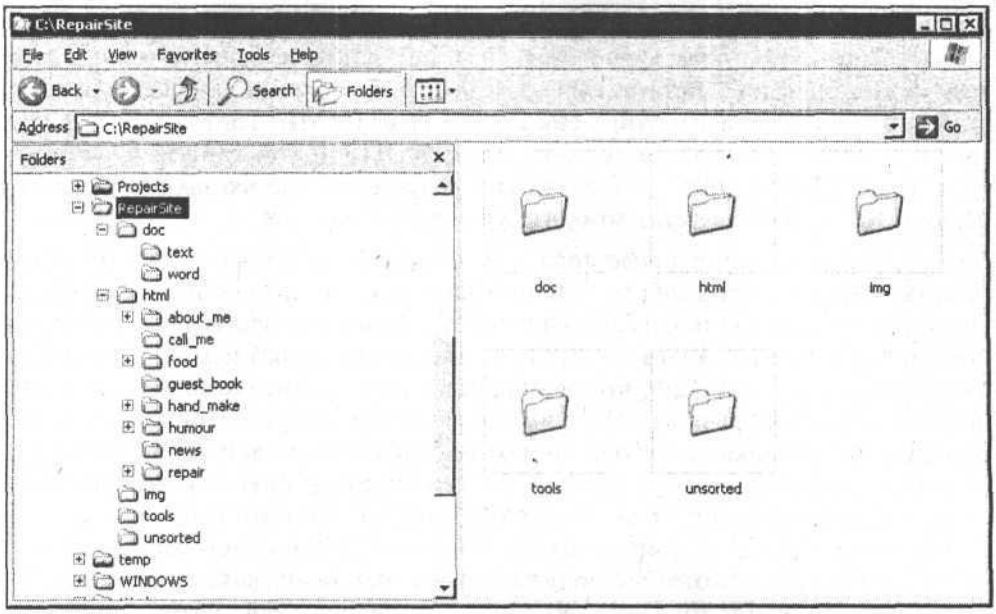


Рис. 3.8. Представление сайта в файловой системе компьютера.

После того как вы определитесь со структурой, необходимо аккуратно переместить всю имеющуюся информацию в соответствующие каталоги.

Теперь о возможных проблемах с вашим рабочим местом. Компьютер бывает ненадежен, а информация и ваши труды, как правило, бесценны, поэтому нужно побеспокоиться о том, чтобы их не потерять. Чтобы не пришлось потом, как говорится, кусать локти, я каждый день архивирую содержание всего своего рабочего каталога и перемещаю этот архив в какое-нибудь другое место на жестком диске. Архивный файл я называю так, чтобы можно было понять, когда он был создан (например, MyRepairSite-30-08-2004.zip). Чтобы "не замусоривать" каталоги устаревшей информацией, я удаляю с жесткого диска те архивы, которые были созданы более недели назад. Почему именно неделя? Потому что один раз в неделю я записываю свой последний недельный набор архивов на лазерный диск (CD-RW).

Какие программы нужны для создания простого сайта

Чтобы создать простой сайт, может понадобиться не так уж и много программ. Лично я пользуюсь определенным набором, который сформировался уже достаточно давно и практически не меняется.

Первая и самая главная программа — один из редакторов HTML-кода, который мы уже рассмотрели ранее, он необходим для создания самих web-

страниц. Я создаю код в Far-e, к HomeSite обращаюсь редко — только если надо сделать какую-нибудь сложную таблицу, над которой лень думать. Думаю, что новичкам не стоит пользоваться визуальными редакторами, поскольку без них лучше усвоится сам язык HTML. Когда система подсказывает теги и их атрибуты, в памяти ничего не остается. Стоит человеку, привыкшему к такому сервису, сесть за другой компьютер, где нет этого редактора, и он беспомощен, ничего не помнит. Недаром люди, привыкшие к калькулятору, чаще всего не могут в уме сложить трех- и даже двузначные числа.

Несмотря на то, что практически в любой из современных операционных систем существует встроенный браузер для просмотра HTML-страниц, вам понадобится еще несколько браузеров других производителей. Во-первых, существуют несколько (примерно четыре-пять) разновидностей браузеров различных производителей, которые чаще всего установлены на компьютерах пользователей Интернета. Стандартным для операционной системы Microsoft Windows (и потому привычным для многих пользователей) является браузер Internet Explorer (рис. 3.9).

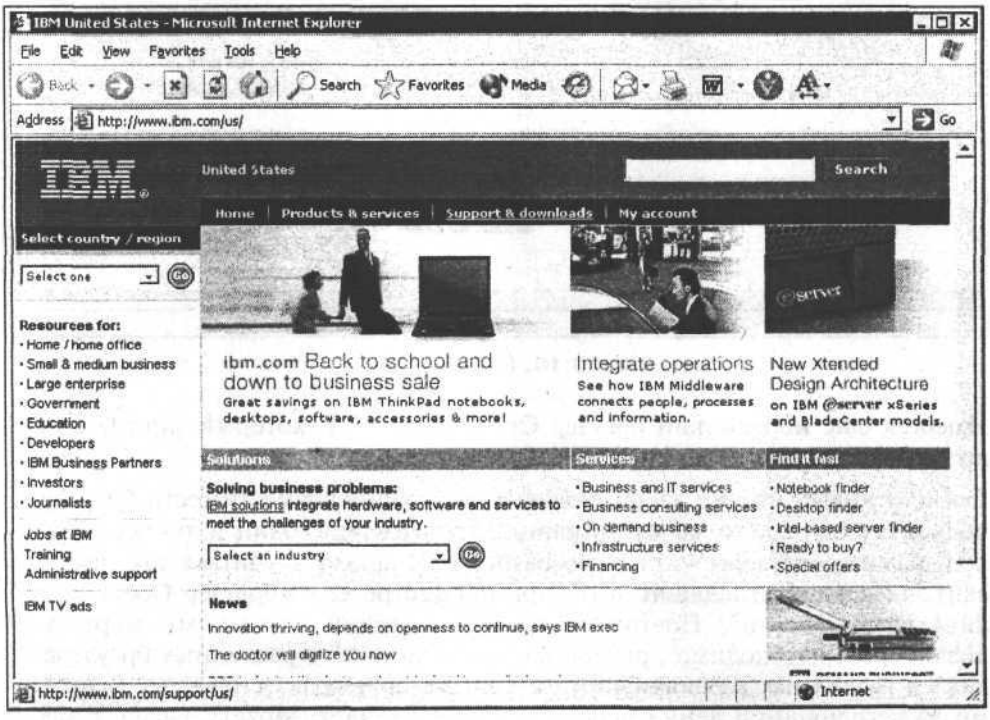


Рис. 3.9. Внешний вид браузера Internet Explorer

Имеется браузер с необычным названием Mozilla (рис. 3.10), с ним многие опытные пользователи работают вместо стандартного для Microsoft Windows

Internet Explorer (IE), поскольку в обеспечении безопасности и защиты от хакерских атак у IE постоянно обнаруживаются "дыры". А в Mozilla таких "дыр" нет.

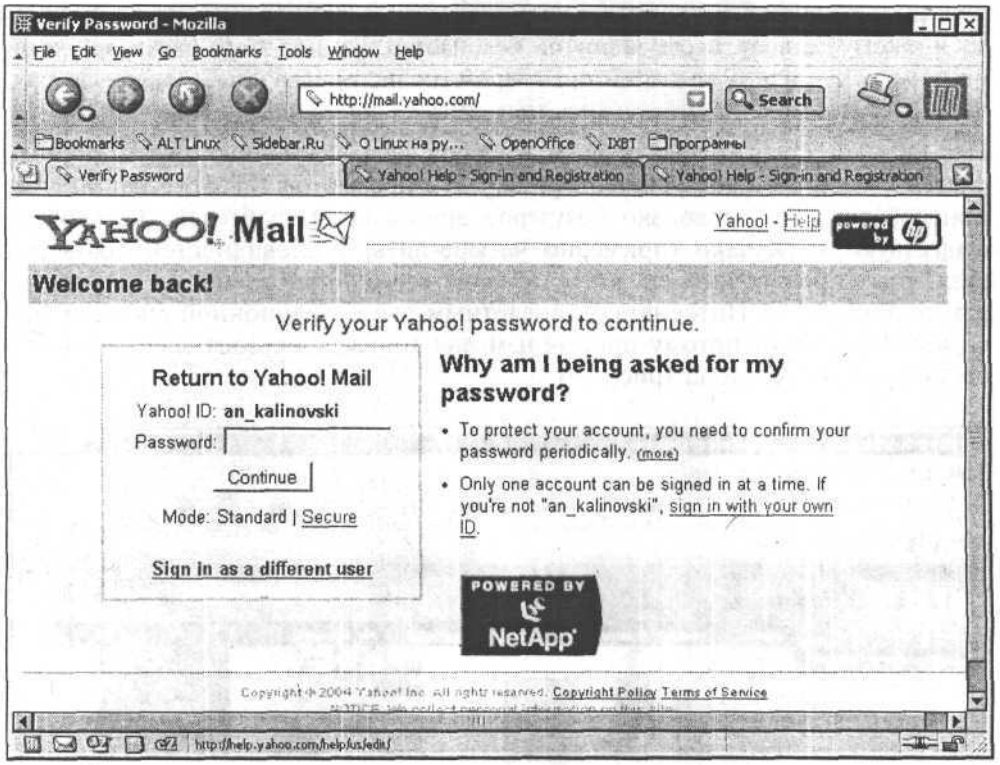


Рис. 3.10. Браузер Mozilla

Имеется еще норвежский браузер Opera (рис. 3.11), который многие считают более быстрым и удобным, чем остальные.

Так или иначе, но уже давно появилась проблема совместимости браузеров, поскольку одни и те же web-страницы, содержащие один и тот же HTML-код, разные браузеры читают по-разному. И может случиться так, что ваш сайт, идеально выглядящий в IE, при просмотре через браузер Opera "разваливается на части". Поэтому вам, как и всему остальному миру web-дизайнеров, необходимо проверять своего "хомяка" в различных браузерах и даже в различных версиях одного и того же браузера. Основными браузерами на сегодняшний день являются Internet Explorer, Mozilla, Opera, а также различные их версии, разновидности и надстройки. К примеру, некоторым из вас может быть больше известен браузер Netscape Navigator, но он вместе с Mozilla относится к так называемому семейству браузеров GECKO и различия в них минимальные и несущественные. Есть такой браузер, как

MyIE, который на самом деле и не браузер вовсе, а что-то вроде дополнения к системной части IE. Он выглядит по-другому, имеет свои удобные настройки и функции, но использует, как говорится, тот же "движок", что и IE, поэтому явных отличий в отображении web-страниц в этих двух браузерах также не будет. У меня на компьютере установлены Internet Explorer 6.0, Mozilla 1.7 и Opera 7.0. Для не очень сложных сайтов этого вполне достаточно.

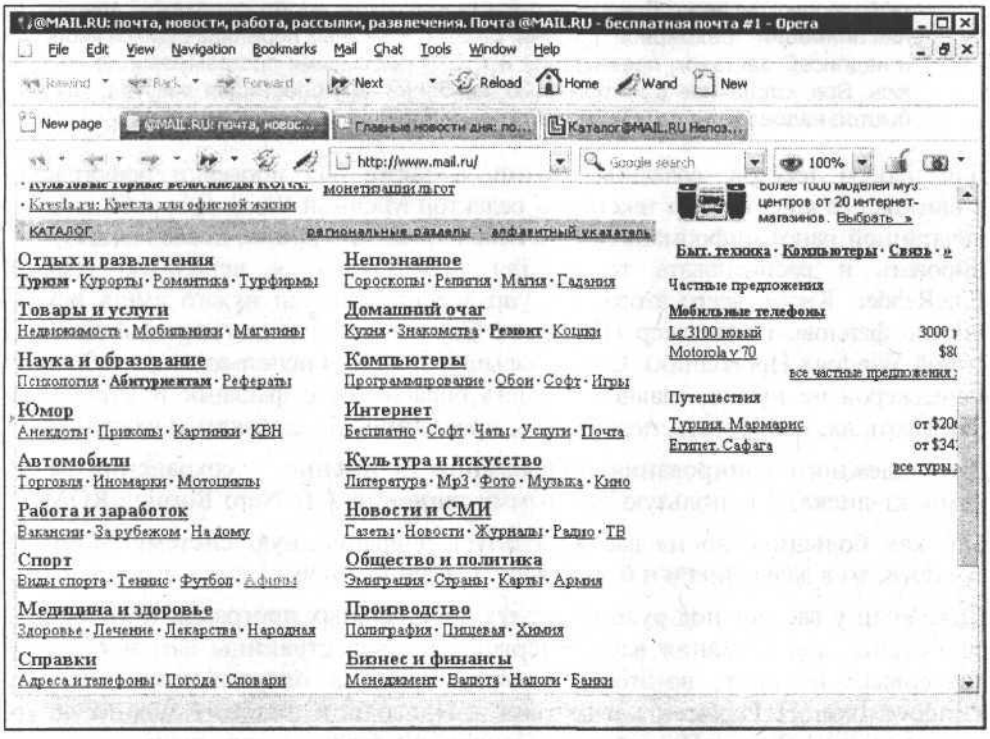


Рис. 3.11. Браузер Opera

Так как на сайтах бывает не только текст и гиперссылки, но еще и графика, то вам непременно понадобятся редакторы, в которых можно обрабатывать различные изображения и схемы. Для таких целей я использую два различных редактора — это Adobe Photoshop и CorelDRAW, предназначенные соответственно для обработки растровой и векторной графики. Растровые изображения состоят из набора точек (пикселов) различного цвета. Простым примером растрового изображения может служить любая отсканированная фотография. Векторное изображение — это изображение, основанное на некоторых координатах, линиях и геометрических фигурах. Именно как векторное, оно может существовать только в специальных редакторах типа CorelDRAW. Чтобы поместить такое изображение на сайт, придется преобразовать его

в растровое. Вот поэтому-то (как я и говорил ранее) все схемы и прочую векторную графику стоит держать также и в исходном, векторном формате.

Сканировать различные изображения и фотографии можно при помощи Adobe Photoshop.

Примечание

Что касается графики, то существуют еще программы, которые могут пригодиться любителям красивого блестящего и динамического — это редакторы для создания анимации, трехмерной графики, разных объемных переливающихся кнопочек и надписей, заставок, презентаций и т. д. Я подобными программами не увлекаюсь. Все, сделанное с их помощью, не более чем блестящая мишура, которая быстро надоедает и портит впечатление у большинства пользователей.

Для набора текстов, создания таблиц, а также для проверки орфографии и синтаксиса я использую текстовый редактор Microsoft Word. Для работы с напечатанной ранее информацией мне необходима программа, позволяющая сканировать и распознавать текст. Для таких целей я использую ABBYY FineReader. Кроме всего этого, для управления файлами нужно иметь какой-нибудь файловый менеджер (Far, Total Commander или стандартный для Microsoft Windows Проводник). Особо больших знаний в использовании файловых менеджеров не нужно, главное — уметь обращаться с файлами и каталогами (создавать их, копировать, перемещать, просматривать, запускать и удалять).

Для надежного копирования информации (а именно — сохранения ее на компакт-диске) я использую программу записи на CD Nero Burning ROM.

Так как большинство из вас использует операционную систему Microsoft Windows, то в дальнейшем я буду говорить именно о ней.

Даже если у вас нет под рукой всех тех специальных программ, которые перечислены, для создания вашей первой тестовой страницы вам необходим во-первых, Блокнот, во-вторых — встроенный в операционную систему Windows Internet Explorer, а в-третьих — Проводник, который можно запустить из меню **Пуск | Программы** (Проводник практически аналогичен элементу Мой компьютер).

Проверка связи. Тестовая web-страница

Что мы должны проверить в первую очередь, прежде чем приступать к созданию сайта, так это то, что web-страницы, которые будем создавать в HTML-редакторе, можно нормально открывать и просматривать браузером. Изначально (по умолчанию) в операционной системе все настроено так, что если вы двойным щелчком мыши запустите любой HTML-файл, то откроется браузер (опять же, по умолчанию Internet Explorer), который и покажет эту web-страницу, поскольку таким образом устроены ассоциации файлов (так называемые mime-типы). Это значит, что у операционной системы Windows

есть инструкция — в случае запуска файла с расширением html надо использовать для его просмотра браузер Internet Explorer. Даже если вы переименуете какой-нибудь графический файл в файл с расширением html, то система все равно откроет вам его в браузере не в виде картинки, а в виде бинарного файла, испещренного разными "краказябрами".

Итак, подготовим небольшую тестовую страничку. Для этого, если на вашем компьютере еще нет HTML-редактора, откроем Блокнот и напишем там текст, представленный в листинге 3.1.

Листинг 3.1

```
<html>
<head>
 <title>Тестовая страничка</title>
</head>
<body>
  Привет! Это тестовая страничка.
</body>
</html>
```

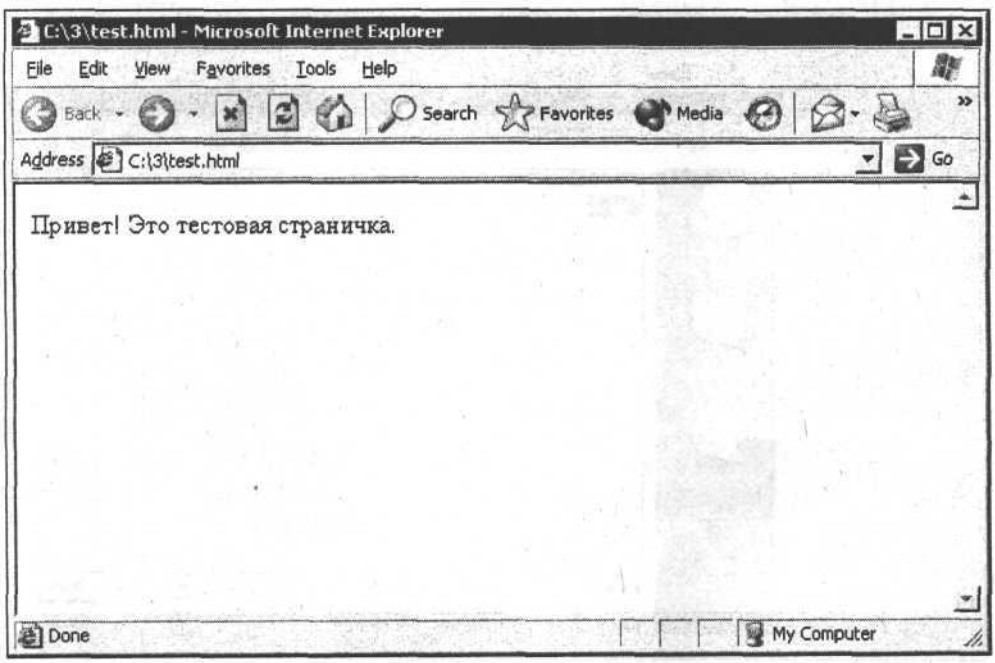


Рис. 3.12. Тестовая страничка

На самом деле это далеко не все, что должно быть написано даже на такой простой страничке, но для теста этого хватит. После того как вы наберете содержание листинга в Блокноте, необходимо вызвать пункт меню **Файл | Сохранить как**, затем в появившемся диалоговом окне ввести нужное имя файла для вашей странички, указав также место на жестком диске, где вы будете его сохранять. Потом необходимо найти этот файл при помощи какого-нибудь файлового менеджера (или в программе Мой компьютер), а затем запустить его двойным щелчком мыши или клавишей <Enter>. У вас откроется браузер, в котором появится эта тестовая страница (рис. 3.12).

Кроме того, существует еще один способ открытия файлов с жесткого диска — вам необходимо запустить сам браузер: **Start** (Пуск) | **Programs** (Программы) | **Internet Explorer**, а затем в главном меню выбрать **File** (Файл) | **Open** (Открыть), затем нажать на кнопку **Browse** (Обзор), после чего указать месторасположение вашей тестовой странички.

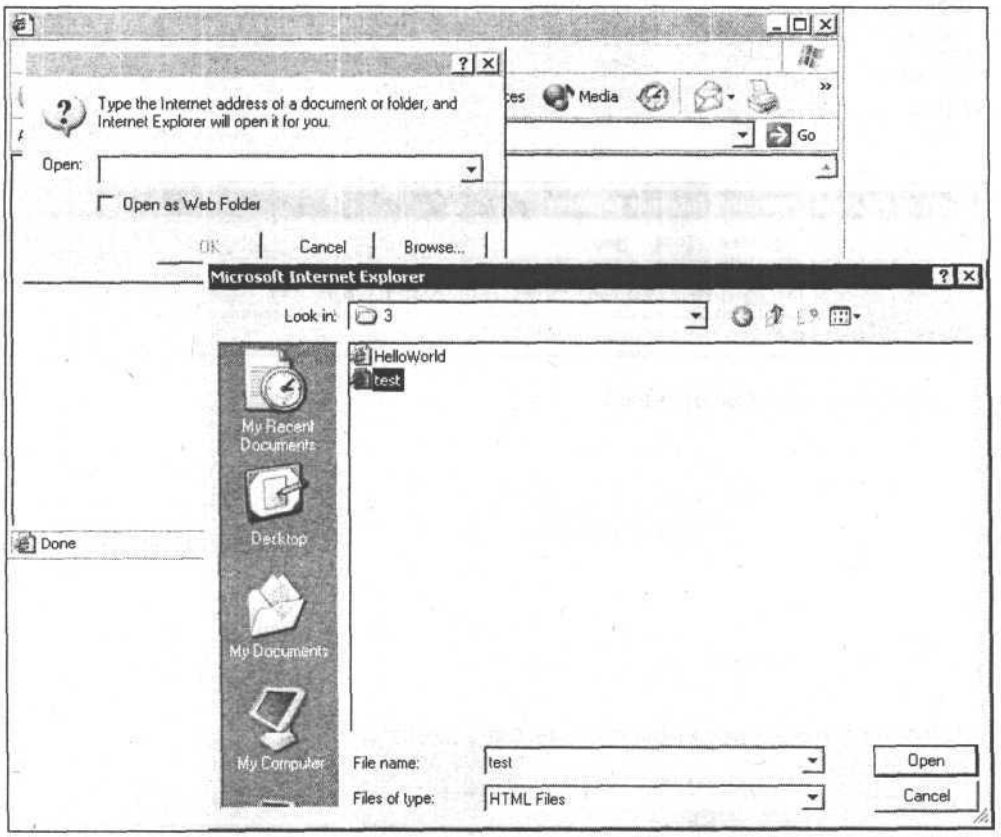


Рис. 3.13. Открытие файла из браузера Internet Explorer

Из чего состоит браузер

Теперь пришло время поговорить о том, из чего состоит браузер. Браузеры бывают разные, однако те их составные части, о которых я буду говорить, являются основными практически в любом из существующих браузеров, а кроме того, они очень схожи между собой. Поэтому для примера рассмотрю только Internet Explorer. На рис. 3.14 представлено окно браузера со всеми его основными элементами. Я специально ввел адрес сайта **Microsoft.com**, хотя знаю, что в данный момент компьютер не подключен к Интернету, — мне нужно, чтобы проявились все элементы браузера.

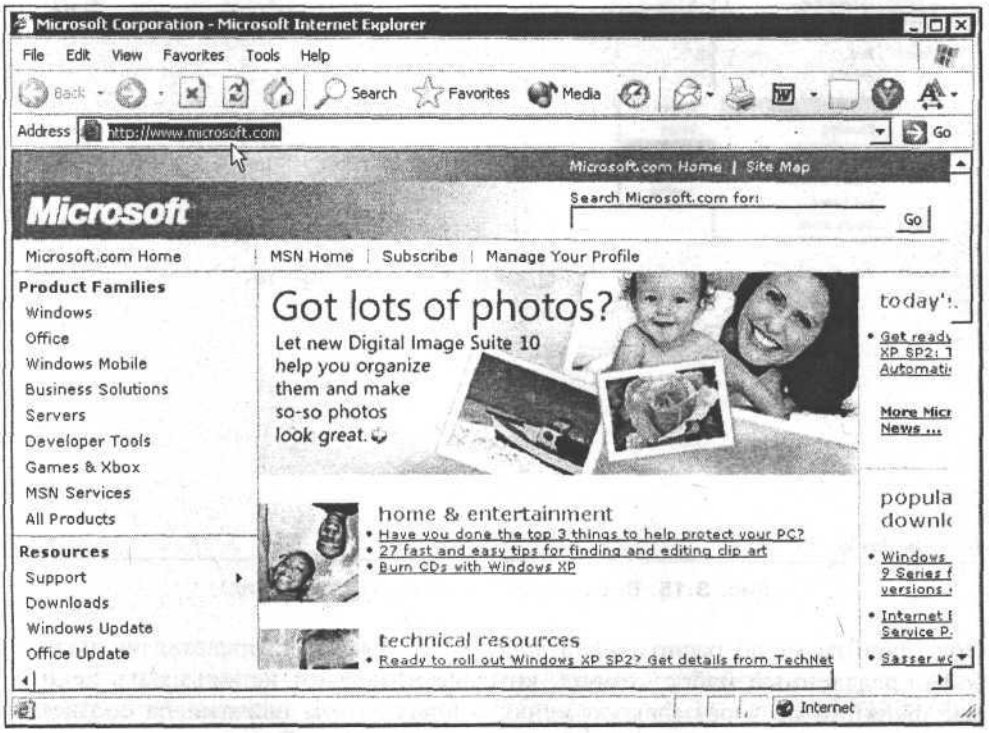


Рис. 3.14. Окно браузера

Итак, перед нами окно браузера, в котором отображена HTML-страница.

В верхней части на полосе с темным фоном расположен *заголовок окна* **Microsoft Corporation - Microsoft Internet Explorer**. Обычно заголовок должен кратко описывать то, что находится на данной странице. Сам текст заголовка показывается не только в окне, но и в Панели задач самой операционной системы, рядом с кнопкой **Start** (Пуск).

Чуть ниже расположено *главное меню*, состоящее из пунктов: **File** (Файл), **Edit** (Правка), **View** (Вид), **Favorites** (Избранное), **Tools** (Сервис), **Help** (Справка). В главном меню сосредоточены основные функции браузера, которые позволяют настраивать соединение с Интернетом, сохранять файлы, настраивать кодировку и величину шрифта, просматривать исходный HTML-код страницы и т. д. На рис. 3.15 представлено главное меню с раскрытым списком команд для пункта меню **View** (Вид).

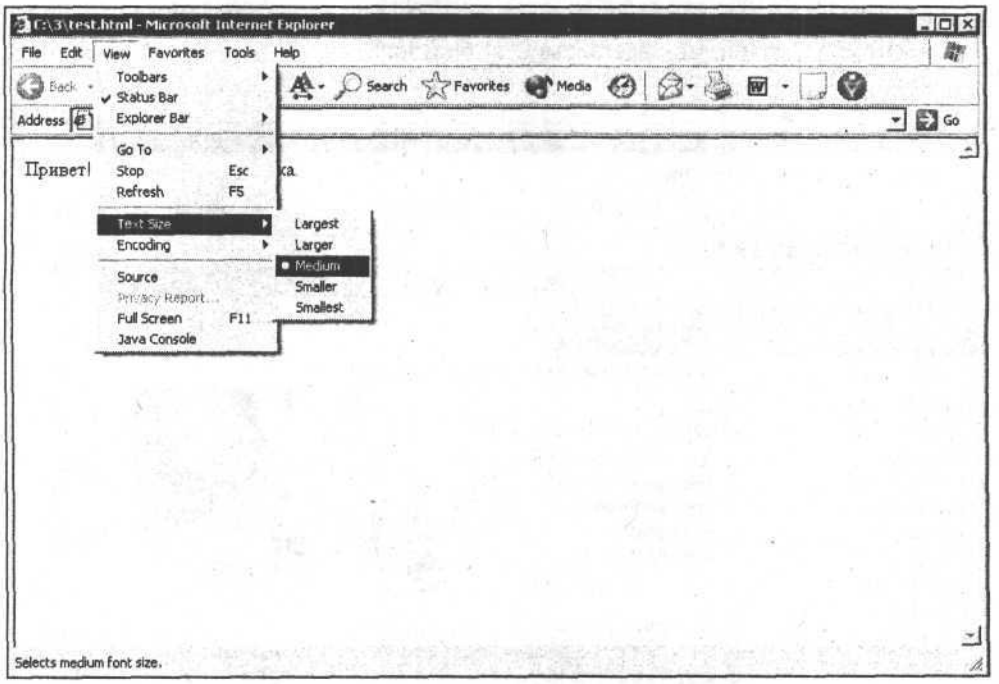


Рис. 3.15. Выпадающее меню пункта **View** (Вид)

Под главным меню расположена *панель инструментов*, представляющая собой определенный набор команд, которые позволяют использовать некоторые функции не через главное меню, а посредством нажатия на соответствующую кнопку с картинкой (пиктограммой). Панель инструментов предназначена для более оперативного вызова тех же самых команд, что имеются в пунктах главного меню. Рис. 3.16 иллюстрирует возможности кнопки панели инструментов, с помощью которой можно изменять размер шрифта.

Панель инструментов можно настраивать так, как удобно именно вам. Например, существует очень удобная и полезная кнопка, позволяющая менять размер шрифта web-страницы, но она по умолчанию не вынесена на панель инструментов. Чтобы сделать ее доступной, необходимо зайти в настройки

браузера (пункт главного меню **View** (Вид) | **Toolbars** (Панели инструментов) | **Customize** (Настройка)), в окне **Customize Toolbary** выбрать нужную, после чего нажать на кнопку **Add** (рис. 3.17).

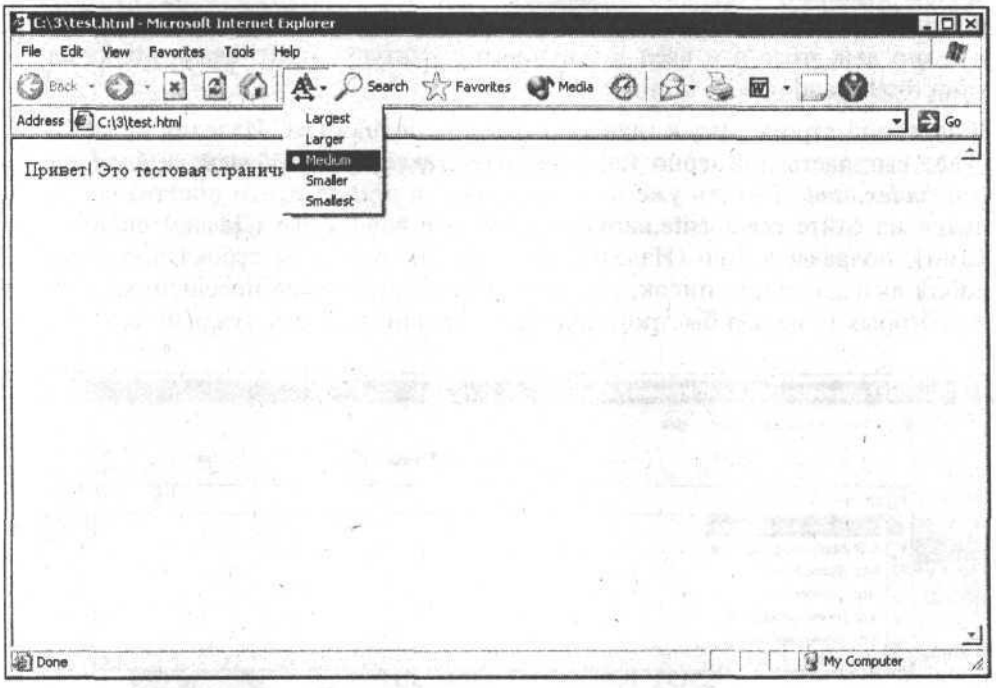


Рис. 3.16. Кнопка панели инструментов, посредством которой можно изменять размер шрифта

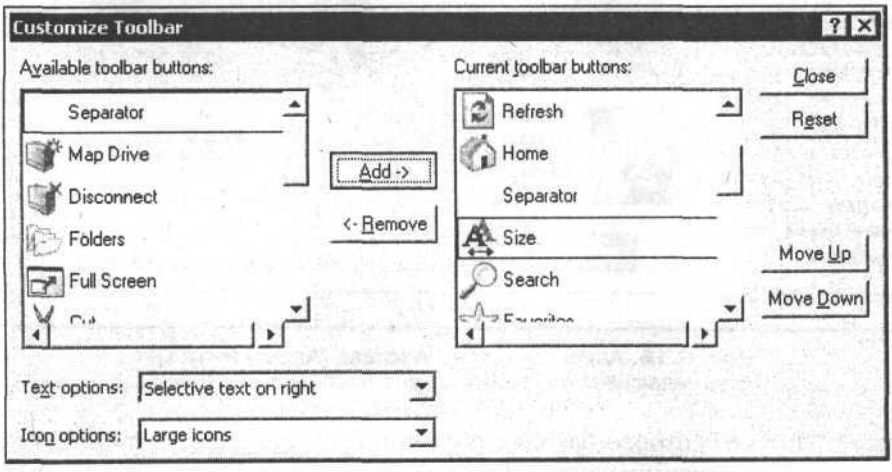


Рис. 3.17. Диалоговое окно, позволяющее настраивать панель инструментов браузера Internet Explorer

Под панелью инструментов имеется поле ввода, называемое *адресной строкой* (рис. 3.18). В ней вводится адрес сайта, так называемый *URL (Uniform Resource Locator – универсальный определитель ресурса).* При переходе на другую страницу этот URL меняется соответственным образом. Адрес часто помогает опытным пользователям Интернета ориентироваться на сайте, именно для этого я и ввел в файловую структуру своего сайта подкаталоги типа food, hand_made, iron и т. п.

В адресной строке путь к главной странице подраздела "Изделия из металла" будет выглядеть примерно так: ***http://www.repairsite.somedomain.ru/hand_make/iron/index.html.*** То есть уже из самого адреса понятно, что посетитель находится на сайте **remontsite.narod.ru** в разделе **hand_make** (Делаем своими руками), подразделе **iron** (Изделия из металла). Адресная строка представляет собой выпадающий список, где сохраняются последние посещенные адреса, из которых (с целью быстрого перехода) можно выбрать нужный вам.

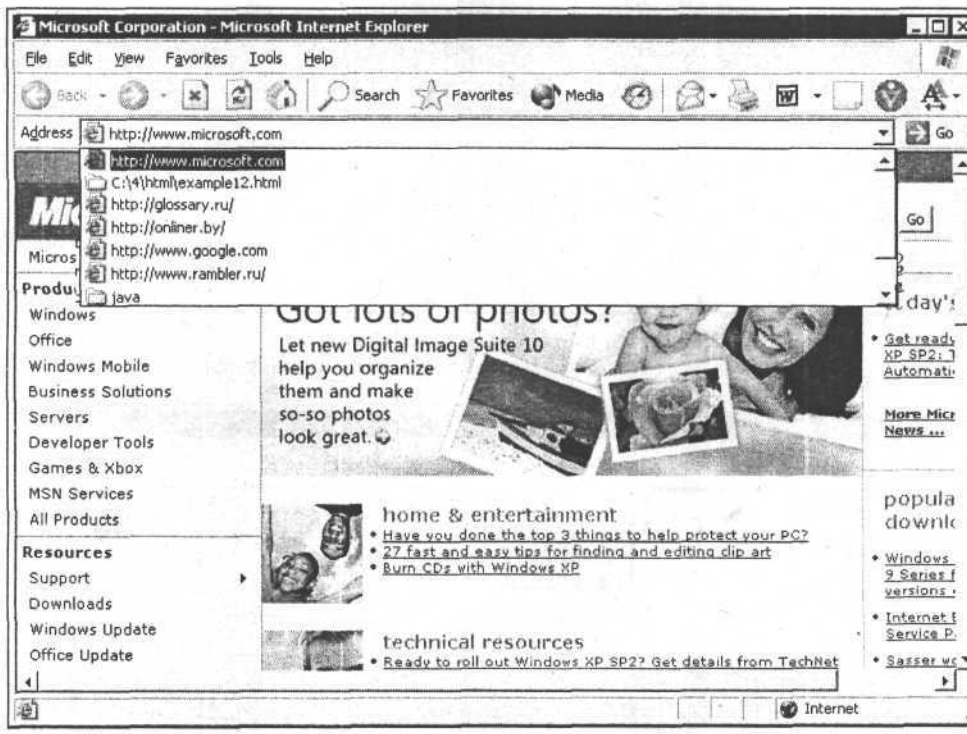


Рис. 3.18. Адресная строка **Address** (Адрес) браузера с выпадающим списком недавно посещенных страниц

То пространство, которое находится внутри окна браузера, и в котором отображаются web-страницы, называется *рабочая область окна.* Она и является основным контейнером для содержимого сайта. Если содержимое не поме-

щается в размер одного экрана, то появляется *полоса прокрутки* (небольшой ползунок справа от рабочей области окна, потянув за который мышью, можно просмотреть все содержимое страницы).

Наконец, в самой нижней части окна находится *строка состояния* (или *статусная строка*). Это небольшая область окна, где отражается процесс загрузки web-страниц, адреса, на которые указывают гиперссылки (когда наводишь на них курсор мыши). Там может быть и та информация, которую пожелаете показать вы.

Еще одним важным элементом окна браузера является *указатель мыши* (или просто *курсор*). Дело в том, что чаще всего в Интернете при наведении курсора (например, на гиперссылку) обычный курсор-стрелка превращается в руку с вытянутым указательным пальцем. Иногда, когда гиперссылки на сайте не очень заметны, именно такое изменение внешнего вида указателя мыши и позволяет понять, что это гиперссылка.

Из чего состоит пустая страница

Говоря о пустой странице, я имею в виду страницу, которая ничего не показывает в браузере. Сам HTML-код в ней, естественно, есть, просто он состоит из элементов, не предназначенных для отображения. Этих элементов несколько. Они описывают различного рода служебную информацию, особо не влияющую на то, как внешне выглядит страница. Вернемся снова к нашей HelloWorld!-странице. В листинге 3.2 представлен ее код безо всякого лишнего содержимого.

Листинг 3.2

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html>
<head>
 <meta http-equiv="Content-Type" content="text/html; charset=windows-
↳1251">
 <title>HelloWorld!</title>
</head>
<body>
</body>
</html>

Самая первая строка:
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
```

Это объявление типа документа, т. е. как бы указание системе того, что в данном документе будет использоваться язык разметки HTML версии 4.0. Эта строка — самая первая в коде HTML-страницы.

Сразу за ней следует открывающая часть корневого (самого верхнего уровня) тега — `<html>`, который в свою очередь содержит внутри два вложенных тега: `<head>` (заголовок документа) и `<body>` (тело документа).

Внутри заголовка находится описание различной служебной информации, например, так называемые мета-теги. Все мета-теги предназначены для описания различных сведений о документе, начиная с тех, которые необходимы для его регистрации, систематизации и индексирования поисковыми роботами, и заканчивая описанием авторских прав. В примере описан только один мета-тег, который указывает на то, что содержимое документа — это текст или язык разметки, а также кодировка, в которой необходимо отображать данный документ (windows–1251):

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

Если не указывать, какая именно должна использоваться кодировка, то браузер может не справиться с ее автоматическим определением и покажет вам сплошные закорючки (рис. 3.19) вместо нормального русского текста (естественно, для английского языка кодировка не имеет такого значения).

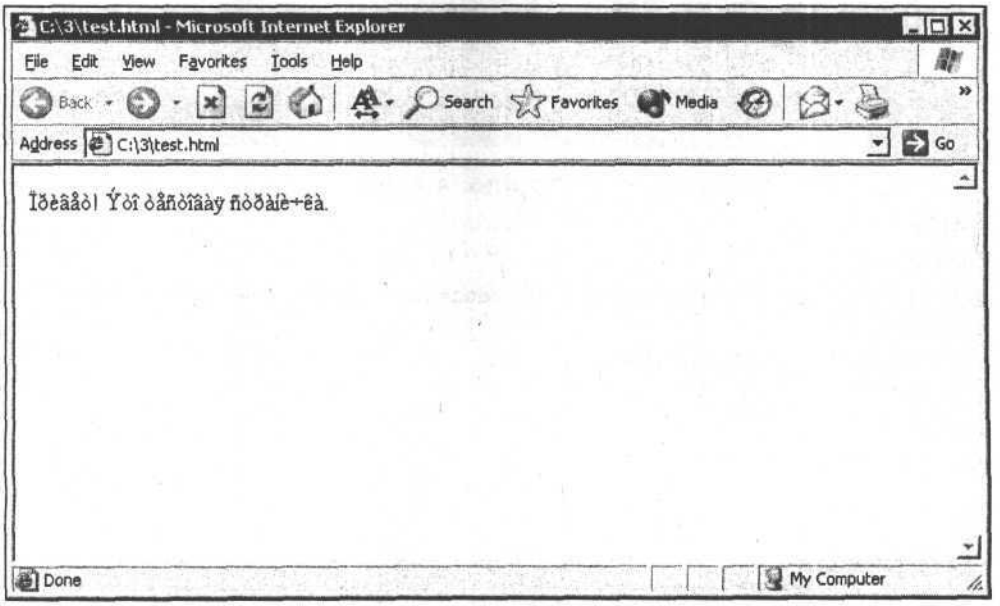


Рис. 3.19. Внешний вид текста в том случае, если кодировка не будет указана в заголовке документа

Кроме мета-тегов и других служебных тегов, которые будут рассмотрены позже, заголовок документа `<head>` должен содержать описание заголовка окна браузера. Оно описывается внутри тега `<title>`. В приведенной мною пустой странице это будет заголовок "Hello World!".

Тело документа — это тег `<body>`. Внутри него и должно находиться само содержимое web-страницы, которое будет отображаться браузером.

Примечание

Самое интересное, что можно даже полностью пропустить все такие правила и написать внутри файла просто кусок HTML-кода, например:

```
Привет! <font color="red">Это</font> тестовая страничка.
```

Браузер поймет эти инструкции и покажет web-страницу должным образом, но при определенных обстоятельствах проблемы все равно возникнут.

Итак, что мы имеем внутри пустой HTML-страницы? Она состоит из описания типа документа, корневого тега `<html>`, внутри которого расположен заголовок документа `<head>` с мета-тегами и прочей служебной информацией, а также тело документа `<body>`, включающее непосредственно само содержимое web-страницы.

Какими должны быть имена страниц, картинок и других файлов

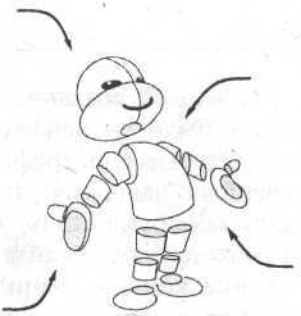
Многие пользователи операционной системы Microsoft Windows привыкли к тому, что не имеет значения, какие буквы используются в наименовании файлов — строчные или прописные. Если вы назовете файл TestPage.html, а обращаться к нему (например, устанавливая на него гиперссылку) будете как к файлу testpage.html, все равно операционная система найдет его и будет воспринимать записи TestPage.html и testpage.html, как одно и то же. Но это только в Microsoft Windows. В Интернете же многие сайты находятся на серверах под управлением операционных систем UNIX или LINUX, а для этих операционных систем различное написание всех символов имеет свое значение. Если вы, опять же, назовете свою страницу TestPage.html, а обратитесь к ней как к testPage.html, то получите результат: **Данная страница не найдена**. Это очень важный момент и его необходимо запомнить. Это касается любых файлов: и web-страниц, и картинок, и программ.

Желательно также не давать слишком длинные имена файлам (хоть практически все современные операционные системы это позволяют), поскольку будет увеличиваться вероятность ошибки при указании имени файла, к тому же такие имена трудно читать.

Имена всех файлов желательно давать буквами английского алфавита, т. к. проблемы с кодировкой могут возникнуть неожиданно не только с самим содержимым web-страниц, но и с именами файлов.

Необходимо также учитывать и то, что вы не будете постоянно работать со всем содержимым вашего сайта и, следовательно, со временем забудете, что значат те имена, над которыми вы поленились хорошенько подумать. Если, к примеру, вы назовете файл web-страницы Page-2.html, то очень скоро забудете, что в нем содержится. Но если вы назвали его setupDoors.html, то даже через пять лет поймете, что в нем идет речь об установке дверей. Кроме того, если кто-то другой должен будет разобраться в работе вашего сайта, то это гораздо легче сделать, когда названия файлов будут "говорящими". В этом случае понижается вероятность появления файлов с одинаковыми названиями. Конечно, если вы не знаете английского языка или вам очень не хочется постоянно задумываться и пользоваться словарем, то можно давать русские названия, но английскими буквами (например, ustanovka_dverey.html).

Чтобы в дальнейшем не путаться, вы должны выработать для себя определенные правила по формированию названий файлов и стараться следовать им всегда. Например, я никогда не называю каталоги с большой буквы, в этих именах только маленькие буквы. Если имя каталога состоит из двух или более слов, то я ставлю между этими словами знак подчеркивания, чтобы не было пробелов в именах файлов (хотя пробелы и допустимы). Все имена я стараюсь делать по возможности короткими и только на английском языке. Когда каких-то схожих между собой файлов много (например, фотографии, которые вы хотите поместить в своей виртуальной галерее), то, конечно же, лучше давать названия с помощью цифр (например 024.jpg, 025.jpg) вместо "говорящих" названий вроде we_are_in_forest_behind_fire.jpg, watching_at_the_lake_water.jpg и т. п. Такие файлы проще лишний раз просмотреть в специальном просмотрщике, чем помнить длинные сложные имена.



Глава 4

Разработка макета сайта и шаблонов страниц. Скелет "хомяка"

Как ускорить разработку web-сайта

Поскольку мы еще не очень хорошо разбираемся в web-дизайне и HTML, а нам предстоит достаточно серьезная работа по созданию web-страниц, поэтому необходимо, насколько возможно, облегчить и ускорить процесс разработки. Проблема ускорения разработки чего бы то ни было всегда волновала людей, особенно тех, кто связан с компьютерными системами. Для таких целей используются различные подходы, но чаще всего считается, что ускорить процесс позволяет хорошее управление самой разработкой, визуализация процесса программирования (т. е. когда код создается автоматически) и шаблонизация (когда система основана на заранее заготовленных компонентах и шаблонах). Этот подход может быть в некоторой степени применен и в отношении разработки простенького "хомяка".

Сначала об управлении. Необходимо постараться составить для себя план, по которому нужно создавать сайт. Пусть даже этот план будет у вас только в голове, уж лучше так, чем когда его нет вовсе, а вы тем временем мечетесь от одного к другому: то беретесь за дизайн, то подбираете шрифт, то набираете очередную статью. Или, что еще хуже, сидите, глядя на экран компьютера, и ничего не делаете, потому что не понимаете, за что взяться в данный момент. Кроме того, важно постараться сгруппировать свои задачи по созданию сайта таким образом, чтобы делать нечто схожее. Когда человек занимается схожими делами, он меньше отвлекается, работает более продуктивно и приносит больше пользы. Если я, например, берусь обрабатывать графику, которую планирую поместить на сайте, то занимаюсь только графикой и не перехожу к другим проблемам, пока не закончу с ней.

Есть еще одна вещь, о которой помнить все время очень сложно и о которой я часто забываю, — нельзя отвлекаться на мелочи. Иногда, бывает,

занимаешься созданием макета главной страницы сайта, а потом подумаешь — было бы неплохо сделать тут красивую иконку или придумать логотип. Открываешь графический редактор, ищешь что-нибудь похожее в собственных "закромах", подбираешь цвета или размер иконки, а к двум часам ночи обнаруживаешь, что вместо того чтобы сделать большую половину макета своего сайта, ты провозился с какими-то картинками, которые, в конце концов, решил вообще не использовать. Так поступать не стоит. Главное — это самые основные вещи на сайте — страницы, текст на них, гиперссылки. Даже цвет и картинки не столь важны. Если делать качественный сайт, то прежде всего надо заботиться о самом главном, все остальное можно усовершенствовать позже. Всяческие украшения далеко не самое важное. В мире существуют тысячи web-дизайнеров, за которыми нам все равно не угнаться, зато у них, может быть, нет такой ценной информации.

Однако необходимо рассмотреть еще не менее важный момент разработки сайта — это *создание каркаса*. Именно о нем мы поговорим в этой главе. Для нас это будет уже третий смысловой уровень сайта.

Первый уровень — это уровень идеи (идея самого сайта, для чего он нужен, кому предназначен, и вообще — в чем его суть).

Второй уровень — структура сайта (т. е. информация, разделенная на разделы по смыслу). Оба эти уровня мы уже рассмотрели ранее.

Теперь поговорим о каркасе (или скелете), что и составляет третий уровень. Суть каркаса в том, что он описывает, как должна быть расположена информация на сайте, из каких смысловых блоков она может состоять. Вспомните, как сделана обычная газета. На странице есть, например, заголовок, который находится сверху; в правой части коротенькие новости; внизу — координаты редакции газеты; посередине — главная статья. Таким же образом формируется каркас web-сайта. Здесь у нас будет логотип, здесь — главное меню, здесь счетчик посетителей и т. д.

В зависимости от специфики вашего сайта этот каркас будет отличаться от других, как отличается скелет тигра от скелета акулы (они ведь живут в разных условиях, по-разному охотятся и передвигаются).

Обычно перед тем как приступить к написанию кода страницы и созданию каркаса для нее, я делаю ее бумажный прототип. Это очень просто и удобно. Я беру обычный лист бумаги и начинаю карандашом рисовать на нем будущий внешний вид сайта. Мне совсем не обязательно садиться за компьютер, хорошо знать HTML, рисовать красиво и ровно. Я могу перепробовать десяток идей. После того как нарисован каркас сайта на бумаге, я должен буду создать его и на компьютере, только уже с гораздо меньшими затратами времени.

Одним из главнейших условий быстрой разработки сайта является использование шаблонов и заготовок. Не пожалейте времени на тщательное создание образца — того файла-шаблона, на основе которого можно будет гото-

вить ваши разнообразные странички. В последующей работе вам останется только вставлять новый текст, не заботясь о дизайне.

Примечание

В принципе, к шаблонам можно относить не только заготовки каркаса страницы, это могут быть и подготовленные заранее компоненты web-страниц (например, таблицы для фотогалереи, выпадающее меню на несколько пунктов, которое потом можно будет увеличить, всевозможные заготовки с примерами цветов и шрифтов различных размеров).

Зачем нужны шаблоны страниц

Ну что ж, думаю, вы хотели бы видеть конкретные примеры, а не общие фразы о том, зачем нужны эти самые шаблоны. Я в своих первых страничках не пользовался шаблонами. Максимум, что делал — это небольшими фрагментами копировал код из уже созданных страниц и вставлял его в новые. Иногда я делал более удобные вещи, близкие к использованию шаблонов, — брал предыдущую страницу, переименовывал ее в ту, которую собирался сделать, убирал все лишнее содержание и вписывал нужное. Но, удаляя, например, ненужную таблицу, я захватывал и кусок таблицы каркаса, и вот тогда приходилось много возиться, иногда начиная все сначала, т. к. страница рассыпалась как карточный домик. Позже я понял, что поступаю неправильно. У меня должен быть пустой каркас, в который буду вставлять новое содержимое всякий раз, когда это понадобится.

При использовании шаблонной страницы я могу более продуктивно работать, не отвлекаясь на отладку HTML-кода для каждой страницы. Я добьюсь того, чтобы все правильно работало в шаблоне, а потом буду использовать уже работающий код.

Следующий аспект более важен для внешнего восприятия вашего сайта, чем для его создания. Единый стиль — вот что с точки зрения качества изготовления сайта и удобства его использования весьма важно. Все до единой ваши странички, относящиеся к одному сайту, должны выглядеть одинаково, а отличаться они могут только содержимым. Если фон серый, то должен быть серым везде. Если используется шрифт определенного размера, то информация такого же уровня и важности должна быть везде представлена таким шрифтом. Если ваши гиперссылки синие, они должны быть синими везде. Я хочу обратить на этот момент особое внимание, поскольку во Всемирной паутине это является правилом хорошего тона. Нельзя делать одну страницу с зеленым текстом на желтом поле, вторую — с зеленым текстом на черном фоне, а третью — белым по черному.

Во время работы над сайтом вам, безусловно, постоянно придется что-то подправлять. Поскольку та часть, которая относится к шаблону страницы,

везде будет выглядеть одинаково, то ее легче отыскать глазами в коде (ведь благодаря использованию шаблонов, у вас непроизвольно получится единый стиль не только во внешнем виде, но и в коде страниц).

Делаем качественный шаблон

Что же нужно предусмотреть при создании шаблона для того, чтобы он оказался максимально полезным и удобным? Во-первых, код в нем надо форматировать таким образом, чтобы его легче было читать и понимать.

Форматирование кода — это его внешнее оформление, особенно переносы строк и отступы. Хорошо отформатированный код в несколько раз повышает свою читаемость. Пример кода безо всякого систематизированного форматирования (как говорится, как получится) приведен в листинге 4.1.

Листинг 4.1

```
      <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transi-
tional//EN"><html>
      <head>
        <meta http-equiv="Content-Type" content="text/html; char-
set=windows-1251">
<meta name="ROBOTS" content="ALL">
<meta http-equiv="description" content="Персональный сайт о ремонте">
<meta name="author" content="&amp;copy; Пупкин Вася">
<meta name="security" content="Public">
<title>
      HelloWorld!</title>
<link type="text/css" rel="stylesheet" href="css/all.css">
</head>

<body> <table width="100%" height="100%">  <tr>
          <td height="1%" style="border:1px solid #bbbbbb">
            <h1 align="center">HelloWorld!</h1></td>
      </tr><tr>
          <td align="center" height="20">Copyright (C) nynkuH
      2004
           </td></tr></table>
</body>
</html>
```

Согласитесь, читать такое просто невозможно — совершенно непонятно, где начинается и где кончается какой-либо из тегов, неясно, где содержание,

а где служебная информация. Разобраться в такой мешанине, конечно, возможно, но с большим трудом.

Первое, что я делаю, когда хочу понять, как сделана чья-то чужая страница, — это форматирую ее код так, как привык делать я. Пусть такая работа занимает определенное время, но по этому поводу еще в известном мультфильме было сказано страусу: "Запомни, лучше день потерять, потом за пять минут долететь". Кроме того, большинство визуальных редакторов (если вы не пишите в них код, а моделируете свои страницы при помощи форм и мастеров) грешат именно этим — они форматируют код жутко, его невозможно читать, он подобен тому, что находится в листинге 4.1. А теперь посмотрите на пример того же кода, но только отформатированного (листинг 4.2).

Листинг 4.2

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"><html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
  <meta name="ROBOTS" content="ALL">
  <meta http-equiv="description" content="Персональный сайт о ремонте">
  <meta name="author" content="&amp;copy; Пупкин Вася">
  <meta name="security" content="Public">

<title>HelloWorld!</title>
<link type="text/css" rel="stylesheet" href="css/all.css">
</head>

<body>
 <table width="100%" height="100%">
  <tr>
    <td height="1%" style="border:1px solid #bbbbbb">
       <h1 align="center">HelloWorld!</h1>
    </td>
  </tr>
  <tr>
    <td align="center" height="20">
     Copyright (C) nynkuH 2004
    </td>
  </tr>
 </table>
</body>
</html>
```

Правда, есть разница? И хоть вы пока еще не знаете многих тегов, но тем не менее вам должно быть понятно, в чем общая суть форматирования кода.

Еще на заре появления различных языков программирования была придумана очень полезная вещь — *комментарий*. Это специальным образом оформленная пометка внутри программного кода, которая содержит краткое пояснение той части кода, где эта пометка находится. Например, если в программе начинается какая-нибудь функция или процедура, то комментарий расскажет тому, кто читает код, что делается в этой функции/процедуре. Когда речь идет о HTML-коде, то комментарии часто описывают структуру страницы. Это связано с тем фактом, что HTML-страницы бывают внутри весьма сложными (несколько таблиц могут быть вложены одна в другую, длинные тексты, находящиеся на web-странице в разных местах, в коде могут располагаться совсем рядом и т. д.). В таком случае при помощи комментариев можно описать, где начинается и где заканчивается какая-то часть страницы (например, форма опроса или разделы главного меню). Когда такое описание имеется, гораздо легче ориентироваться в коде.

Комментарии бывают двух видов — однострочные и многострочные.

Они начинаются сочетанием символов: `<!--` (открывающаяся угловая скобка, восклицательный знак, два тире).

Завершаются комментарии сочетанием символов: `-->` (двойное тире, закрывающаяся угловая скобка).

Браузеры не показывают комментарии на странице, т. к. они относятся к такой же служебной информации, как и мета-теги. Все, что попадает между открывающим и закрывающим знаками комментария, невидимо для глаза. Таким приемом иногда пользуются, когда необходимо какой-либо фрагмент кода скрыть от браузера, но не удалять его совсем. В листинге 4.3 приведен пример использования комментариев.

Листинг 4.3

```
<!--  Текст комментария в одну строчку  -->
<!--  Текст комментария
написанного в несколько
строчек
-->
```

Обычно в тех редакторах, где HTML-код подсвечивается, комментарии обозначены более бледным цветом, чем основное содержимое, поскольку для самой страницы они не важны.

Далее для каждой страницы мы собираемся использовать заранее заготовленный шаблон, который будет влиять на все параметры этих страниц.

На своем компьютере вы почти не заметите разницы в загрузке в браузер простых, легких страниц и страниц со сложной структурой, с большим количеством кода внутри. Однако когда ваш сайт будет находиться в Интернете, значение будет иметь каждая лишняя буква или цифра в коде. Как вам, наверное, известно, любой файл имеет свой размер, который измеряется в байтах (килобайтах, мегабайтах и т. п.). Чем больше этот размер, тем дольше производятся основные операции над этим файлом. Поскольку при работе в Интернете HTML-страницы загружаются с сервера на клиентский компьютер, то чем больше их размер, тем длительнее время загрузки. При существующей у многих пользователей средней скорости модема чересчур объемные страницы могут загружаться на компьютер пользователя от одной до нескольких минут. А поскольку код web-страницы — это текстовая информация, то каждая лишняя строчка замедляет загрузку страницы, каждый лишний килобайт отдаляет вашего посетителя от вас. Поэтому нужно постараться сделать ваши страницы как можно меньшего размера. Этому как раз и может помочь качественно сделанный каркас сайта и оптимально написанный шаблон.

Теперь надо сказать еще об одной важной вещи. Шаблон можно испортить, если вы случайно начнете его редактировать как новую web-страницу, а сохранить ее под новым именем забудете. Со мной такое случалось. Я вставлял информацию в пустой шаблон, долго работал с ней, подправлял, менял, а позже обнаруживал, что мой шаблон исчез, потому что я его попросту переименовал. Чтобы подобных проблем не возникало, полезно сохранять шаблоны страниц еще в каком-нибудь запасном каталоге, но при этом не забывать их обновлять (если вы все-таки откорректировали их в рабочем каталоге).

Жесткий или резиновый дизайн

В фильме "Назад в будущее" (по-моему, во второй части) был эпизод, в котором главный герой, попав в далекое будущее, обнаружил, что та одежда, в которую ему предложили облачиться, чтобы выглядеть современно, сама меняет свой размер, подстраиваясь под конкретного человека. Таким образом при производстве подобной одежды не уделялось внимание выпуску большого количества разнообразных размеров. А зачем? Она и так подходит каждому. К чему я это рассказываю?

Существуют две разновидности дизайна web-страниц с точки зрения размеров. Размер можно задавать в так называемых абсолютных единицах (например, к таким единицам относятся пиксели (px), точки (pt), миллиметры (mm) и т. п.). Такой дизайн называют жестким, потому что содержимое страницы не меняет своего размера и положения, когда изменяются размеры окна браузера. Положите кусок льда в два разных бокала, и он останется куском льда той же формы и размера, но если вы нальете воду в разные бокалы, то она займет по ширине все предоставленное пространство. Также

себя ведет содержимое страницы при резиновом дизайне. Резиновым называют такой дизайн, когда размеры основных элементов на странице указывают в относительных величинах, т. е. в процентах от возможной ширины. Например, если вы сделаете таблицу и укажите, что ее ширина равна 50 %, то она всегда будет растягиваться ровно на половину предоставленного ей места.

В чем же основные различия жесткого и резинового дизайна?

Во-первых, жесткий дизайн всегда делается под какое-то конкретное разрешение экрана (т. е. количество точек экрана по вертикали и горизонтали). Это количество устанавливается в настройках рабочего стола в соответствии с предпочтениями пользователя. Обычно, если монитор небольшой (например, 14 дюймов), то ставят разрешение 800 × 600 точек. Если монитор покрупнее (например, 17 дюймов), то устанавливается разрешение 1024 × 768 точек. Раньше наиболее часто встречающимся разрешением было 800 × 600. Сейчас все больше склоняются к разрешению 1024 × 768 точек. В случае жесткого дизайна вы должны выбрать какое-то конкретное разрешение и строить свои странички на его основе. При этом пользователи с другим разрешением либо не будут видеть части страничек, либо увидят пустое место. Пример жесткого дизайна для конкретного разрешения приведен на рис. 4.1.

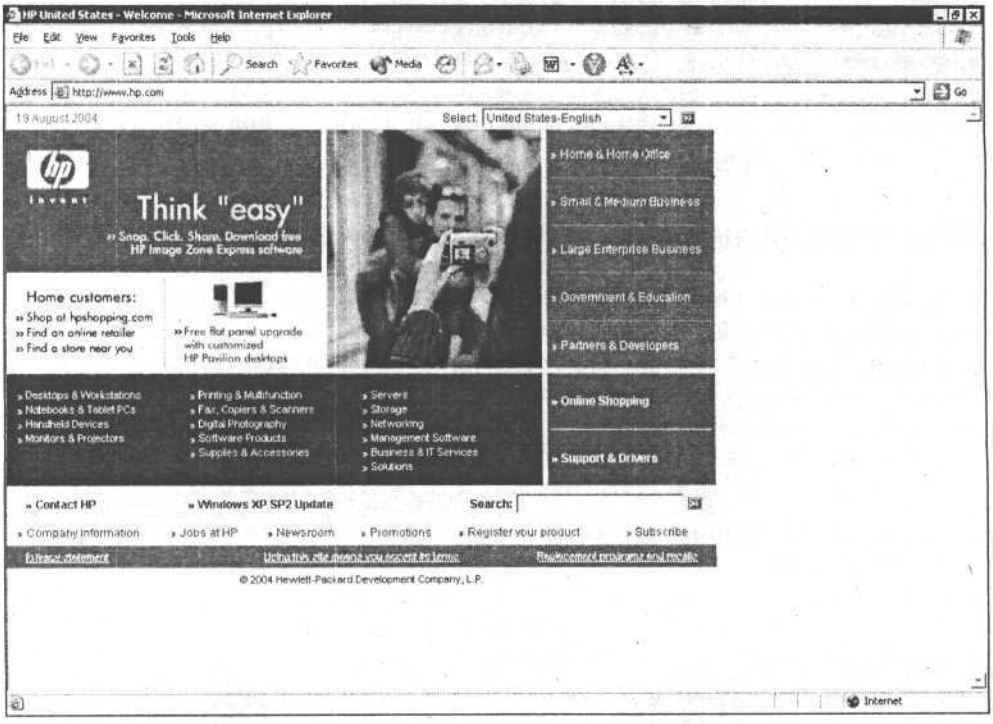


Рис. 4.1. Жесткий дизайн для разрешения 800 × 600

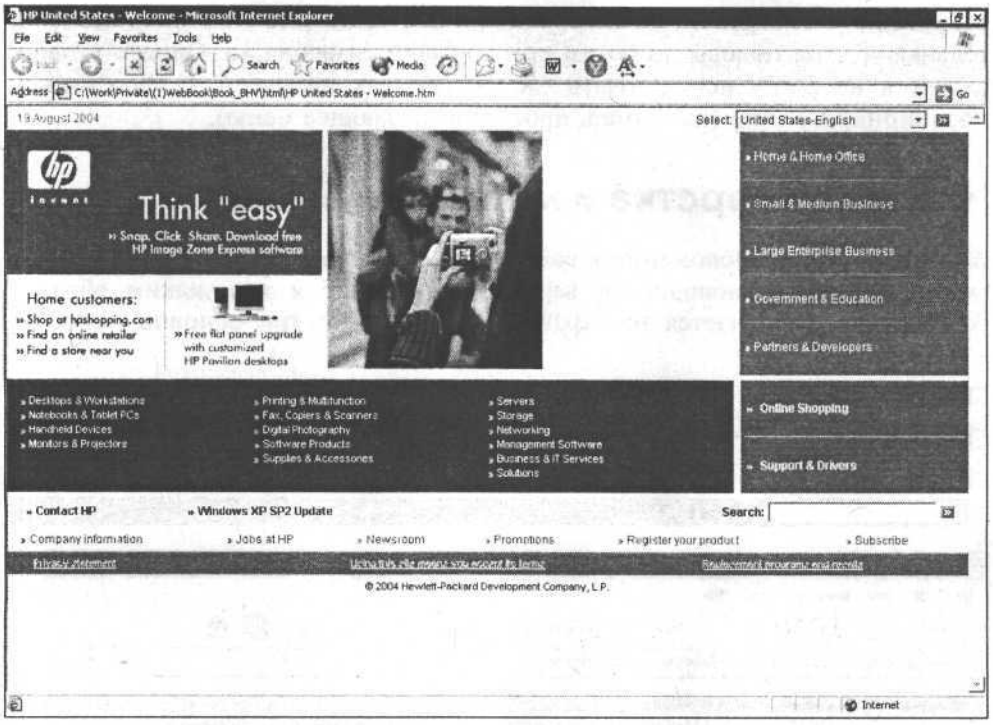


Рис. 4.2. Искусственно сделанная резиновая страница

Во-вторых, если вы хотите создать на страничках какую-то четкую, нерушимую композицию, то вам не обойтись без жесткого дизайна, иначе ваши труды "расползутся" по всему экрану. На рис. 4.2 приведен пример такой искусственно сделанной резиновой страницы, изначально созданной при помощи жесткого дизайна. Композиция разрушилась при большом разрешении.

В-третьих, жесткий дизайн делать проще, чем резиновый, страницы одинаково выглядят при всех разрешениях монитора, не нужно постоянно переключаться и проверять, не "уползает ли" содержимое страниц за допустимые пределы, не появились ли переносы строк там, где это не нужно, красиво ли отформатирован текст.

В четвертых, размеры некоторых элементов, задаваемые в процентах, могут некорректно показываться различными браузерами. И если вы задали какому-то элементу размер 80 %, а браузер считает, что у такого элемента не может быть относительного размера, то он все расположит на свое усмотрение.

Лично я предпочитаю резиновый дизайн. Мне не нравится, когда на сайте возникают пустые места или появляется горизонтальная полоса прокрутки. Мне не нравится, когда из-за этих пустых мест необходимо выравнивать всю страницу влево, а тем более — вправо. Более красивым мне кажется вы-

равнивание содержимого по центру страницы, когда остаются лишь незначительные пустые полосы по бокам при большем разрешении. Однако в таком случае я не смогу использовать элементы с так называемым абсолютным позиционированием (например, простое выпадающее меню).

Что такое верстка и какая она бывает

Верстка — это расположение в рабочей области окна браузера содержимого web-страницы. Разновидности верстки определяются тем, каким образом содержимое размещается на странице. Существуют три основных способа верстки:

- текстовая;
- табличная;
- фреймовая.

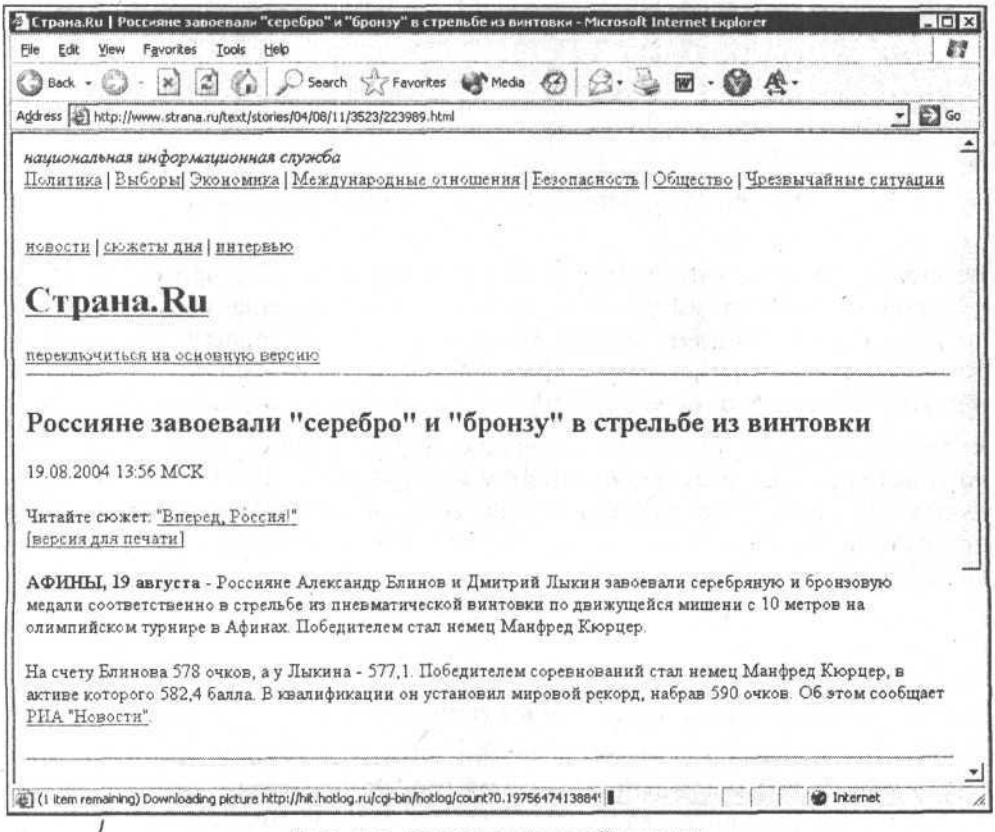


Рис. 4.3. Пример текстовой верстки

Текстовая верстка — это такой способ расположения содержимого на странице, когда кроме текста, картинок, гиперссылок и тому подобного никакие специальные элементы не используются. Например, текстовая верстка используется для книги, которую вы читаете. В ней есть и заголовки, и иллюстрации, и примечания, и обычный текст, но все эти элементы расположены сами по себе, без каких бы то ни было таблиц или рамок. Такая верстка очень проста и позволяет делать лишь примитивные по своей структуре сайты. Она не требует особых знаний и умения, но не слишком подходит самой идее web-сайтов и Интернета. Иногда текстовую верстку используют в том случае, когда необходимо предложить посетителю чисто текстовую версию сайта (рис. 4.3), скорость загрузки которой значительно выше скорости загрузки сайтов со сложными компонентами типа таблиц.

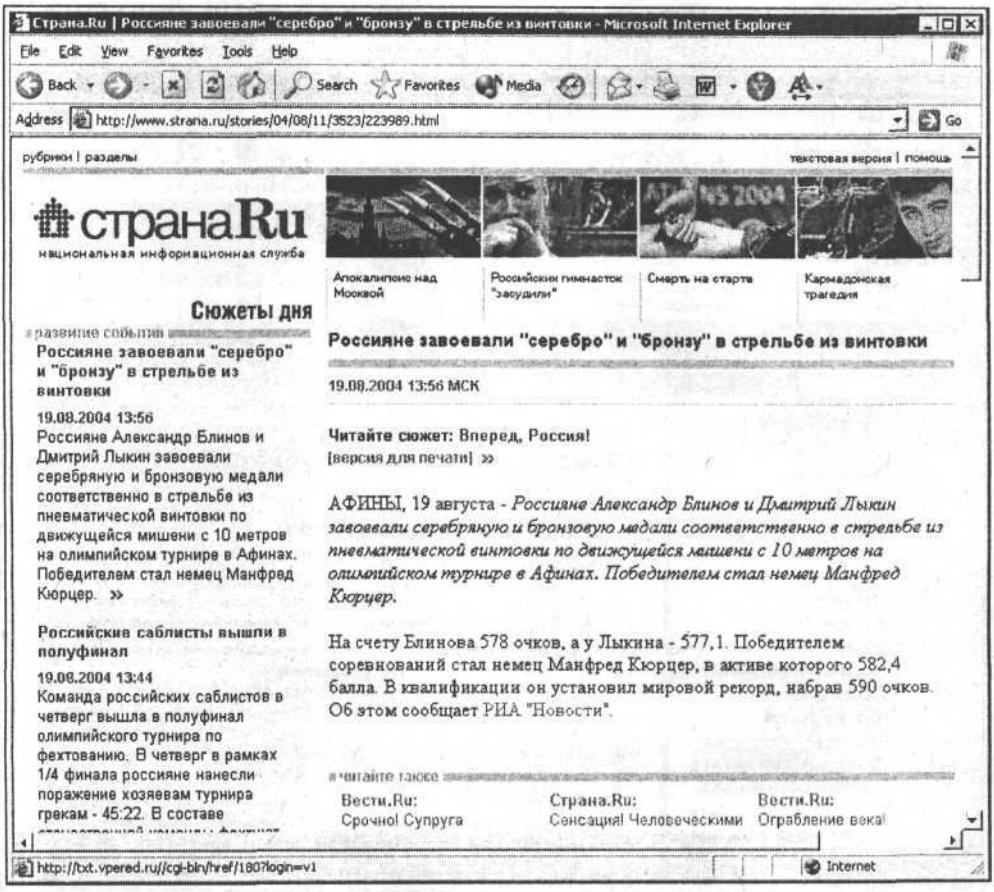


Рис. 4.4. Пример табличной верстки

Вторым способом является *табличная верстка*. Основой для содержимого страницы при такой верстке является специальная таблица, внутри ячеек которой содержатся все остальные элементы страницы. При помощи такой верстки можно размещать части содержимого там, где нужно, и управлять внешней структурой страницы при помощи разного расположения ячеек таблицы и их размеров. Пример такой верстки приведен на рис. 4.4.

Третий вид — *фреймовая верстка*. Это верстка при помощи фреймов (рамок). Смысл такой верстки в том, что страница разбивается на несколько частей, в каждой из которых в свою очередь показывается своя отдельная страница. Страница, на которой описано расположение рамок и их количество, не меняется, а меняются лишь те страницы, которые загружаются в конкретную рамку. Это вроде фанеры, на которой нарисована какая-то картинка и прорезана дырка для лица. Разные люди просовывают лица в эту дырку, а фотограф их "щелкает". Содержимое фотографии меняется, сама же фанера с рисунком остается неизменной. Пример фреймовой верстки приведен на рис. 4.5.

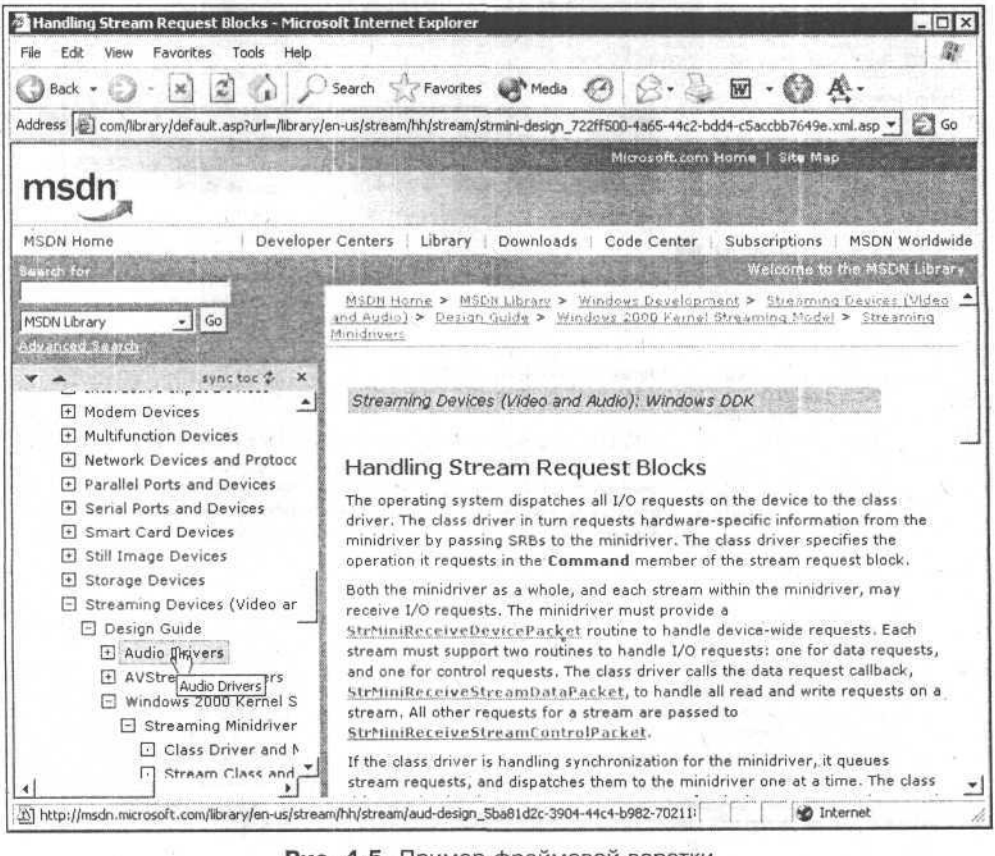


Рис. 4.5. Пример фреймовой верстки

Как текстовая, так и табличная верстка используются для каждой отдельной страницы, верстка же при помощи фреймов — для нескольких страниц сразу. Например, страница внешне может выглядеть, как состоящая из трех частей: сверху какой-нибудь дизайнерский рисунок, в середине содержимое, а внизу имя автора и год создания сайта.

При табличной верстке — это одна страница, на которой описана таблица из трех ячеек и их содержимого.

При верстке фреймами это могут быть целых четыре страницы: одна с описанием расположения, размеров рамок и страниц, которые должны быть показаны внутри каждой рамки, а еще три страницы с содержимым (одна с рисунком, вторая с содержанием и третья со сведениями об авторе). Кстати, если используется фреймовая верстка, то каждая рамка просматривается при помощи собственной полосы прокрутки, а главная страница и другие рамки не двигаются с места. При табличной верстке прокручивается вся страница сразу. Кроме того, при нажатии на гиперссылку можно указать, в какую именно рамку необходимо загрузить эту страницу.

Для своего сайта я выбрал табличную верстку, поскольку текстовая слишком проста для него (ведь у моего "хомяка" довольно сложный скелет). А вот фреймовую верстку я не хочу использовать, потому что с ней дольше возиться, появляется много полос прокрутки, сложнее навигация (т. е. способы перемещения посетителя по содержимому сайта).

И еще есть немаловажные моменты для последующей раскрутки "хомяка". Роботы с поисковых серверов отыскивают страницы по гиперссылкам. Если ваш сайт построен при помощи фреймов, то робот наткнется на страницу без всяких гиперссылок (а лишь с описанием того, в какую из рамок и что нужно загрузить), ваши страницы не попадут в специальную базу данных такого поискового сервера (в его индекс). Следовательно, никто не узнает о той информации, которая имеется у вас на сайте. Если даже вы обманете такого робота и специально напишете ему внутри описания рамок список гиперссылок на свои страницы, он, конечно, найдет ваши страницы, но потом может произойти казус. Проиндексировав (занеся в свою базу данных) такие страницы, поисковик когда-нибудь выдаст их пользователю по результатам его запроса, но это будет не вся страница, что вы задумывали показать, а только та часть, которая должна быть показана в определенной рамке. Если на такой странице не окажется никаких способов перейти к другим страницам, ваш посетитель окажется в ловушке. С этой страницы ему будет некуда уйти.

Создаем заготовку главной страницы сайта при помощи табличной верстки

Итак, я решил, что буду делать свой сайт при помощи табличной верстки и резинового дизайна. Что теперь?

Страница должна состоять из одной основной таблицы, которая представляет собой как бы скелет для всего содержимого. Для начала не буду делать слишком сложный скелет. Основу внешнего вида будущего сайта можно представить в виде табл. 4.1.

***Таблица 4.1.** Основа внешнего вида будущего сайта*

<table>
<tr><td colspan="3">Рисунок</td></tr>
<tr><td rowspan="3">Новости и прочая информация</td><td colspan="2"></td><td>Заголовок страницы</td></tr>
<tr><td colspan="3">Основное содержание</td></tr>
<tr><td colspan="3">Копирайт и e-mail</td></tr>
</table>

Вот так схематически будет выглядеть страница на моем сайте. Теперь посмотрим, как ее создать при помощи HTML.

Во-первых, таблица всегда находится внутри тега TABLE (листинг 4.4).

Листинг 4.4

```
<TABLE>
...
</TABLE>
```

Во-вторых, помимо указания того, что это сама таблица, необходимо описать, как располагаются ее строки, столбцы и отдельные ячейки. Строки описываются при помощи тега <TR> (table row — строка таблицы).

Каждая строка должна содержать внутри себя хотя бы одну ячейку, которая описывается посредством тега <TD> (table dot). Естественно, что ячейки, расположенные одна под другой, образуют столбец, но он специальным образом никак не отмечается в HTML — только строки и ячейки.

Примечание

Немного забегая вперед, хочу сказать, что в целях отладки (чтобы было понятно, как выглядят ячейки таблицы) необходимо задать границу таблицы. Для этого надо указать у тега <table> атрибут border="1". Тогда вокруг ячеек появится объемная граница, и мы явно увидим результат наших экспериментов.

Самая простая таблица — это одна ячейка внутри одной строчки (листинг 4.5, результат которого представлен на рис. 4.6).

Листинг 4.5

```
<TABLE>
 <TR>
   <TD>Таблица с единственной ячейкой</TD>
 </TR>
</TABLE>
```

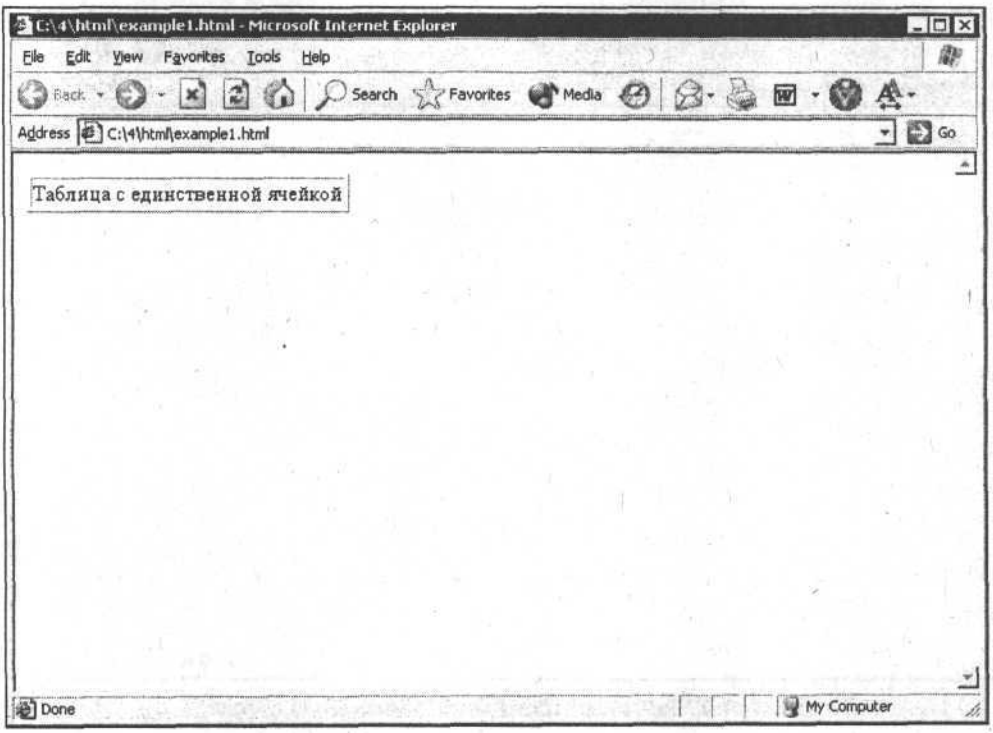


Рис. 4.6. Таблица с единственной ячейкой

Если я хочу добавить еще несколько столбцов в эту строчку, то `<TR>` (т. е. описание строки) остается прежним, а внутри него я напишу столько тегов `<TD>`, сколько хочу иметь столбцов в таблице (т. е. в этой единственной строке).

Например, таблица из трех столбцов будет описана таким образом, как представлено в листинге 4.6 (результат — на рис. 4.7).

Листинг 4.6

```
<TABLE>
 <TR>
   <TD>Первая ячейка</TD>
   <TD>Вторая ячейка</TD>
   <TD>Третья ячейка</TD>
 </TR>
</TABLE>
```

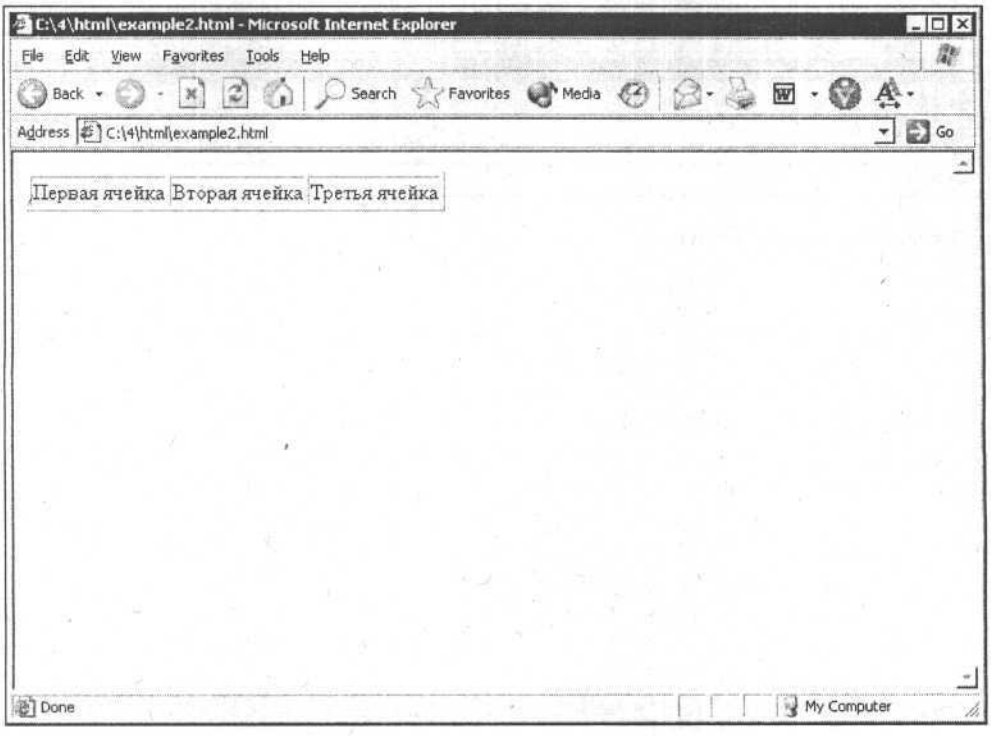


Рис. 4.7. Таблица с тремя ячейками в одной строке

Если мы хотим добавить количество строк, то, соответственно, надо описать столько тегов <TR>, сколько необходимо, при этом описание столбцов внутри всех строк должно совпадать (листинг 4.7, результат на рис. 4.8).

Листинг 4.7

```
<TABLE>
  <TR>
```

```
    <TD>Строка 1, ячейка 1</TD>
    <TD>Строка 1, ячейка 2</TD>
    <TD>Строка 1, ячейка 3</TD>
  </TR>
  <TR>
    <TD>Строка 2, ячейка 1</TD>
    <TD>Строка 2, ячейка 2</TD>
    <TD>Строка 2, ячейка 3</TD>
  </TR>
</TABLE>
```

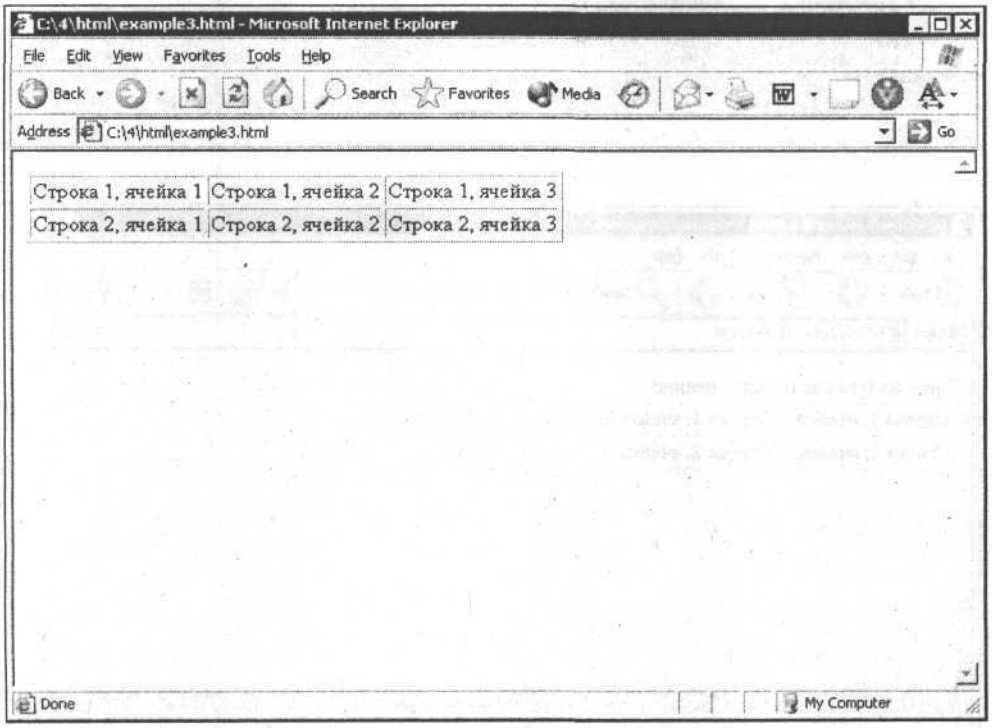


Рис. 4.8. Таблица из двух строк и трех столбцов

Внимание

Содержимое ячеек может находиться только внутри тегов <TD> и нигде больше, ни между <TR> и <TD>, ни между двумя ячейками <TD> и т. д. В противном случае содержание отобразится некорректно (листинг 4.8, рис. 4.9).

Листинг 4.8

```
<TABLE>
  <TR>Здесь ничего располагать нельзя!
     <TD>Строка 1, ячейка 1</TD>
     <TD>Строка 1, ячейка 2</TD>
     <TD>Строка 1, ячейка 3</TD>
  </TR>
  <TR>
     <TD>Строка 2, ячейка 1</TD>
     <TD>Строка 2, ячейка 2</TD>
     <TD>Строка 2, ячейка 3</TD>
   </TR>
</TABLE>
```

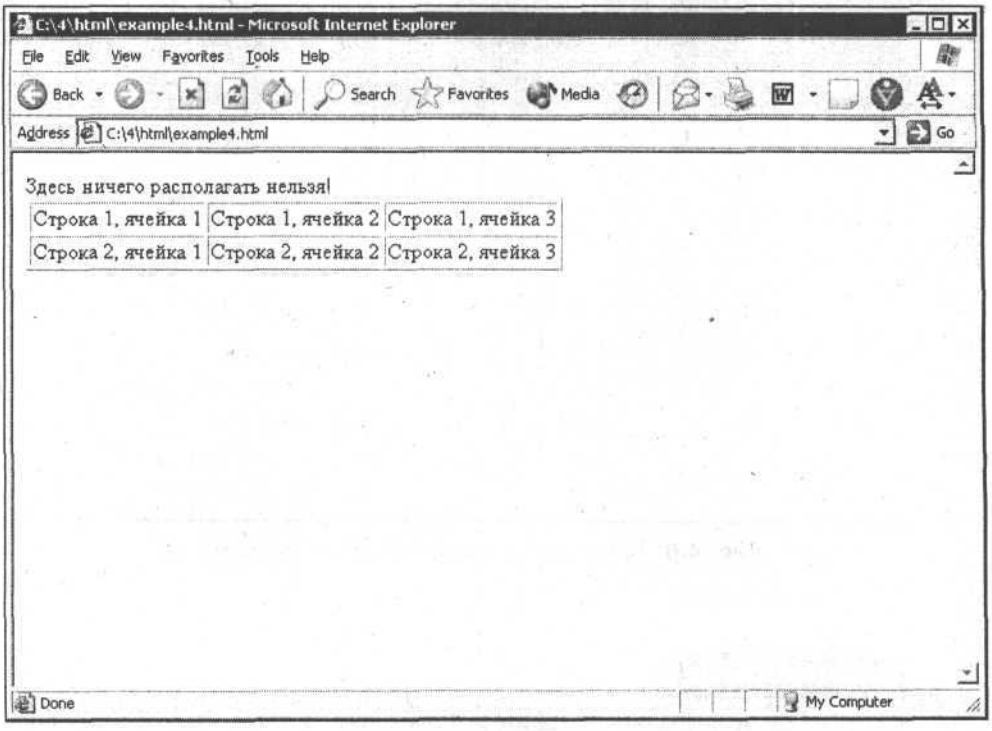


Рис. 4.9. Некорректное отображение содержания, написанного между тегами таблицы, а не внутри них

На первый взгляд не так уж и страшно — подумаешь, кусок текста выскочил перед самой таблицей. Но представьте себе, что вместо текста в ячейку вставлена еще одна таблица или фрагмент изображения. Вот тогда могут возникнуть серьезные проблемы с вашей страничкой.

Если необходимо объединить ячейки одной строки, то в ячейке, с которой начинается объединение, прописывается атрибут `colspan="n"` (где *n* — это число объединяемых ячеек одной строки, в т. ч. и начальная ячейка) (листинг 4.9, рис. 4.10).

Листинг 4.9

```
<TABLE>
  <TR>
     <TD colspan="2">Строка 1, ячейки 1 и 2</TD>
     <TD>Строка 1, ячейка 3</TD>
  </TR>
  <TR>
     <TD>Строка 2, ячейка 1</TD>
     <TD>Строка 2, ячейка 2</TD>
     <TD>Строка 2, ячейка 3</TD>
  </TR>
</TABLE>
```

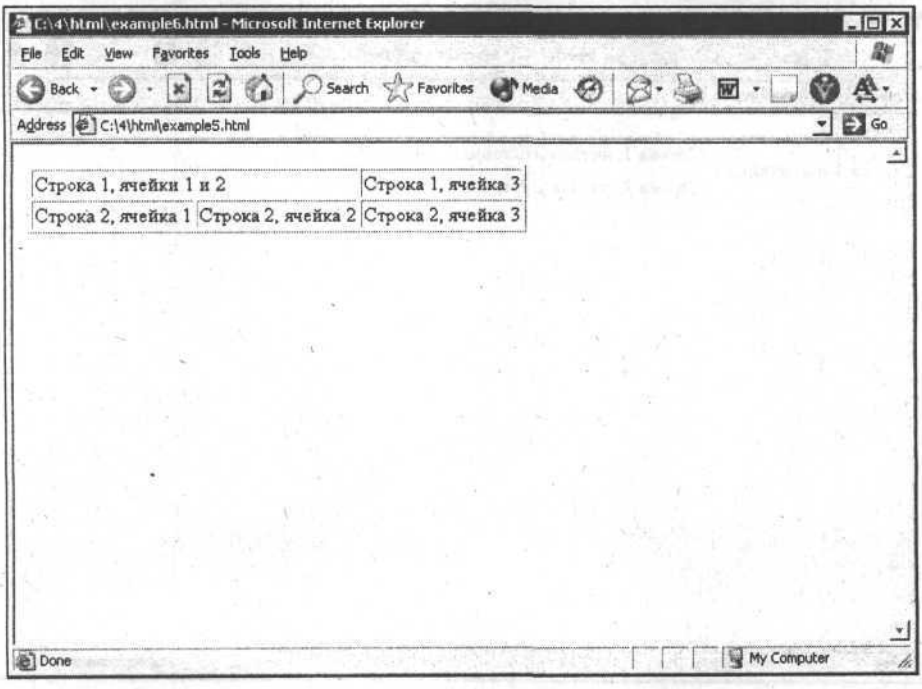


Рис. 4.10. Результат объединения в таблице двух ячеек одной строки

Если необходимо объединить ячейки одного столбца, то указывается атрибут `rowspan="n"`, где *n* — количество объединяемых в столбце ячеек (листинг 4.10, рис. 4.11). Соответственно, нам необходимо убрать описание лишней ячейки из второй строчки.

Листинг 4.10

```
<TABLE>
  <TR>
    <TD rowspan="2">Строка 1 и 2, ячейка 1</TD>
    <TD>Строка 1, ячейка 2</TD>
    <TD>Строка 1, ячейка 3</TD>
  </TR>
  <TR>
     <TD>Строка 2, ячейка 2</TD>
     <TD>Строка 2, ячейка 3</TD>
  </TR>
</TABLE>
```

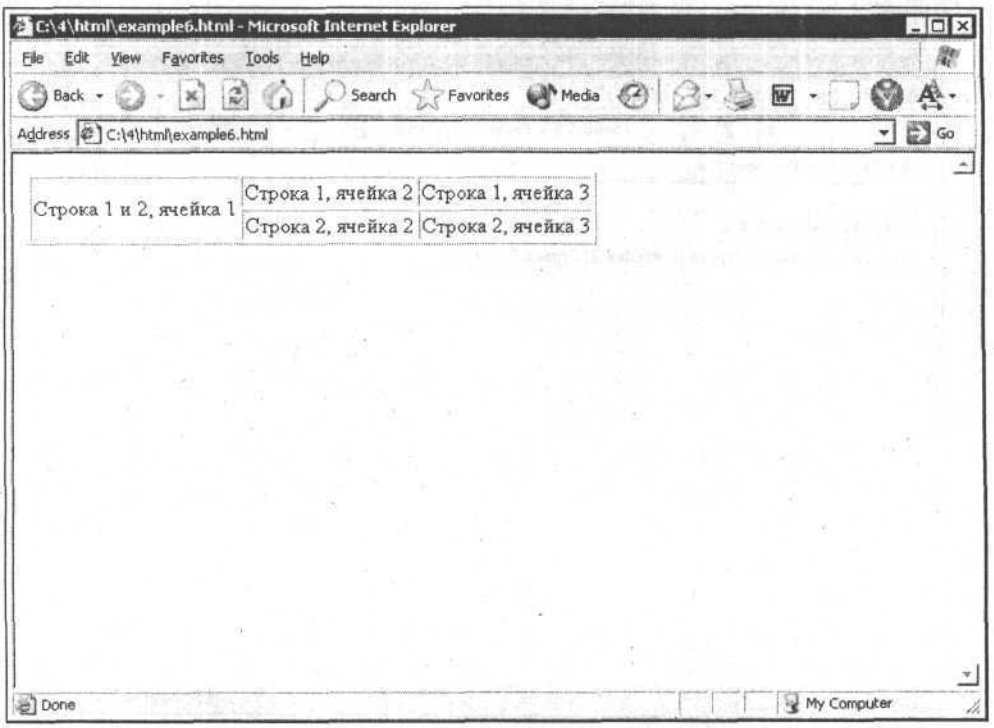


Рис. 4.11. Результат объединения в таблице двух ячеек одного столбца

Теперь полученных знаний достаточно, чтобы приступить к созданию каркаса нашей будущей страницы. Каркас, который я хочу получить на своем сайте, можно представить в табл. 4.2.

Таблица 4.2. *Таблица-каркас внешнего вида будущего сайта*

В этой таблице на самом деле всего четыре строчки и четыре столбца, но некоторые из ячеек объединены. В табл. 4.3 приведен внешний вид начальной заготовки, чтобы легче понять, как происходит построение каркаса и объединение соответствующих ячеек.

Таблица 4.3. *Заготовка для создания таблицы-каркаса*

1.1	1.2	1.3	1.4
2.1	2.2	2.3	2.4
3.1	3.2	3.3	3.4
4.1	4.2	4.3	4.4

Я пронумеровал ячейки таблицы, чтобы не запутаться, когда буду исправлять HTML-код (первая цифра соответствует номеру строки, вторая — номеру столбца). В листинге 4.11 представлен соответствующий HTML-код.

Листинг 4.11

```
<TABLE>
  <TR>
    <TD colspan="4">Строка 1, ячейки 1,2,3,4</TD>
  </TR>
  <TR>
    <TD rowspan="3">Строки 2,3,4, ячейка 1</TD>
    <TD>Строка 2, ячейка 2</TD>
    <TD>Строка 2, ячейка 3</TD>
    <TD>Строка 2, ячейка 4</TD>
  </TR>
  <TR>
```

```
    <TD colspan="3">Строка 3, ячейки 2,3,4</TD>
  </TR>
  <TR>
    <TD colspan="3">Строка 4, ячейки 2,3,4</TD>
  </TR>
</TABLE>
```

Теперь необходимо объединить соответствующие ячейки между собой, чтобы получить нужный вид таблицы. Рассмотрим этот процесс поэтапно:

1. Объединяем все ячейки первой строки. Их четыре, поэтому для верхней строки код будет выглядеть так, как представлено в листинге 4.12.

Листинг 4.12

```
<TR>
  <TD colspan="4">Строка 1, ячейки 1,2,3,4</TD>
</TR>
```

2. Вторая строка содержит четыре ячейки, первая из которых будет объединяться с двумя последующими в этом же столбце (т. е. необходимо указать все четыре ячейки, но у первой сделать описание: `rowspan="3"` (листинг 4.13)).

Листинг 4.13

```
<TR>
  <TD rowspan="3" style="width:170px">Строки 2,3,4, ячейка 1</TD>
  <TD style="height: 1%">Строка 2, ячейка 2</TD>
  <TD style="height: 1%">Строка 2, ячейка 3</TD>
  <TD style="height: 1%">Строка 2, ячейка 4</TD>
</TR>
```

3. Третья строка состоит из ячейки первого столбца (которой нет, т. к. она уже объединена в предыдущем примере) и трех следующих объединенных ячеек строки (они представлены в виде одной ячейки с атрибутом: `colspan="3"`). HTML-код показан в листинге 4.14.

Листинг 4.14

```
<TR>
  <TD colspan="3">Строка 3, ячейки 2,3,4</TD>
</TR>
```

4. Четвертая строка по конструкции аналогична третьей.
5. В итоге получится HTML-код, представленный в листинге 4.15. Результат — на рис. 4.12.

Листинг 4.15

```
<TR>
   <TD colspan="4">Строка 1, ячейки 1,2,3,4</TD>
</TR>
<TR>
   <TD rowspan="3">Строки 2,3,4, ячейка 1</TD>
   <TD>Строка 2, ячейка 2</TD>
   <TD>Строка 2, ячейка 3</TD>
   <TD>Строка 2, ячейка 4</TD>
</TR>
<TR>
   <TD colspan="3">Строка 3, ячейки 2,3,4</TD>
</TR>
<TR>
   <TD colspan="3">Строка 4, ячейки 2,3,4</TD>
</TR>
```

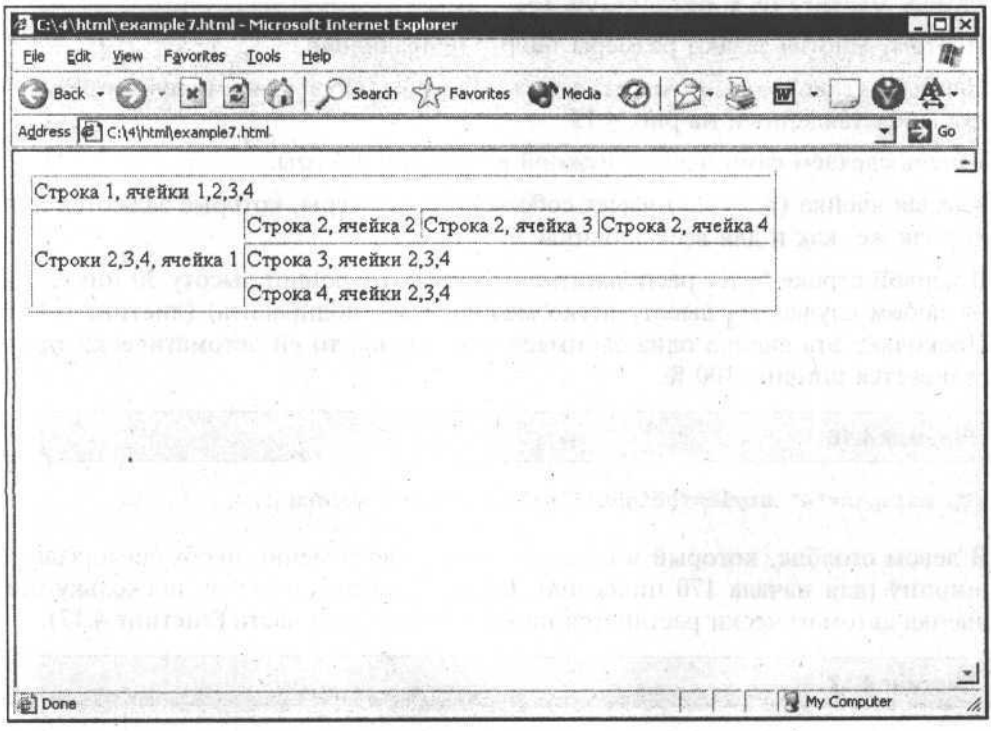


Рис. 4.12. Результат объединения в таблице-каркасе соответствующих ячеек

Итак, таблица нужной конфигурации получена, теперь необходимо растянуть эту таблицу на всю величину страницы, а ячейки сделать соответствующего размера.

Для задания размера таблиц необходимо указать их ширину и высоту. Так как я решил, что буду делать сайт посредством резинового дизайна, то размер буду указывать в процентах от ширины страницы (в данном случае по 100 %, чтобы таблица растянулась на всю ширину и высоту). Если содержимое не будет помещаться в таблицу, то ширина ее не должна меняться, а высота должна растягиваться по мере необходимости.

Как задаются размеры? На самом деле у тега `<TABLE>` нет атрибутов `width` и `height` (т. е. ширины и высоты). Их нужно задавать при помощи стилей (атрибут `style`) (`style="атрибут:значение; атрибут:значение; ..."`), однако редко кто утруждает себя написанием такого кода:

```
<TABLE style="width:100 %; height:100% ">.
```

Все привыкли к тому, что "всеядный" Internet Explorer все равно понимает запись такого вида:

```
<TABLE width="100 %" height="100 %">
```

или даже такого:

```
<TABLE width=100 % height=100 %>.
```

Поэтому многие задают размеры таблиц неправильно.

Написав `<TABLE style="width:100 %; height:100 %">`, мы получим результат, представленный на рис. 4.13.

Теперь сделаем сами ячейки нужной ширины и высоты.

Каждая ячейка (тег `<TD>`) имеет собственные размеры, которые задаются для нее так же, как и для всей таблицы.

В первой строке будет располагаться рисунок, имеющий высоту 30 пикселов (в любом случае эту высоту легко можно будет подправить) (листинг 4.16). Поскольку эта ячейка одна занимает всю строку, то ей автоматически присваивается ширина 100 %.

Листинг 4.16

```
<TD colspan="4" style="height:30px">Строка 1, ячейки 1,2,3,4</TD>
```

В левом столбце, который я оставил для бокового меню, необходимо задать ширину (для начала 170 пикселов). Высоту задавать не буду, поскольку эта ячейка автоматически растянется на всю оставшуюся часть (листинг 4.17).

Листинг 4.17

```
<TD rowspan="3" style="width:170px">Строки 2,3,4, ячейка 1</TD>
```

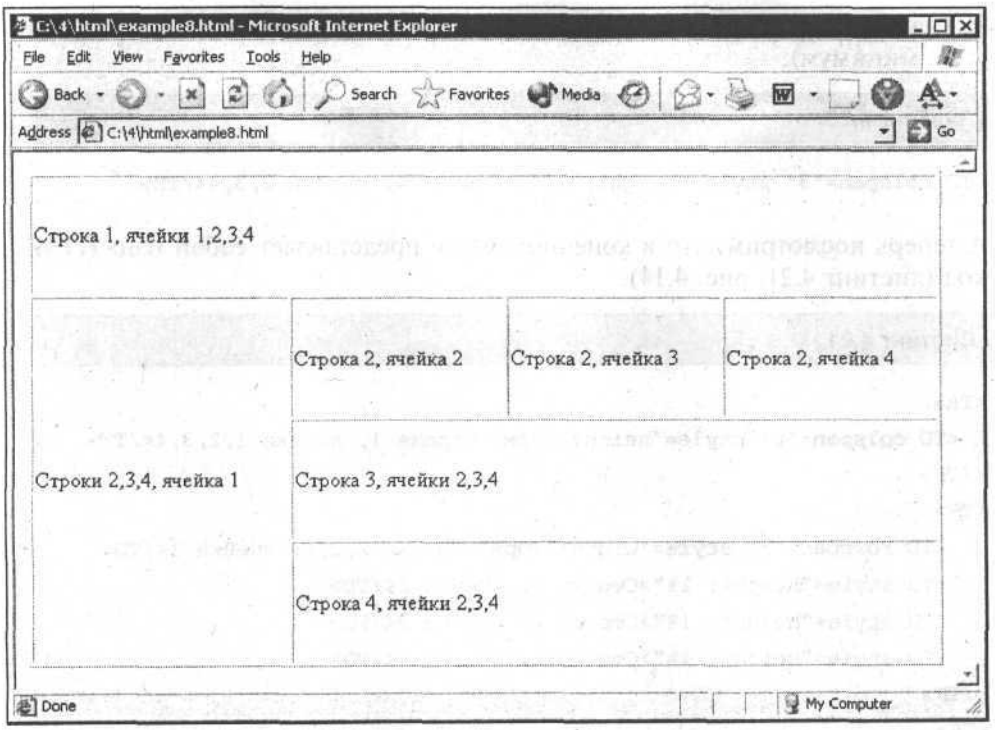


Рис. 4.13. Полученная ранее таблица растянута по всей ширине окна

Высоту второй строки, в которой расположены три одинаковые ячейки, я задам со значением 1 % (внутри каждой ячейки), как показано в листинге 4.18, чтобы они минимально растягивались сами по мере заполнения. Ширину ячеек трогать не буду — пусть остаются одинаковыми.

Листинг 4.18

```
<TD style="height:1%">Строка 2, ячейка 2</TD>
<TD style="height:1%">Строка 2, ячейка 3</TD>
<TD style="height:1%">Строка 2, ячейка 4</TD>
```

В объединенных ячейках третьей строки я не стану задавать никаких размеров (по умолчанию это объединенная ячейка, в которой планирую располагать основное содержание) — при использовании кода, показанного в листинге 4.19, она будет растягиваться вслед за всей таблицей.

Листинг 4.19

```
<TD colspan="3"> Строка 3, ячейки 2,3,4</TD>
```

И, наконец, в последней строке (листинг 4.20) я задам высоту также 1 % (т. е. минимум).

Листинг 4.20

```
<TD colspan="3" style="height: 1%"> Строка 4, ячейки 2,3,4</TD>
```

А теперь посмотрим, что в конечном итоге представляет собой наш HTML-код (листинг 4.21, рис. 4.14).

Листинг 4.21

```
<TR>
  <TD colspan="4" style="height:30px">Строка 1, ячейки 1,2,3,4</TD>
</TR>
<TR>
   <TD rowspan="3" style="width:170px">Строки 2,3,4, ячейка 1</TD>
   <TD style="height: 1%">Строка 2, ячейка 2</TD>
   <TD style="height: 1%">Строка 2, ячейка 3</TD>
   <TD style="height: 1%">Строка 2, ячейка 4</TD>
</TR>
<TR>
   <TD colspan="3">Строка 3, ячейки 2,3,4</TD>
</TR>
<TR>
   <TD colspan="3" style="height: 1%">Строка 4, ячейки 2,3,4</TD>
</TR>
```

На рис. 4.14 видно, что первая строка имеет высоту, явно меньшую 30 пикселов. Это произошло потому, что при задании абсолютных значений размеров, ячейка все равно растягивается и сжимается по содержимому. Если вставить в нее рисунок высотой 30 пикселов, то у ячейки будет такая же постоянная высота, но рисунок вставлять я пока не буду, этим займусь позже.

Теперь необходимо выровнять содержимое внутри ячеек. Вы видите, что сейчас текст внутри ячеек расположен не вверху по высоте, а в середине. Такое расположение можно подправить указанием вертикального выравнивания, атрибута `valign` (vertical align), который задается у каждой конкретной ячейки (`<TD valign="top">`). Существуют три способа выравнивания по высоте:

- по верхнему краю (`top`);
- по середине (`middle`);
- по нижнему краю (`bottom`).

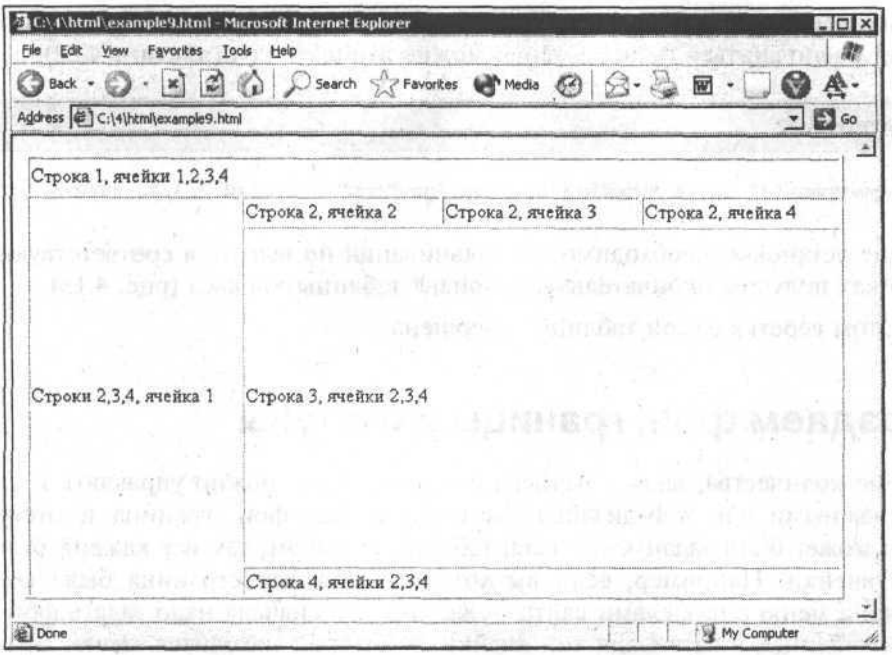


Рис. 4.14. Каркас таблицы с заданными размерами ячеек

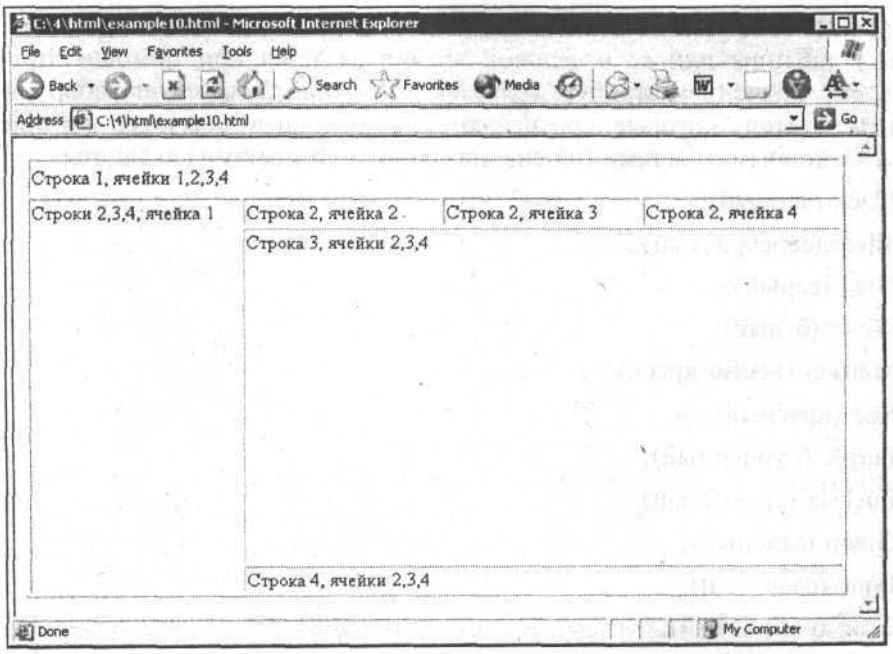


Рис. 4.15. Окончательный вариант таблицы-каркаса

В данном случае я использую способ top. Например, для ячейки, в которой будет располагаться главное меню, можно записать код (листинг 4.22).

Листинг 4.22

```
<TD rowspan="3" style="width:170px" valign="top" > Строки 2,3,4, ячейка 1</TD>
```

После установки необходимого выравнивания по высоте в соответствующих ячейках получим окончательный вариант таблицы-каркаса (рис. 4.15).

На этом верстка самой таблицы завершена.

Создаем фон, границы и отступы

Кроме количества, вида и размера ячеек таблицы, можно управлять и такими важными для web-дизайна элементами, как фон, граница и отступы. Фон может быть задан как у всей таблицы целиком, так и у каждой отдельной ячейки. Например, если вы хотите, чтобы вся страница была серого цвета, а меню с разделами сайта — желтого, то сначала надо задать фон для всей таблицы, а затем для той ячейки, в которой находится меню. Браузер будет заполнять все серым фоном, а потом уже нужную ячейку желтым цветом поверх серого. Однако это будет происходить так быстро, что посетитель не успеет заметить серый цвет под желтым.

Фон у таблицы или ее отдельной ячейки задается при помощи атрибута bgcolor. Значением атрибута является цвет фона. Существует набор специальных цветов, которые обозначаются с помощью обычных английских слов. Основными цветами (их еще называют web-цветами) являются:

- Black (черный);
- Silver (серебристый);
- Gray (серый);
- White (белый);
- Maroon (темно-красный);
- Red (красный);
- Purple (пурпурный);
- Fuchsia (сиреневый);
- Green (зеленый);
- Lime (салатный);
- Olive (оливковый);
- Yellow (желтый);

- Navy (морской);
- Blue (синий);
- Teal (темно-бирюзовый);
- Aqua (водный).

Браузеры отличаются друг от друга даже тем, что, помимо указанного набора названий цветов, в некоторых из них существует еще и свой набор, где попадается нечто вроде darknightblue (темно-ночной-синий).

Цвет может также обозначаться и при помощи цифр. Любой из цветов может быть описан при помощи сочетаний трех цветов: красного, зеленого и синего. Это одна из так называемых моделей цвета — модель RGB (Red-Green-Blue). Каждая из этих трех составляющих может описываться целым числом от 0 до 255 (так же, как и части цифрового обозначения домена, если помните). Таким образом можно получить миллионы сочетаний цветов. Белый цвет образуется, когда все три значения равны 255; черный — когда все три значения равны 0; любые оттенки серого получаются, если все три составляющие равны между собой. Ну а все остальные цвета можно скомбинировать по специальным правилам, вроде тех, которым учат в школе на уроках рисования (если смешать синий и желтый, то получится зеленый и т. п.). Набор, которым обозначается цвет, представляет собой шесть цифр с символом # (решетка) перед ними. Он строится по принципу: `#rrggbb`, где `rr`, `gg` и `bb` —значения составляющих (red, green, blue) от 0 до 255, но только в шестнадцатеричной системе счисления. Например, `#F5F5DC` — это бежевый цвет в модели RGB, описанный как 245, 245, 220.

Есть несколько простых принципов, следуя которым можно создавать более-менее правильные цветовые композиции на сайте. Считается, что на сайте не должно быть более трех основных цветов, желательно, чтобы они были подобраны в соответствии с законами сочетания цветов в композиции. Необходим достаточно высокий уровень контраста между фоном и шрифтом. Например, синий шрифт на черном фоне читать очень трудно, а порой и невозможно. Я всегда стараюсь использовать на своих страничках наиболее читабельную схему — темный шрифт на светлом фоне. Некоторые небольшие элементы страницы, имеющие декоративное значение, я оформляю светло-серым цветом `#D4D4D4` (листинг 4.23).

Листинг 4.23

```
<td bgcolor="#D4D4D4">
```

Границы бывают у элементов, которые являются контейнером для содержимого (например, это таблицы, фреймы и т. п.). Задаются границы при помощи атрибута `border`, например: `border="1"` (где 1 — значение толщины границы). Когда граница задается подобным образом для таблицы, то она

показывается у всех ячеек такой таблицы. Однако в таком случае границы получаются грубые и некрасивые, применить их можно к очень немногим элементам. Задавать границы можно и при помощи стилей. Подробно о стилях мы будем говорить позднее, а сейчас я просто расскажу, как это сделать. Однако прежде должен обратить ваше внимание на один важный момент. При задании границы с помощью стилей, она появляется только у внешних частей того элемента, у которого задана. Посмотрим, что получится, если воспользоваться различным способом задания границ. На рис. 4.16 представлена одна и та же таблица, только в первом случае (вверху рисунка) границы заданы при помощи стилей, а во втором (внизу) — при помощи атрибута `border`.

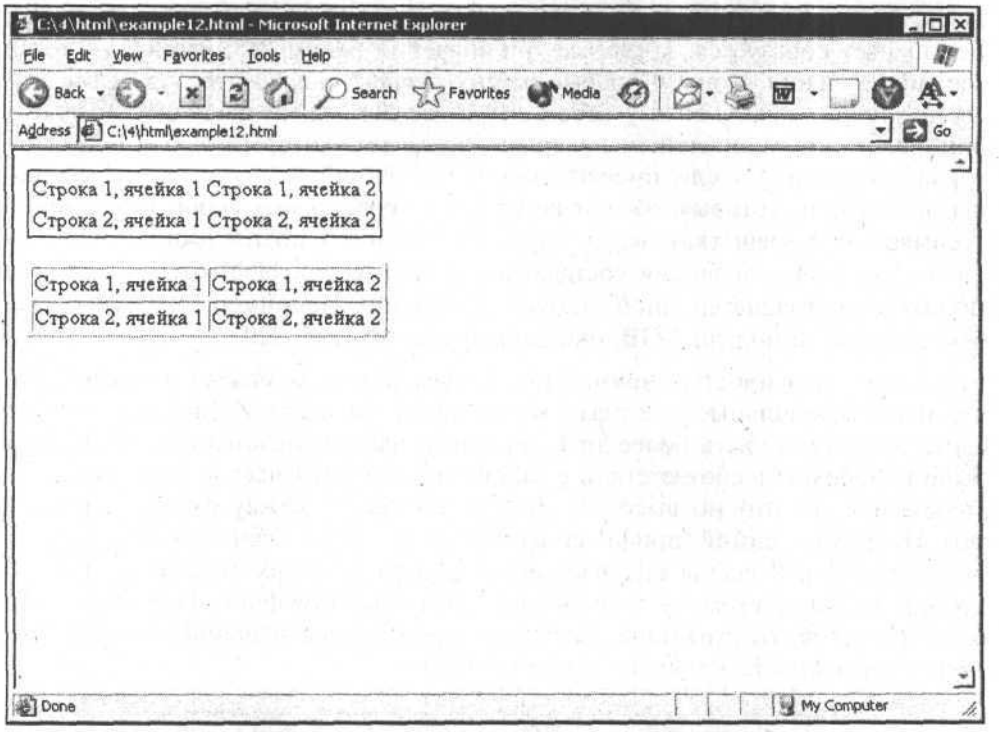


Рис. 4.16. Одна и та же таблица, заданная разными способами

У атрибута `border` существует несколько параметров, которые могут быть заданы при помощи стилей (листинг 4.24). К таким параметрам можно отнести толщину границы (`border-width`), цвет (`border-color`), способ начертания (`border-style`) и т. д.

Листинг 4.24

```
<table style="border:1px;border-style:solid;border-color:black">
```

Как видите, запись получается слишком громоздкой и сложной, поэтому была учтена возможность сокращенного задания стилей. Поскольку все параметры относятся к атрибуту `border`, а значения этих параметров спутать невозможно (цвет не может быть равен одному пикселу, а толщина не бывает черной), то эту запись можно сократить, указывая только значения различных параметров атрибута `border` через пробел, как показано в листинге 4.25.

Листинг 4.25

```
<table style="border:1px solid black">
```

Таким образом, одной такой записью вы даете сразу три инструкции по отображению границы.

И последнее, что необходимо сделать на этапе подготовки шаблона, это создать в таблице нужные отступы (внутри ячеек, между ячейками, для основной таблицы-каркаса, да и для самой страницы тоже).

Итак, как вы помните, содержимое страницы находится внутри тега `<body>`. Нам нужно задать отступы для всей страницы. С этой целью укажем атрибут `margin` и его значение (величину отступа в пикселах) при помощи стилей: `style="margin:0"`. Чтобы понять, как работает этот атрибут, вы можете попробовать изменить его значения.

Для таблиц и ячеек картина несколько иная. У тега `<table>` есть два атрибута — `cellpadding` и `cellspacing`, значения которых также указываются в пикселах (по умолчанию равные соответственно 1 и 2). Таким образом, даже при отсутствии этих атрибутов на странице, браузер все равно будет показывать такие отступы, как если бы он увидел внутри открывающей части тега запись:

`cellpadding="1" cellspacing="2".`

`Cellpadding` — это величина отступа между границей ячеек и их содержимым.

`Cellspasing` — величина отступа между самими ячейками внутри таблицы.

Пример четырех таблиц с различными отступами приведен на рис. 4.17. Таблицы расположены одна под другой и каждая из них состоит из двух ячеек.

Теперь я вернусь к заготовленному каркасу страницы и добавлю фон, границы и отступы там, где они требуются. Получится итоговый каркас страницы (листинг 4.26, рис. 4.18).

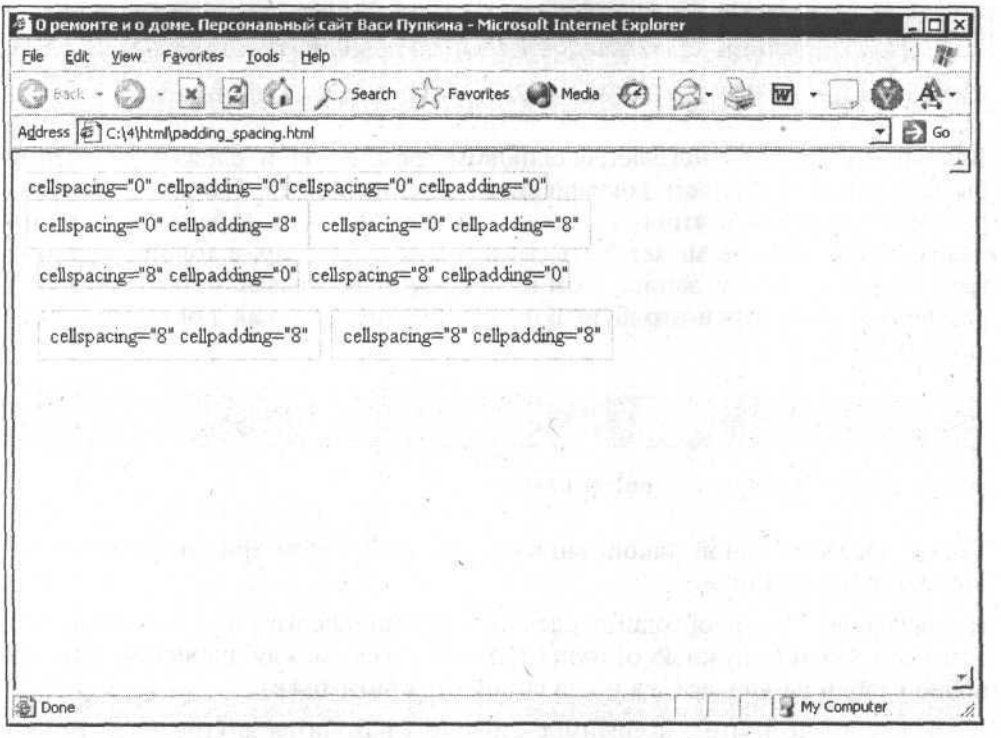


Рис. 4.17. Различные отступы внутри и снаружи таблицы и ее ячеек

Внимание

Важно помнить, что это не окончательный шаблон. По мере работы над сайтом будут добавляться еще теги и атрибуты (в конечном итоге в шаблоне должен получиться почти полностью внешний вид сайта, за исключением основного содержимого, его заголовка и т. п.).

Листинг 4.26

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
<title>О ремонте и о доме. Персональный сайт Васи Пупкина</title>
<link type="text/css" rel="stylesheet" href="css/style.css">
</head>

<body margin="0">
```

```
<!-- Начало таблицы-каркаса-->
<table style=" width:100%;height:100%;" cellspacing="4" cellpadding="0">
 <tr>
   <td colspan="4" style="height:60px;border:1px solid #d4d4d4">
    <!-- Начало верхнего блока страницы-->
      <table style="width:450px;height:30px">
       <tr>
        <td> testetste</td>
       </tr>
      </table>
    <!-- Конец верхнего блока страницы -->
  </td>
</tr>
<tr>
<td rowspan="3" style="width:20%;border:1px solid #d4d4d4" valign="top">
    <!-- Начало блока бокового меню -->
       <table style="width:100%;height:1%">
        <tr>
         <td> </td>
        </tr>
       </table>
    <!-- Конец блока бокового меню -->
    </td>
    <td style="height: 1%" bgcolor="#d4d4d4"> </td>
    <td style="height: 1%">
    <!-- Заголовок основного содержимого  страницы -->
      
     </td>
    <td style="height: 1%" bgcolor="#d4d4d4"> </td>
  </tr>
  <tr>
    <td colspan="3" valign="top" style="border:1px solid #d4d4d4">
     <!-- Начало основного содержимого страницы-->
      
     <!-- Конец основного содержимого страницы -->
    </td>

  </tr>
  <tr>
    <td colspan="3" style="height: 1%">
```

```
      <!-- Начало нижнего блока-->
       <table width="100%" height="1%">
        <tr>
           <td height="20px" colspan="2" bgcolor="#d4d4d4"> </td>
        </tr>
        <tr>
          <td align="center"> </td>
          <td > </td>
        </tr>
       </table>
      <!-- Конец нижнего блока-->
     </td>
    </tr>
  </table>
  <!-- Конец таблицы-каркаса -->
</body>
</html>
```

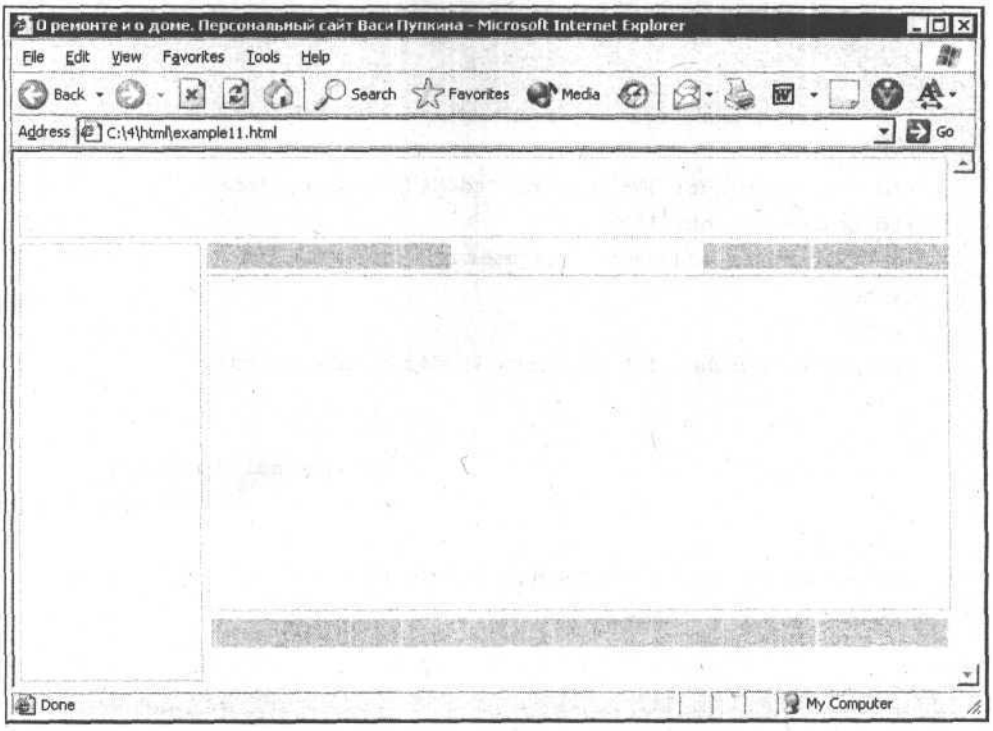


Рис. 4.18. Таблица-каркас с установленными границами, фоном и отступами

Что такое "рыба" и для чего ее используют

Когда каркас вашего сайта готов, необходимо наполнить его каким-либо содержимым, чтобы можно было оценить, как будет выглядеть ваша web-страничка в будущем. Естественно, что для этого совсем не обязательно заполнять страницу реальным содержимым (заголовками, текстом, гиперссылками, картинками). Бывает так, что нужно кому-то показать, как будет выглядеть ваш сайт в смысле дизайна, но у вас нет еще готового содержимого. Или вам надо прикинуть, что необходимо подправить в каркасе, где добавить нужное выравнивание содержимого ячеек таблицы и т. д. Для этих целей достаточно любого текста (или, как еще говорят, "рыбы").

"Рыбой" обычно называют произвольное, чаще всего бессмысленное содержимое, служащее заменой реальному. Когда-то я принимал участие в КВН (Клуб веселых и находчивых). Если какая-то песня была готова только наполовину, но нужно репетировать именно с ней, то придумывали "рыбу". Это был текст, соответствующий тому, что должен быть в реальности (по количеству куплетов, по размеру песни, по мелодии), но представлял он собой полную чушь. Например, на известную песенку из "Трех мушкетеров" можно было "наложить рыбу", которая звучала так: "Опять хрипит потертое весло и ветер молодых совсем не стану, но если дать вам в руки ремесло, куда же вам картинку по карману".

Имеются три разновидности "рыбы" (во всяком случае, мне известны три).

Первая — для самых ленивых. Это "рыба", состоящая из двух-трех слов. Она хороша тем, что для ее создания не нужно тратить много времени — написал эти три слова и скопировал в нужном количестве в зависимости от требуемого размера текста. Плоха она с точки зрения соответствия реальному тексту. В настоящем тексте могут быть и длинные слова, а не только такие, как в "рыбе". В некоторых случаях (например, при выравнивании текста по ширине), если встречаются длинные слова, могут возникать большие промежутки между ними, что некрасиво, а вы этого при подобной "рыбе" и не заметите. Кроме того, "рыба" из двух-трех одинаковых слов будет резать глаз тому, кто будет на нее смотреть, что помешает правильно оценить, красив ли сайт.

Вторая разновидность "рыбы "— чужой текст. Это содержимое, взятое откуда угодно, хоть из файла помощи по MS Windows или из электронного учебника по ботанике. Такое содержимое уже соответствует внешнему виду обычного текста, но не соответствует общему направлению темы сайта. Пусть этот вариант немного лучше первого, но все равно приятнее видеть что-то сходное с тематикой сайта. Вот так и появляется еще одна разновидность "рыбы".

Третьей разновидностью является псевдореальный текст. Это некоторое содержимое, которое в большей степени соответствует будущему контенту,

только оно может быть еще не обработано — не исправлены ошибки, не созданы гиперссылки, не вставлены картинки.

Целесообразнее всего делать "рыбу" третьей разновидности. Правда, не всегда в наличии имеется тематический текст, который можно использовать, да к тому же необходим рисунок, пример шаблона с "рыбой" и без нее. Исходя из этих условий, я чаще всего использую вторую разновидность "рыбы". Пример быстро приготовленной "рыбы" для того каркаса, который я использовал, приведен на рис. 4.19.

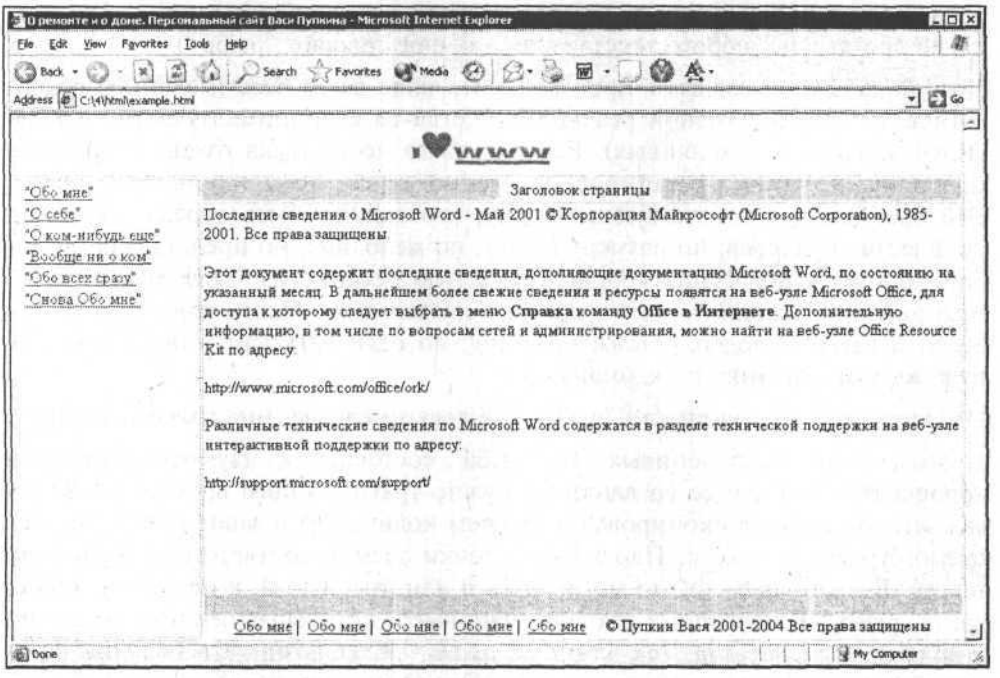


Рис. 4.19. Таблица-каркас с "рыбой"

Глава 5

Создание и заполнение реальных web-страниц. Чудо рождения

Превращаем текст в HTML

Чаще всего основным содержимым web-страниц является текст, и от его оформления зависит многое: и качество восприятия информации, и комфортность пользователя, и интерес посетителя к вашему сайту.

К оформлению текста можно отнести:

- структуру — результат разбиения текста на заголовки, абзацы, списки и т. д.;
- выравнивание — выбор расположения текста по левому краю, по правому, по центру или по ширине;
- способ выделения важной информации;
- цвет, шрифт и начертание.

Прежде чем приступать к превращению обычного текста в HTML-код, нужно понять, каковы должны быть правила построения текста для web-страниц. Дело в том, что чтение пользователями текста на сайте совершенно не похоже на чтение обычной книги.

Во-первых, многие оплачивают связь с Интернетом из своего кармана, следовательно, не намерены долго выискивать драгоценную информацию, они, как правило, путешествуют по Интернету быстро — как только найдут необходимые сведения, сразу отключаются.

Во-вторых, пользователи избалованы огромным количеством сайтов по сходной тематике и поэтому не будут особо возиться с одним сайтом, чтобы понять, нужен ли он им вообще, они скорее уйдут на другой.

В-третьих, для пользователей часто более важна поверхностная информация о чем либо, нежели глубокие фундаментальные исследования.

Процесс поиска и восприятия информации рядовым пользователем состоит из следующих моментов:

1. Пользователь делает запрос на поисковом сервере.
2. По результатам поиска он выбирает то, что наиболее соответствует его потребностям, но в то же время не содержит ненужной информации.
3. Открыв заинтересовавший его сайт в новом окне браузера, пользователь сразу захочет получить информацию о том, что же является основным содержанием открывшейся страницы, где расположены необходимые сведения;
4. Найдя нужное место, он читает текст, делая вывод о пригодности и полноте полученной информации.
5. Если пользователь удовлетворен полученными сведениями, то сохраняет эту информацию на диске.

Предположим, что человек нажал на ссылку, в которой указан ваш сайт. Первое, что он должен увидеть, когда страница откроется, это заголовок, говорящий о том, куда пользователь попал, и что он может узнать. Для примера я рассмотрю статью моего сайта, в которой собираюсь рассказать о том, как надо менять раковину в ванной комнате. Заголовок страницы, на которой расположится статья, будет звучать так: "Замена старой раковины и установка новой". Если посетитель хочет прочитать именно об этом, то сразу поймет, что попал в нужное место, а если его этот вопрос не интересует, то и не будет зря терять время.

При написании текстов для своих сайтов я придерживаюсь некоторых правил, которые являются весьма полезными:

- ❒ не делать длинных абзацев и предложений;
- ❒ в заголовках каждой страницы писать четко и понятно то, о чем она рассказывает;
- ❒ не помещать на одной странице очень много информации, а наоборот — постараться часть излишней информации скрыть от пользователя за гиперссылкой, ведущей к более подробному раскрытию темы;
- ❒ при написании текстов использовать принцип "начинать с конца" (т. е. сразу раскрывать суть вопроса, а только затем описывать основное содержимое статьи);
- ❒ использовать маркированные списки при перечислении чего-либо вместо обычного указания через запятую.

Почему я следую всем этим правилам, сейчас объясню.

Известно, что текст с большими абзацами читать гораздо труднее, чем с более мелкими. Дело в том, что в плане восприятия информации человек устроен несколько ограниченно. Когда он впервые осознает какие-то сведения, то держит их в кратковременной памяти (памяти, рассчитанной на воспри-

ятие текущего момента времени), а потом записывает в долговременную память (обычную). Если абзац длинный (а абзац, как известно, это некоторая законченная мысль или часть повествования), то человек читает и читает, держа все в кратковременной памяти. У мозга нет возможности сделать передышку и отложить полученную информацию в долговременную память. Но размер кратковременной памяти ограничен, и информация, полученная в начале такого длинного абзаца, частично теряется. Если же размер абзаца не очень велик, то обработка такого текста в мозгу происходит быстро. Длинные предложения также воспринимаются с трудом.

Необходимо четко осознать само понятие гипертекста и гиперссылок. Гипертекст — это основная идея работы во Всемирной паутине. Когда информации много, необходимо ограничивать объем и глубину сведений, поступающих к пользователю с web-страницы. Например, вы описываете расчет каких-нибудь параметров при помощи теоремы Пифагора. Среди ваших посетителей могут оказаться те, кто помнит эту теорему, и те, кто уже успел ее забыть. В книге вы смогли бы лишь написать что-нибудь вроде: "на основании теоремы Пифагора".

В Интернете все по-другому. Во фразе "на основании теоремы Пифагора" слова "теоремы Пифагора" необходимо сделать гиперссылкой, ведущей на страницу, где описывается эта теорема. И тогда тот, кто помнит школьную программу по геометрии, просто прочтет это как текст, а тот, кто забыл, имеет возможность нажать на гиперссылку и увидеть подробное описание.

Еще один ценный совет — "начинайте с конца". Чтобы привлечь пользователя к тексту, надо сразу преподнести ему главную мысль (это касается и текста и заголовков тоже). Представьте себе продавца, у которого вы спросили, имеется ли в продаже сырокопченая колбаса. Если продавец работает по принципу "начинайте с конца", то скажет: "Да, есть. Стоит столько-то. Еще есть отличная вареная колбаса сегодняшнего завоза. Могу также порекомендовать вам отличный паштет". С таким продавцом у вас не возникнет проблем — получив необходимую информацию, вы уже сами решаете, что делать дальше. Если же он начинает отвечать примерно так: "Вы знаете, сегодня нам завезли отличную вареную колбасу. А каков паштет! М-м... Пальчики оближете... Что вы спрашивали — сырокопченая? Но, какой же паштет сегодня отличный! Просто отличный". Представили свою реакцию? Такая же реакция может возникнуть у посетителей сайта на долгие описательные или рекламные тексты вместо главной сути.

Для примера возьму одну из статей, которую планирую разместить на сайте в ближайшем будущем. Я создаю из обычного текста его HTML-версию, адаптированную для читателей web-страниц. Итак, начинаю писать статью:

> Замена старой раковины и установка новой.
>
> В этой статье я расскажу о том, как менял раковину в ванной комнате. Вместо старой раковины я решил установить современную раковину на "пьедестале".

> Моя старая раковина была с небольшой трещиной, и я решил заменить ее на новую. Кронштейны, на которых держалась раковина, были закреплены болтами, вбитыми в стену. Я аккуратно снял раковину с них и очистил поверхность стены от остатков плитки. Поскольку я собирался перекладывать керамическую плитку в туалете, то просто выбил молотком болты с обратной стороны...

Стоп. Это плохой стиль — слишком много описания, не относящегося к самой теме статьи. Особенности, с которыми я столкнулся при работе, для подавляющего большинства читателей будут не интересны, потому что у некоторых может быть совмещенный санузел, а другие не собираются снимать плитку в туалете. Такое "пустословие" необходимо убрать. Я немного подумал и написал новую версию этого абзаца:

> Старая раковина крепилась на кронштейнах. Я перекрыл воду, отсоединил сливной сифон и колено, затем снял саму раковину, очистил поверхность стены от старой плитки и извлек крепежные болты. После этого я приклеил на пустое место такую же плитку, оставшуюся от прежнего ремонта, замазал щели фугой и выдержал сутки, чтобы плитка хорошенько "схватилась".

Уже немного лучше. Это первый абзац статьи. В нем заключено конкретное описание того, как надо снимать старую раковину. Второй абзац будет посвящен разметке местоположения крепежа для новой раковины.

> Раковину "на пьедестале", которую собрался установить, я просто поставил к стене, выбрав оптимальное положение, обвел фломастером контуры пьедестала на полу, отметил место под крепления через отверстия в задней стенке раковины. Эти отверстия расположены наискосок и не симметрично (чтобы можно было легко отрегулировать строго горизонтальное положение раковины после установки крепежа).

И далее текст идет в таком же духе.

Очень хорошо воспринимаются пользователями списки — гораздо лучше, чем простой текст. Поэтому в том месте статьи, где надо описать необходимый инструмент, я оформляю текст именно в таком виде. Чтобы понять эту разницу, посмотрите на два следующих абзаца.

Можно написать так:

> Вам понадобятся такие инструменты: дрель (желательно, чтобы она была с перфоратором); сверло с победитовым наконечником; крепежные болты с дюбелями (продаются практически в любом строительном магазине).

А можно написать и по-другому:

> Вам понадобятся такие инструменты:
>
> - дрель (желательно, чтобы она была с перфоратором);
> - сверло с победитовым наконечником;
> - крепежные болты с дюбелями (продаются практически в любом строительном магазине).

Вы почувствовали, что во втором случае читать текст намного проще?

Создаем абзацы, переводы строк и разделители

Редко кто набирает текст прямо в HTML-коде, если этот текст достаточно велик. Устают глаза, неудобно смотреть на результат, да и вообще, обычно удобнее создавать текст в специальном редакторе.

После создания текста необходимо превратить его в HTML-код. Поместив в пустую страницу (шаблон) текст, взятый из редактора (например, из MS Word), вы получите приблизительно такую картину, что представлена на рис. 5.1.

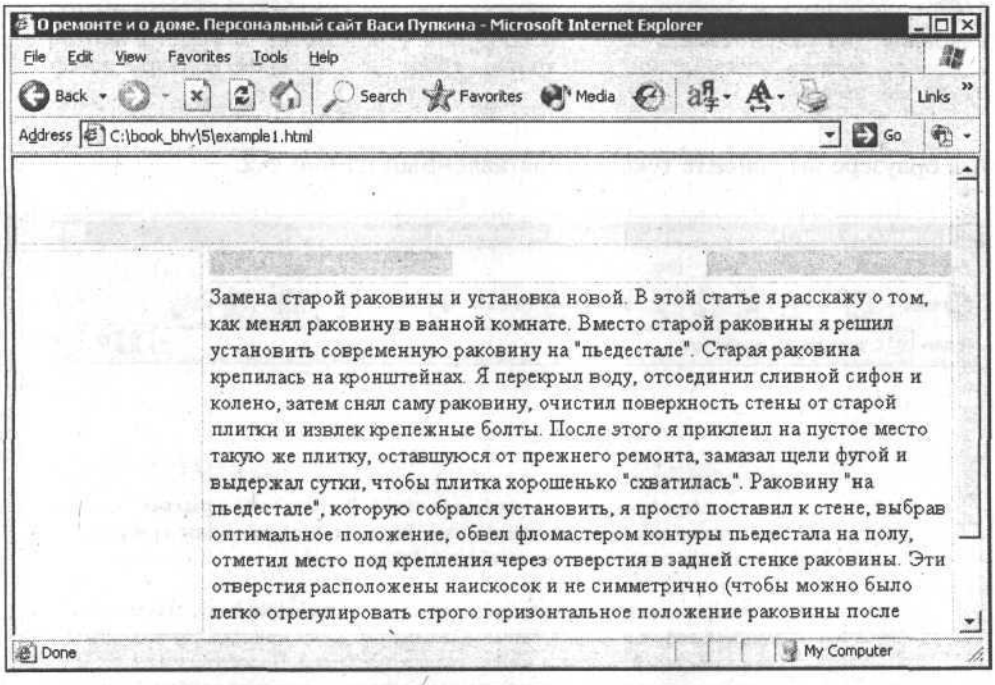


Рис. 5.1. Текст, скопированный из текстового редактора

Думаю, с этим текстом все ясно — его невозможно читать. Чтобы он превратился в HTML-код, нужно немного постараться.

Начнем с абзацев. Я пользуюсь несколькими способами разбиения текста на абзацы.

Первый способ. Абзац может быть создан при помощи тега <p>. При использовании этого тега браузер воспринимает абзацем все, что заключено внутри него. Для примера я помещу один из абзацев приведенного текста статьи внутрь тега <p>.

HTML-код будет выглядеть так, как представлено в листинге 5.1.

Листинг 5.1

```
Замена старой раковины и установка новой. В этой статье я расскажу о том,
как менял раковину в ванной комнате. Вместо старой раковины я решил уста-
новить современную раковину на "пьедестале".
<p>
Старая раковина крепилась на кронштейнах. Я перекрыл воду, отсоединил
сливной сифон и колено, затем снял саму раковину, очистил поверхность
стены от старой плитки и извлек крепежные болты. После этого я приклеил
на пустое место такую же плитку, оставшуюся от прежнего ремонта, замазал
щели фугой и выдержал сутки, чтобы плитка хорошенько "схватилась".
</p>
Раковину "на пьедестале", которую собрался установить, я просто поставил
к стене, выбрав оптимальное положение, обвел фломастером контуры пьеде-
стала на полу...
```

А в браузере вы увидите текст, представленный на рис. 5.2.

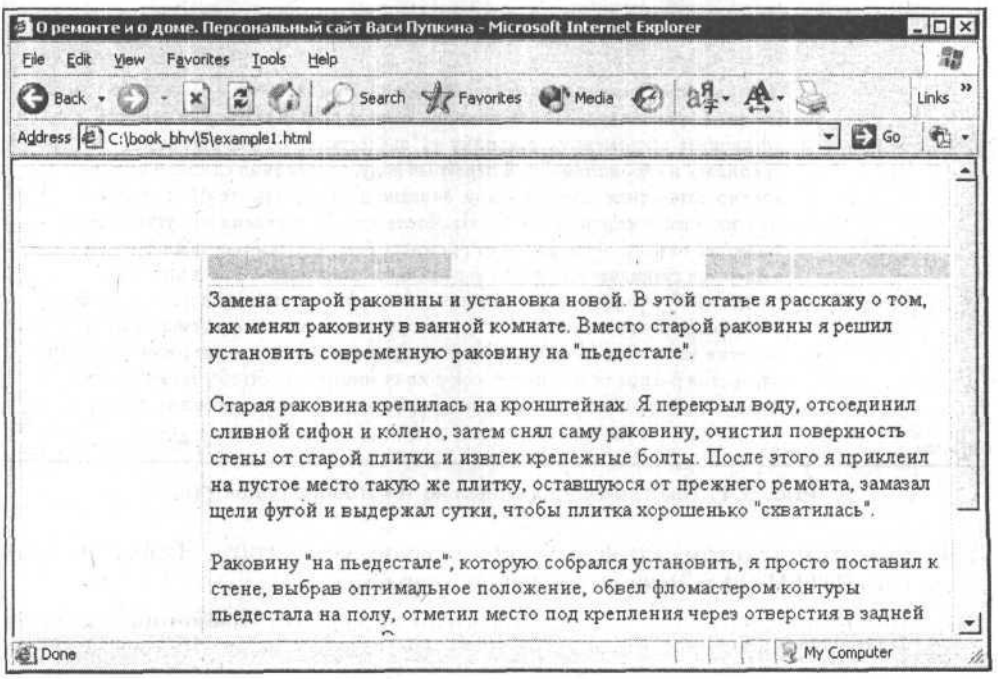


Рис. 5.2. Текст, разделенный на абзацы при помощи тега

Второй способ — разбиение на абзацы при помощи тега <br>. Если вместо тегов вы будете использовать тег перевода строки <br>, то текст будет "переброшен" на следующую строку с того места, где расположен тег <br>.

Тег
 относится к тегам, не имеющим открывающей и закрывающей части. В него ничего не может быть помещено, поэтому я пишу такие теги следующим образом:
 (т. е. сразу и открывающий и закрывающий тег, косая черта не до, а после названия).

При использовании тега перевода строки текст будет выглядеть таким образом, как представлено на рис. 5.3.

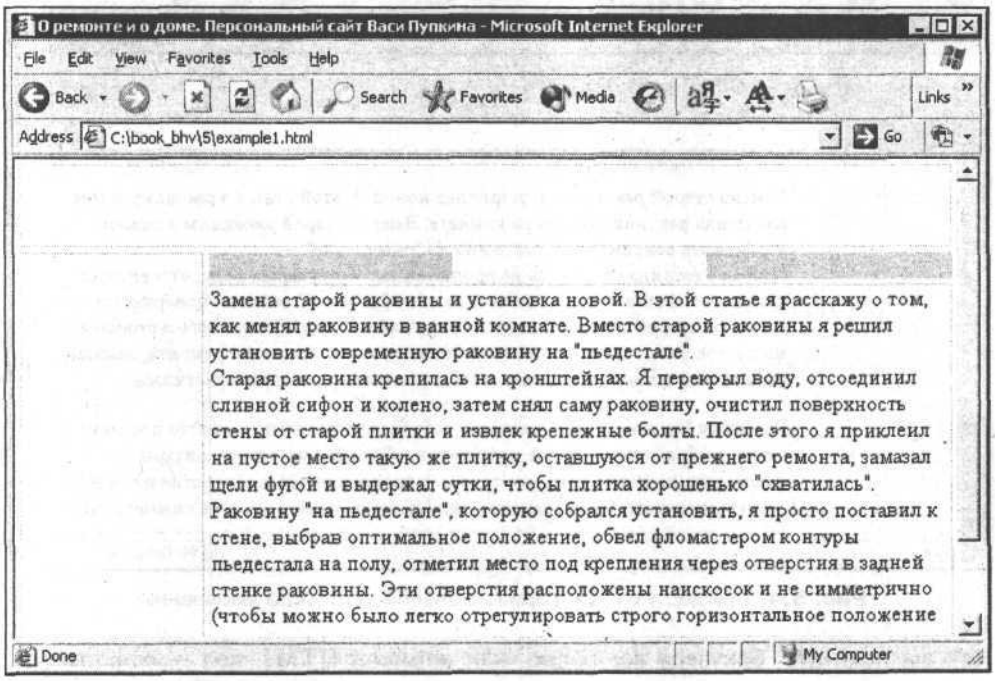


Рис. 5.3. Текст, разделенный на абзацы при помощи тега

Чтобы добиться такого же внешнего вида страницы, как в случае использования тега <p>, необходимо просто продублировать
 (т. е. сделать двойной перевод строки):

```
<br/><br/>.
```

Кроме описанных способов, существуют еще специальные разделители, которыми можно отделять части текста друг от друга. Например, тег <hr/> (горизонтальная черта, имеющая вид тонкой линии с небольшими отступами до и после себя).

Уже известный вам атрибут `width` (ширина) может использоваться и для тега <hr/> (т. е. можно делать горизонтальную черту на половину ширины, на всю ширину, на 10 % и т. д.). Например, запись `<hr color="blue" size="1" width="80%" />` означает, что горизонтальная черта должна быть

синего цвета, толщиной в 1 пиксел и шириной 80 % от того элемента, в котором она находится. На рис. 5.4 представлен пример текста с горизонтальной чертой по умолчанию.

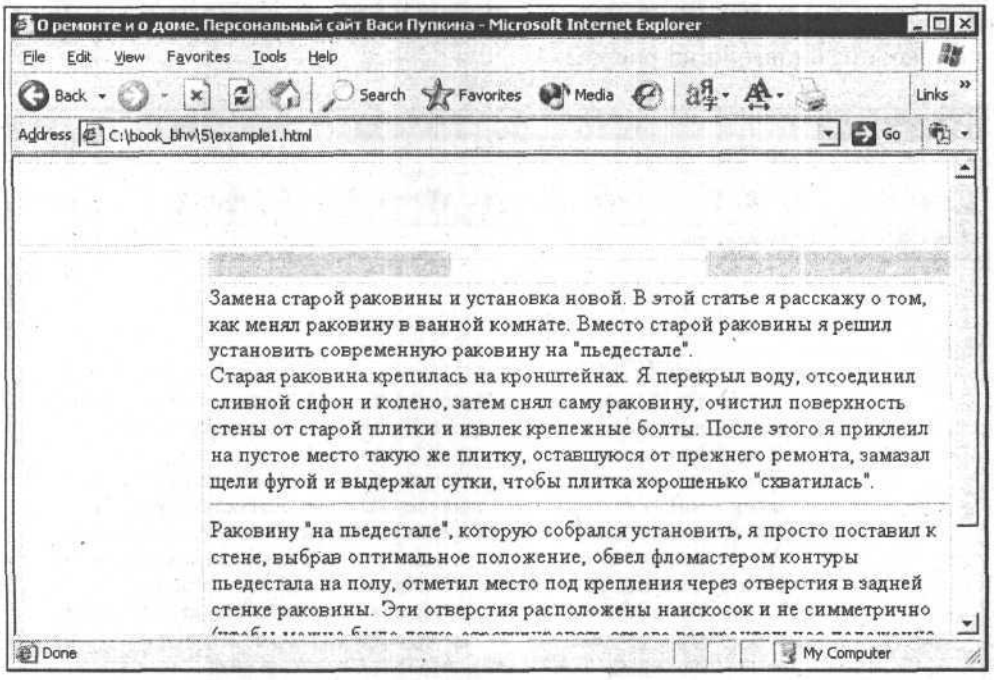


Рис. 5.4. Пример текста с горизонтальной чертой по умолчанию

Как вы помните, браузеру все равно, как написан HTML-код — отформатирован ли он красиво или представлен одной длинной строкой. Поэтому и в том и другом случае текст отобразится одинаково. Как же быть, если необходимо сохранить форматирование и внешний вид текста?

Для этого придуман один очень интересный тег — `<pre>`, который дает инструкцию браузеру отобразить текст, находящийся внутри этого тега, сохранив форматирование, наложенное на него ранее. Например, если вы поместите в HTML-код фрагмент, представленный в листинге 5.2, то получите нужную часть текста в неизменном виде.

Листинг 5.2

```
Текст должен быть отформатирован так:
<pre>
так
    как же,
```

```
        как работает тег PRE?
      Просто — сохраняет форматирование
Текста.'
</pre>
```

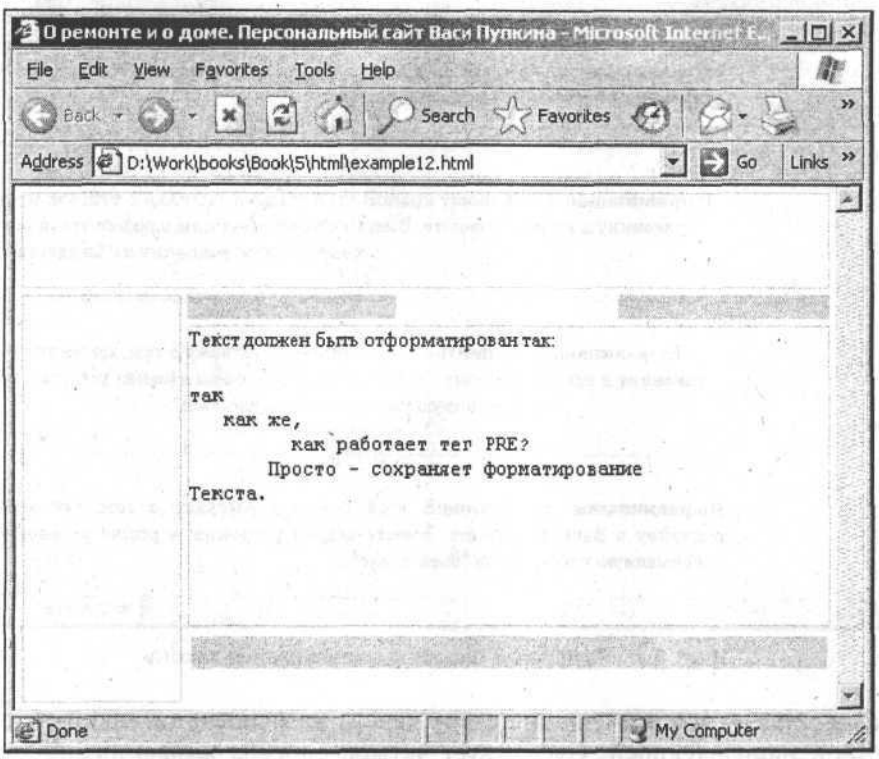


Рис. 5.5. Результат работы тега PRE

Результат, который выдаст браузер, представлен на рис. 5.5.

Помимо указанных, существует еще несколько способов разбиения текста на абзацы (например, можно использовать ячейки таблицы, можно прибегнуть к помощи так называемого контейнера `<div>`, можно найти еще какой-нибудь хитрый способ), но мне вполне хватает рассмотренных, и я не чувствую себя ущемленным в возможностях отделения абзацев текста друг от друга.

Как правильно форматировать текст

Следующим шагом в создании текста для web-страниц будет форматирование. Под словом форматирование понимают и расположение текста определенным образом при помощи отступов и переносов строк (например, форматирование кода), и выравнивание текста.

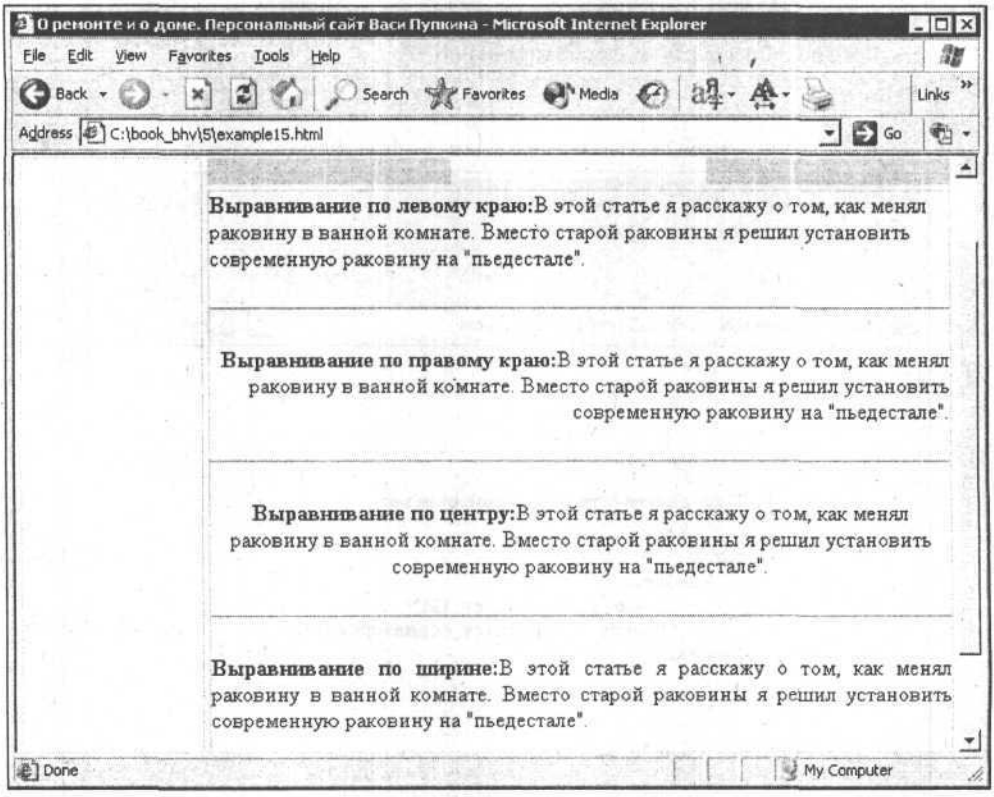


Рис. 5.6. Различные способы выравнивания текста

Выравнивание — это размещение строк текста на основании какой-то определенной направляющей. Существует четыре способа выравнивания текста (рис. 5.6):

- по левому краю (применяется по умолчанию);
- по правому;
- по центру;
- по ширине.

Вообще-то считается, что человеку проще читать текст, выровненный по левому краю. Но с точки зрения дизайна мне кажется более выгодным выравнивание по ширине, хотя при определенных размерах окна браузера и сравнительно большой длине слов будут получаться пустые промежутки между словами, что уж совсем не эстетично. Впрочем, сколько людей, столько и мнений.

Что касается остальных способов выравнивания, там все просто — уже стало привычным, что по правому краю выравнивают эпиграфы, а по центру — заголовки. Для основного текста эти способы используются крайне редко.

Как же задается выравнивание? В листинге 5.3 представлен пример кода, который описывает предыдущий пример с различными способами выравнивания.

Листинг 5.3

```
<!-- Начало основного содержимого страницы-->
  <p align="left"><b>Выравнивание по левому краю:</b></p>
   <hr />
  <p align="right"><b>Выравнивание по правому краю:</b></p>
   <hr />
  <p align="center"><b>Выравнивание по центру:</b></p>
   <hr />
  <p align="justify"><b>Выравнивание по ширине:</b></p>
  <hr />
<!-- Конец  основного содержимого страницы -->
```

То есть выравнивание можно задать при помощи атрибута `align`, используемого с любым тегом, указывающим на абзац, таблицу, ячейку таблицы. Возможные значения этого атрибута:

- `left` — по левому краю;
- `right` — по правому краю;
- `center` — по центру;
- `justify` — по ширине.

На сайте необходимо придерживаться единого стиля во всем — начиная от дизайна страниц и заканчивая выравниванием. Необходимо заранее решить для себя — какой способ выравнивания вы будете использовать и в каких случаях. Не очень красиво получится, если на одной странице ваши заголовки будут выровнены по центру, текст по ширине, а уже на другой — все по левому краю.

Украшаем текст. Шрифт, начертание, размер

Основное содержимое сайта, как правило, представлено текстом. А внешний вид текста, его читаемость и стилистическое оформление определяется шрифтом, которым этот текст написан.

Все шрифты имеют названия и отличаются внешним видом букв. На рис. 5.7 представлен набор достаточно часто используемых шрифтов.

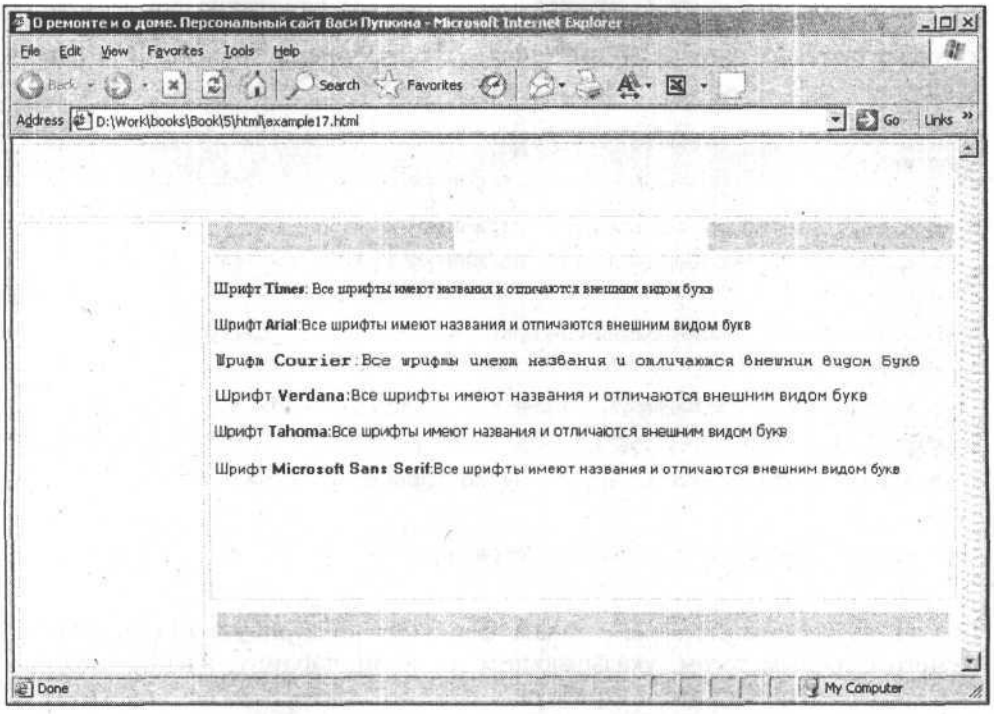


Рис. 5.7. Пример внешнего вида различных шрифтов

Параметры шрифта задаются при помощи тега `<font>` и его атрибутов. Например, если нам нужно написать какой-то текст шрифтом Arial, крупными, синими, подчеркнутыми символами, то выглядеть это будет примерно так, как представлено в листинге 5.4 (на рис. 5.8 результат просмотра браузером).

Листинг 5.4

```
<p>Дальше будет находиться текст,
<font face="Arial, Times, Verdana"
      color="blue"
      size="+3"
      style="text-decoration: underline">
написанный шрифтом Arial, крупными, синими, подчеркнутыми буквами.
</font>
</p>
```

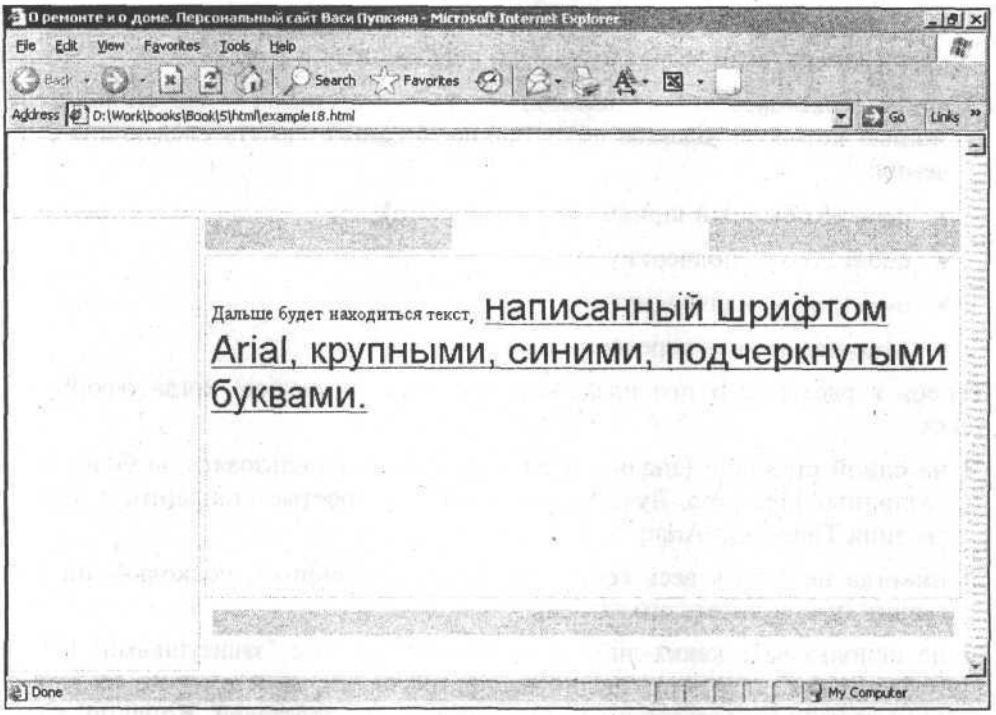


Рис. 5.8. Внешний вид текста, оформленного специальным образом при помощи тега `<font>`

Рассмотрим подробнее отдельные составляющие текста, представленного в листинге 5.4.

- `face="Arial, Times, Verdana"`. Эта запись указывает на то, каким именно шрифтом будет написан текст. Как вы заметили, упомянут был шрифт Arial, но почему-то указаны три разновидности шрифта. Это сделано для того, чтобы избежать проблем с загрузкой какого-то из шрифтов, ведь нет гарантии, что на компьютере пользователя обязательно будут присутствовать все шрифты, которые вы указываете, особенно если они не являются основными. В таком случае вместо шрифта Arial, указанного первым, будет показан шрифт Times, а уж если и его не окажется (что еще менее вероятно), то произойдет попытка подключить шрифт Verdana;
- `color="blue"`. С этой записью, я думаю, вам все понятно. Такой текст будет показан синим цветом;
- `size="+3"`. Эта запись определяет размер шрифта. Значение атрибута `size="+3"` означает, что такой шрифт должен быть на три относительные единицы больше, чем так называемый базовый шрифт (BASEFONT).

Например, если вы зададите размер +1, то текст будет чуть крупнее среднего размера, если зададите +2 — то еще крупнее и т. д.;

- style="text-decoration: underline". Text-decoration — это стиль, с помощью которого задается оформление, он может иметь следующие значения:
  - none — обычный шрифт (без украшений);
  - underline — подчеркнутый;
  - overline — надчеркнутый;
  - line-through — перечеркнутый.

Теперь я расскажу о правилах, которых придерживаюсь, когда оформляю текст:

- на одной странице (значит, и на всем сайте) использовать не более трех различных шрифтов. Лучше всего, если это простые стандартные шрифты типа Times или Arial;
- никогда не делать весь текст полужирным шрифтом, поскольку он выглядит хуже и читать его тяжелее;
- не использовать каких-либо вычурных шрифтов с "завитушками" и "загогулинками", т. к. они трудно воспринимаются, да к тому же их может не оказаться на сервере и на компьютерах пользователей. Конечно, в исключительных случаях такие особые шрифты можно подключать, но тогда надо вместе с сайтом сбрасывать на сервер и специальный файл для шрифта, а еще указать в HTML-коде необходимость загрузки этого шрифта. Я лично, если уж так необходимо сделать надпись на сайте каким-то особенным шрифтом, чтобы создать определенный антураж (например, стиль средневековья), использую картинку с надписью;
- никогда не делать текст на сайте подчеркнутым (если это не гиперссылка). Большая часть пользователей Интернета привыкла к тому, что гиперссылки по умолчанию подчеркнутые, а поэтому будут справедливо возмущаться, пытаясь "кликнуть" мышью на подчеркнутом тексте, и не увидев результата.

Особые элементы текста. Заголовки, списки, спецсимволы

Заголовки оформляются при помощи тегов <h1>, <h2> и т. д. Эти теги предназначены для создания заголовков разного уровня (<h1> — самый крупный, <h2> — чуть меньше, самый маленький — заголовок 6-го уровня <h6>). На рис. 5.9 представлено изображение заголовков на web-странице.

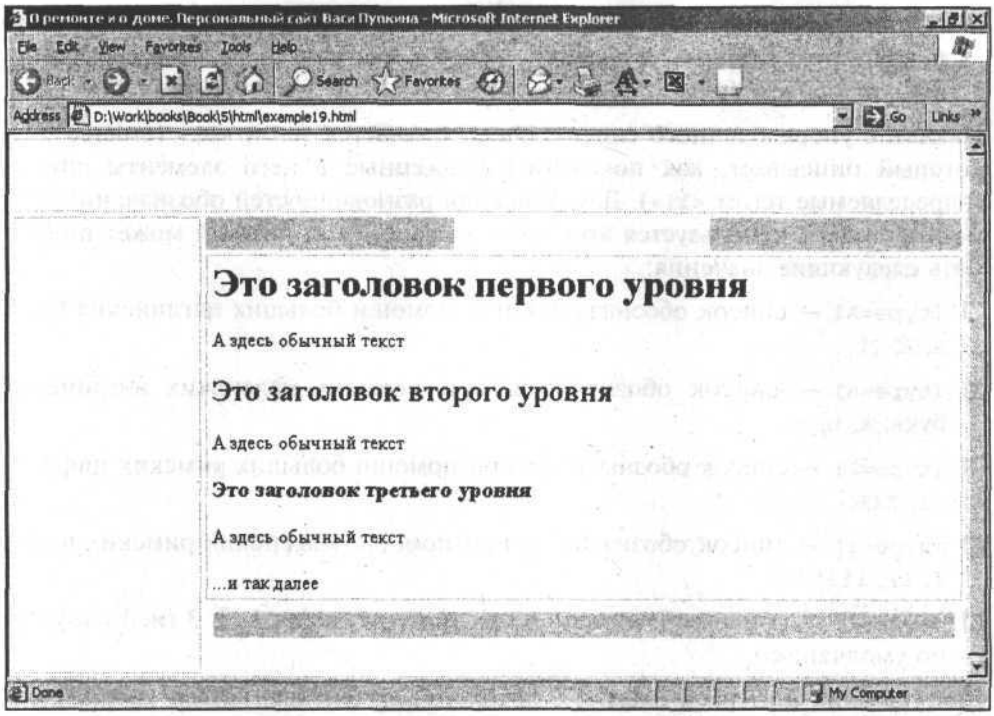


Рис. 5.9. Заголовки на web-странице.

Некоторые разработчики сайтов игнорируют заголовки (т. е. не то, чтобы совсем игнорируют, просто вместо стандартных тегов они частенько используют крупный шрифт или вообще картинку, а зря). Дело в том, что поисковые серверы, индексируя сайты, собирают информацию для дальнейшего отображения по запросам пользователей. Если тег `<h1>` отсутствует, то в результат поиска попадает содержимое тега `<title>` (заголовка страницы). Но, во-первых, этот заголовок не всегда отражает истинную суть страницы, а во-вторых, про тег `<title>` иногда попросту забывают.

Вернусь к своей статье. Там есть заголовок "Замена старой раковины и установка новой". Чтобы оформить его соответствующим образом, необходимо записать так, как представлено в листинге 5.5.

Листинг 5.5

```
<h1>Замена старой раковины и установка новой</h1>
```

Теперь рассмотрим *списки*. Списки бывают:

- упорядоченные, в которых перед элементом списка указывается какая-либо цифра или буква (например: 1, 2, 3 или A, B, C);

- неупорядоченные, у которых в качестве таких отличительных признаков присутствует небольшой значок в виде кружка или квадратика перед каждым элементом списка.

Начнем с упорядоченного списка. Он обозначается тегом `<ol>` (ordered list), который описывает, как показывать вложенные в него элементы списка (определяемые тегом `<li>`). Для описания разновидностей обозначения элементов списка используется атрибут `type` тега `<ol>`, который может принимать следующие значения:

- `(type=A)` — список обозначается при помощи больших английских букв: `A, B, C`;
- `(type=a)` — список обозначается при помощи маленьких английских букв: `a, b, c`;
- `(type=I)` — список обозначается при помощи больших римских цифр: `I, II, III`;
- `(type=i)` — список обозначается при помощи маленьких римских цифр: `i, ii, iii`;
- `(type=1)` — список обозначается при помощи цифр: 1, 2, 3 (используется по умолчанию).

Для указания номера элемента, с которого должен начинаться упорядоченный список, предназначен атрибут `start` тега `<ol>`. Например, если необходимо, чтобы список начинался с маленькой буквы `b`, то текст должен иметь вид, представленный в листинге 5.6 (результат на рис. 5.10).

Листинг 5.6

```
Вам понадобятся такие инструменты:
<ol type="a" start="2">
  <li>дрель (желательно, чтобы она была с перфоратором); </li>
  <li>сверло с победитовым наконечником;</li>
  <li>крепежные болты с дюбелями (продаются в любом строительном магазине).</li>
</ol>
```

Неупорядоченный список обозначается значками: круг, окружность, квадрат. Эти значки задаются при помощи атрибута `type`:

- `type=disc` (круг);
- `type=circle` (окружность);
- `type=square` (квадрат).

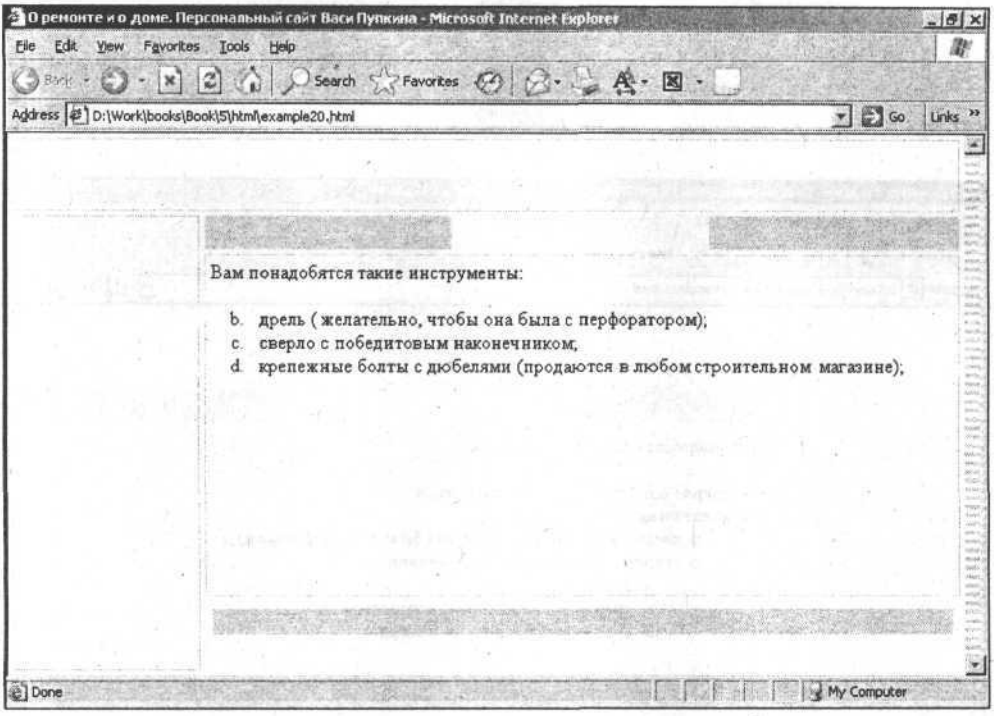


Рис. 5.10. Упорядоченный список, обозначаемый при помощи маленьких английских букв, начинающийся со второго элемента

Если список вложенный (т. е. в нем несколько уровней), то по умолчанию последовательность отображения значков такая: сначала круг, затем окружность, а потом уже квадрат. В листинге 5.7 представлен код текста, а на рис. 5.11 — результат.

Листинг 5.7

```
Вам понадобятся такие инструменты:
<ul>
  <li>сверло с победитовым наконечником</li>
  <li>крепежные болты с дюбелями </li>
  <ul>
    <li>дрель (желательно, чтобы она была с перфоратором)</li>
    <li>сверло с победитовым наконечником</li>
      <ul>
        <li>дрель</li>
        <li>сверло с победитовым наконечником</li>
```

```
        <li>крепежные болты с дюбелями</li>
      </ul>
    </ul>
</ul>
```

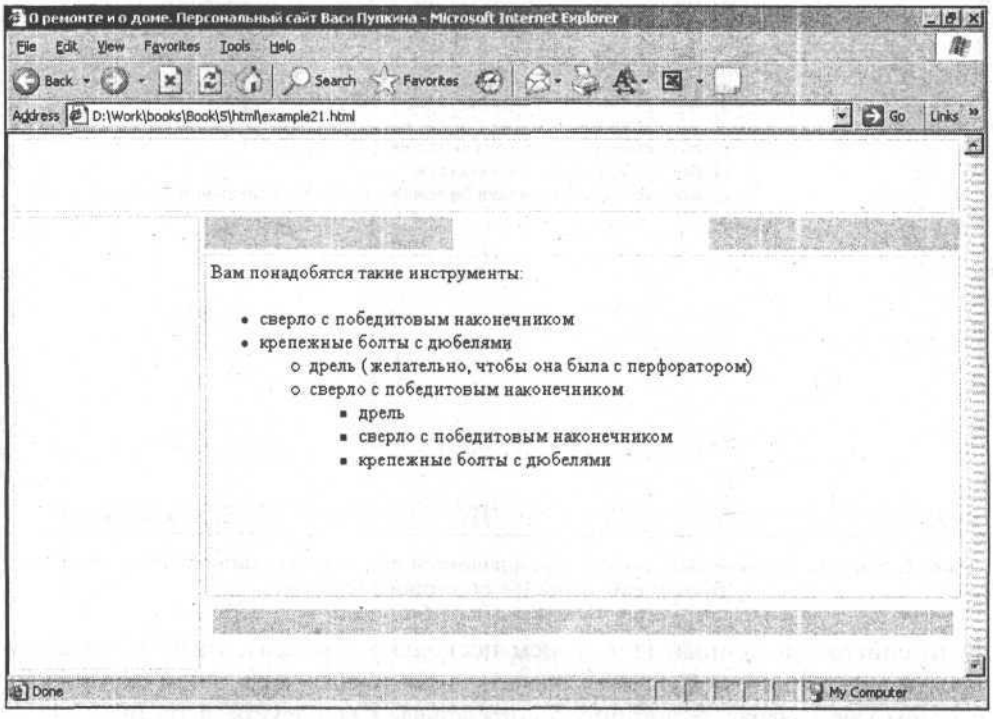


Рис. 5.11. Вложенный список на основе неупорядоченных элементов

Иногда возникает необходимость использовать в тексте специальные символы, которые нельзя ввести с клавиатуры (потому что их нет или невозможно вставить в HTML-код без потери содержимого). Допустим, что на web-странице вам необходимо записать математическое выражение `3<5` или составить описание работы какого-нибудь тега. Если вы вставите в текст символ "<", то он будет воспринят браузером как открывающаяся угловая скобка какого-нибудь тега, и тогда последствия отображения этого текста непредсказуемы — браузер может вполне нормально показать текст, а может и испортить отображение, скрыв часть теста или показав лишнее. Для поддержки возможности вставки в текст web-страниц таких специальных символов были созданы особые варианты их записи. Например, знак авторских прав (copyright) обозначается как `&copy`, а если в тексте страницы браузер обнаружит такую запись, то покажет символ ©. Таких спецсимволов достаточно много, вот основные из них:

- `&amp` — символ & (амперсант);

- &lt — символ < (меньше);
- &gt — символ > (больше);
-   — символ пробела;
- &brvbar — символ горизонтальной черты с разрывом;
- &sect — символ § (параграф);
- &copy — символ © (авторские права);
- &reg — символ ® (зарегистрированная торговая марка);
- &plusmn — символ ± (плюс-минус).

На своем сайте я использую символ авторских прав в нижней части каждой страницы. Пример HTML-кода представлен в листинге 5.8, а результат на рис. 5.12.

Листинг 5.8

```
<!-- Начало нижнего блока-->
  <table width="100%" height="1%">
   <tr>
     <td height="20px" colspan="2" bgcolor="#d4d4d4"> </td>
   </tr>
   <tr>
     <td>&copy; Пупкин Вася 2001-2004 Все права защищены</td>
   </tr>
  </table>
<!-- Конец нижнего блока-->
```

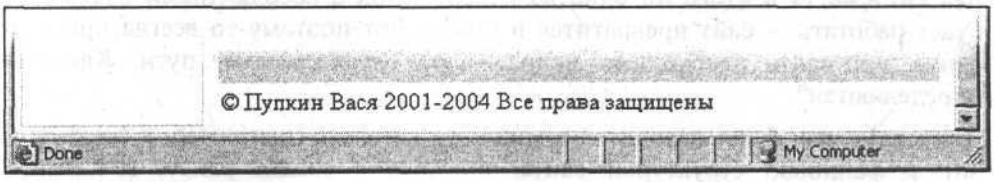


Рис. 5.12. Специальный символ авторских прав в нижней части страницы

Связываем страницы. Создание гиперссылок

Теперь перейдем к самому главному — гиперссылкам. Они связывают страницы между собой. Если с одной страницы вы захотите сделать переход на другую, то этот переход необходимо специальным образом описать. Тег,

создающий гиперссылку, немного отличается от остальных. Дело в том, что гиперссылки бывают разных видов: имеются стандартные гиперссылки, которые служат для связи страниц между собой, а есть еще гиперссылки, работающие внутри одной и той же страницы (так называемые якоря).

Стандартная гиперссылка оформляется при помощи тега `<a>` и его атрибута `href`, значением которого чаще всего является имя файла, к которому вы хотите перейти. Между открывающей и закрывающей частями тега должен находиться сам текст гиперссылки, по которому пользователь будет "кликать" мышью.

Например, если я хочу из своей статьи про установку сантехники (файл change_sanitary.html) сделать переход на следующую статью (например, об облицовке плиткой, файл dalle_cover.html), то я должен в файле change_sanitary.html создать соответствующую запись в том месте, где хочу показать гиперссылку (листинг 5.9).

Листинг 5.9

```
<a href="dalle_cover.html">Следующая статья: «Облицовка плиткой»</a>
```

Здесь особое внимание нужно уделить атрибуту `href`. Его значением является имя файла, к которому необходимо перейти, а точнее не само имя файла, а путь к нему. Если необходимо указать путь к файлу на компьютере, то обычно указывают полный путь (еще говорят — абсолютный), например:

c:\RepairSite\html\Repair\sanitary\dalle_cover.html

Естественно, что этот путь к файлу будет справедлив только для вашего компьютера, потому что мало того, что на другом компьютере не будет таких же каталогов, но там вообще может быть установлена другая операционная система. А в итоге ни одна из гиперссылок с абсолютными путями не будет работать — сайт превратится в ничто. Вот поэтому-то всегда при создании web-сайта необходимо использовать *относительные* пути. Как они определяются?

Допустим, ваш файл находится в каком-то каталоге (например, в соответствии с файловой структурой сайта, которую я создал ранее, в каталоге c:\RepairSite\html\Repair\bath_toilet). Вам нужно установить гиперссылку на файл dalle_cover.html из каталога c:\RepairSite\html\Repair\sanitary. Когда на компьютере нужно перебраться из одного каталога в другой, вы переходите на уровень выше, выбираете нужный вам каталог, а уже внутри него нужный файл. Точно таким же образом записывается и относительный путь. Подъем на уровень выше обозначается двумя точками и косой чертой (например, ..\sanitary). Эта запись говорит о том, что нужно подняться на один уровень вверх и войти в каталог sanitary. Если бы вам надо было из каталога

bath_toilet добраться до каталога c:\RepairSite\html и там вызвать какой-то файл (например, index.html), то относительный путь к файлу index.html выглядел бы так: ..\..\index.html.

На рис. 5.13 показано, каким образом выглядит гиперссылка на рассматриваемом сайте (в нижней части статьи расположена гиперссылка, ведущая к следующей статье).

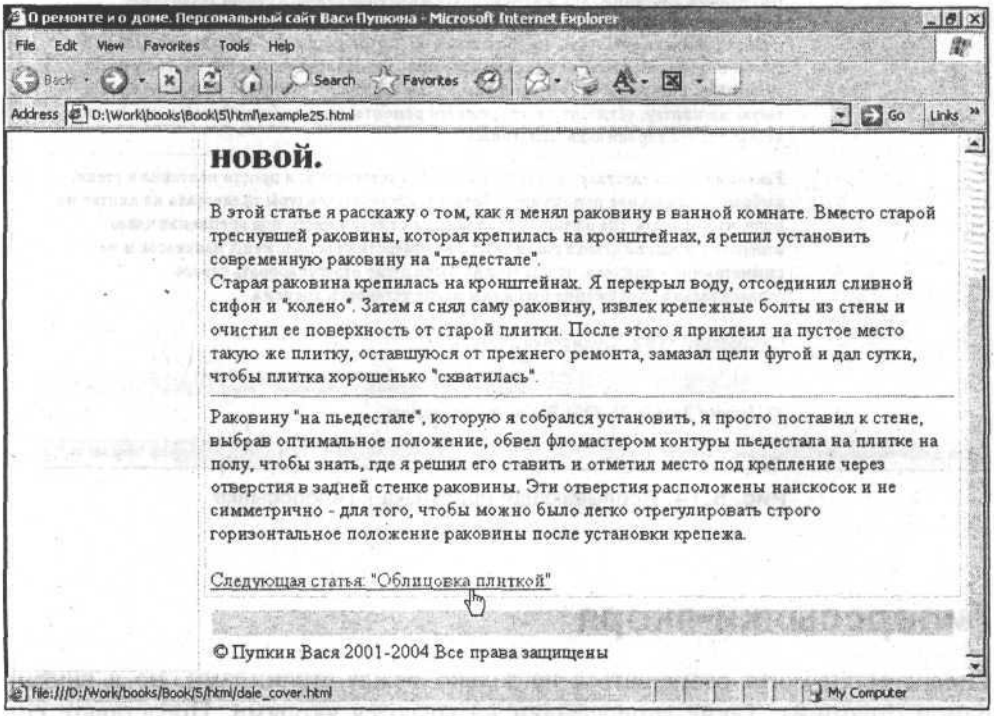


Рис. 5.13. Расположение гиперссылки, ведущей к следующей статье

Еще очень важным моментом в использовании гиперссылок является создание всплывающих подсказок, с помощью которых можно давать пользователю дополнительные пояснения о том, куда приведет его гиперссылка. Всплывающая подсказка представляет собой прямоугольник с текстом, который появляется при наведении курсора на гиперссылку.

Оформляется всплывающая подсказка при помощи атрибута `title` тега `<a>`. Значением атрибута является сам текст подсказки (листинг 5.10, рис. 5.14).

Листинг 5.10

```
<a href="dale_cover.html" title="Щелкните для перехода к следующей
статье"> Следующая статья: "Облицовка плиткой"</a>
```

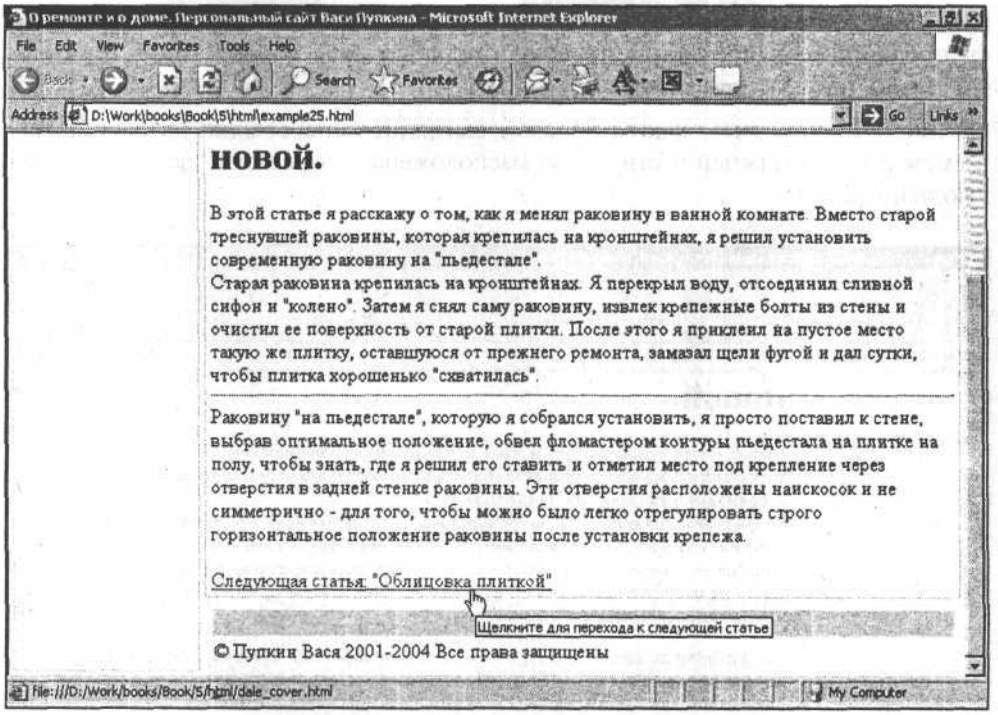


Рис. 5.14. Всплывающая подсказка к гиперссылке

Гиперссылки-якоря

Гиперссылки могут размещаться не только между страницами, но и внутри одной страницы. Такие гиперссылки называются якорями. Представьте себе, что у вас на странице располагается очень большая статья. Чтобы пользователям было удобно перемещаться по содержимому статьи можно прибегнуть к помощи якорей. Когда посетитель нажимает на нужную гиперссылку-якорь, браузер сам показывает на странице то место, где установлен этот якорь. Обычно в конце длинного текста или после каждого раздела располагают такие же гиперссылки-якоря, но уже с существующим по умолчанию якорем в начале страницы. Якоря работают только в том случае, если у страницы есть полоса прокрутки, иначе при нажатии на гиперссылку-якорь ничего не произойдет. В качестве якоря используют атрибут `name="имя якоря"` тега `<a>`.

В качестве значения атрибута `href` другого тега `<a>` используют запись типа: `href="#bathroom"`, где после имени файла страницы ставят символ # (решетка), а после него указывают имя якоря (bathroom).

В случае использования якоря для возврата в начало страницы достаточно просто записать саму гиперссылку на текущую страницу, а после нее написать: `#top`. Если же необходимо перейти на другую страницу и там "стать на якорь", то в атрибуте `href` нужно еще указать имя файла (например, `href="index.html#bathroom`). В листинге 5.11 приведен фрагмент файла index.html, в котором используются гиперссылки-якоря и ссылка "наверх".

Листинг 5.11

```
<a href="#bath_repair">Ремонт в ванной</a><br/>
<a href="house.html#kitchen_repair">Ремонт на кухне</a>
<pre>
Возможны различные варианты того, как...
...
...
<a name="bath_repair">
  <h2>Ремонт в ванной</h2>
</a>
Для того чтобы исправить некоторые...
...
...
<a href="#top">Наверх</a>
...
```

Гиперссылка "Отправить почту"

Помимо описанных видов гиперссылок, существует еще один вид, который используют для того, чтобы посетитель мог при желании отправить послание по электронной почте автору или администратору сайта. Это так называемая mailto-гиперссылка. Когда посетитель "кликает" на ней мышью, то вызывается открытие почтового клиента (программы для отправки и получения электронной почты). В результате нажатия на такую гиперссылку будет вызван почтовый клиент с подготовленными полями **To:** (кому) и **Suject:** (тема письма) (листинг 5.12, рис. 5.15).

Листинг 5.12

```
<A HREF="mailto:user@domain.ru?subject=Мнение о сайте>Написать мне</A>
```

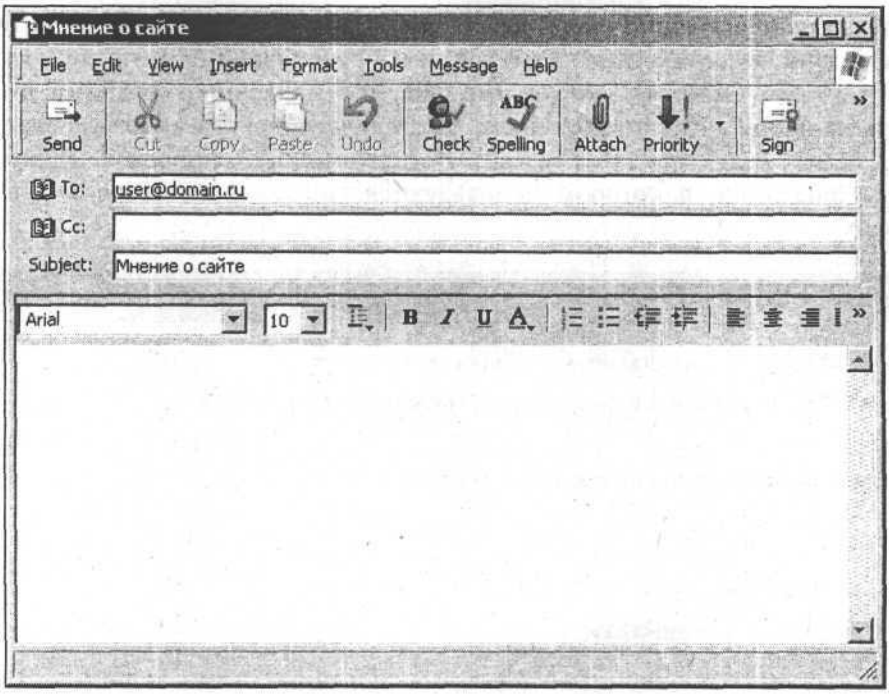


Рис. 5.15. Почтовый клиент, вызванный по ссылке mailto

Способы открытия страницы при нажатии на гиперссылку

Существует несколько способов для открытия гиперссылок.

Первый заключается в том, что страница, вызванная по гиперссылке, открывается в том же окне браузера. Этот способ (установленный в HTML по умолчанию) используется, когда необходимо чтобы пользователь спокойно работал с обычным содержимым сайта — переходил от раздела к разделу, читал статьи, заполнял формы опроса и т. д.

Второй способ — открытие гиперссылки в новом окне — применяется, когда нужно показать посетителю какую-то информацию, не являющуюся основной или важной для содержимого сайта (например, прайс-лист компании, описание содержания книги в интернет-магазине или какая-нибудь картинка в увеличенном виде, т. е. то, что не должно отвлекать пользователя от основного содержания).

Новые окна, открывающиеся при работе с сайтом, называют также "pop-up"-окнами (всплывающие окна). Но дело в том, что их часто используют

для показа посетителям интернет-рекламы. Многих пользователей это раздражает, поэтому они сразу же закрывают новое всплывающее окно, даже не дождавшись его содержания. Некоторые производители браузеров добавляют возможность запрета "pop-up"-окон.

Однако существуют такие ситуации, когда появление всплывающих окон на вашем сайте будет ожидаемым для посетителя:

- если вы сами напишете о том, что ссылка открывается в новом окне (мелким шрифтом рядом со ссылкой или, в крайнем случае, в атрибуте `title`);
- если вы себя "хорошо вели" с посетителем и он вас не заподозрил в причастности к рекламной индустрии Интернета (в этом случае он может и подождать открытия нового окна на вашем сайте);
- если посетитель сам намеренно открывает гиперссылку в новом окне при помощи, например, клавиши <Shift> в Internet Explorer (если нажать на эту клавишу и "кликнуть" на гиперссылке, то появится новое окно). Я, например, так поступаю, когда хочу посмотреть сразу несколько статей, но в то же время не потерять их списка или не ждать, пока они будут загружаться.

Чтобы открыть гиперссылку в новом окне, необходимо указать для нее атрибут `target="_blank"`.

Внешний вид гиперссылок

Изначально гиперссылки выглядят как подчеркнутый текст синего цвета, курсор мыши при наведении на такой фрагмент преобразуется из стрелки в руку. Как правило, гиперссылки по-разному выглядят в зависимости от тех действий, которые вы с ними производили. Обычная гиперссылка имеет состояние "ссылка" (link). Та гиперссылка, на которую вы в данный момент "кликаете", имеет состояние "активная ссылка" (active link). Если вы уже заходили по гиперссылке на страницу, к которой она ведет, то эта гиперссылка принимает состояние "посещенная ссылка" (visited). Внешний вид гиперссылок и их состояний задается в теге `<body>` (листинг 5.13) при помощи атрибутов:

- `link` (ссылка);
- `alink` (активная);
- `vlink` (посещенная).

По умолчанию во всех состояниях гиперссылка подчеркнута и имеет следующие цвета:

- `link=blue (#0000FF)`;
- `alink=red (#FF0000)`;
- `vlink=purple (#800080)`.

При желании можно поменять эти цвета, однако надо учесть то обстоятельство, что все уже привыкли к тому, что гиперссылки подчеркнутые и синие, а если вы решите сделать ссылки, допустим, оранжевыми или при помощи стилей снимите с них подчеркивание, то пользователи не сразу поймут, что это гиперссылки.

Листинг 5.13

```
<BODY link=#0000FF alink=#FF0000 vlink=#800080>
```

Строим модель навигации

Предположим, что на вашем сайте поначалу будет немного статей (примерно по пять или шесть на каждый раздел). Можно было бы упростить себе жизнь и разместить все статьи одного раздела на одной странице, дать сверху их список и установить якорь на каждой статье. Однако необходимо думать и о будущем. Ведь со временем количество статей может увеличиться и тогда страницы, содержащие все статьи раздела будут большими, станут медленно загружаться, пролистывать длинные страницы посетителям будет неудобно, возникнут сложности с распечаткой страниц на принтере (если посетители захотят это сделать). Словом, такой способ построения сайта не совсем удачный.

Чтобы посетителям было удобно работать с содержимым сайта, необходимо тщательно продумать модель навигации, т. е. те способы, посредством которых можно перемещаться между разделами сайта, между статьями, различными страницами, а также и внутри одной страницы. Все это называется *навигацией*. В переводе с английского *navigation* означает управление. Пользователь управляет сайтом. Понятно, что к навигации по сайту относятся такие его элементы, как список разделов, гиперссылки, якоря, иногда кнопки, выпадающие списки и т. д. От того, как устроена модель навигации, зависит комфортность работы посетителя с вашим сайтом.

Сейчас я расскажу о том, как строил модель навигации своего сайта.

Я сделал ссылку на главную страницу сайта слева вверху. Этот прием уже давно стал стандартным и такая гиперссылка служит маяком для пользователей, которые заблудились на сайте. Обычно для подобной ссылки используют либо логотип компании или сайта, либо какой-то текст (например, название сайта или его адрес в Интернете).

Я сделал главное меню сайта со списком основных разделов в его левой части. Тут столкнулся с первой проблемой. У меня ведь имеется несколько уровней структуры сайта (например, раздел "Ремонт в доме", а в нем раздел "Окна и двери"). Если оставлять в главном меню сайта только разделы верхнего уровня, то нужно будет при выборе одного из них, во-первых, показы-

вать выбранный раздел отличным от других (чтобы посетитель не забыл, куда вошел), а во-вторых, показывать список подразделов на месте основного содержимого страницы — примерно так, как показано на рис. 5.16.

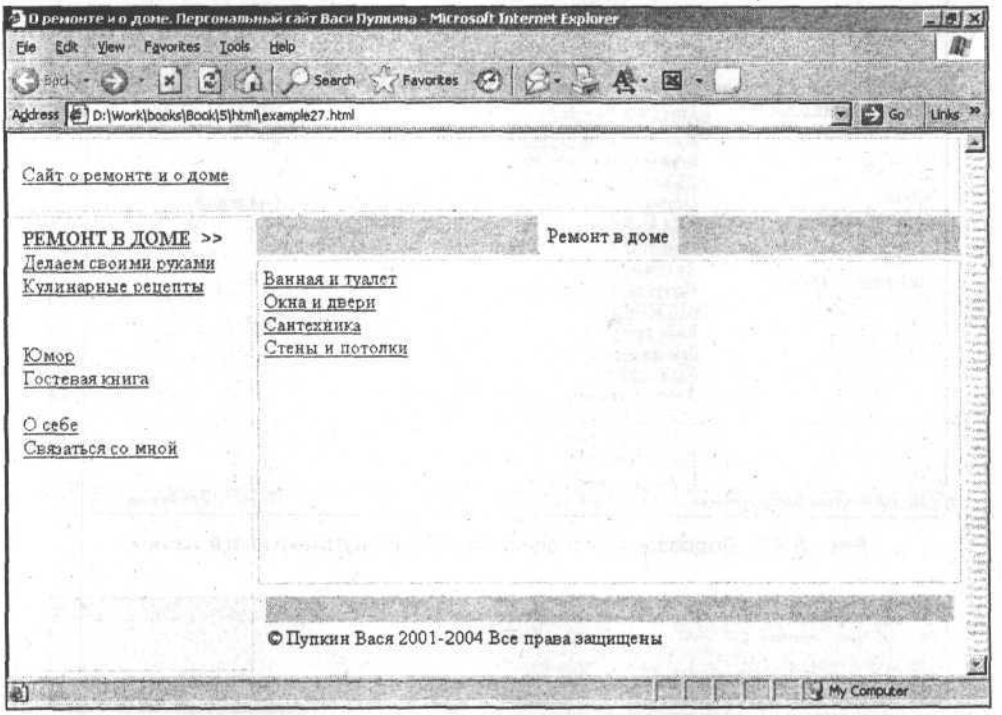


Рис. 5.16. Вариант отображения списка подразделов в поле основного содержимого страницы

Но получается не очень хорошо: во-первых, слишком много пустого места, во-вторых, непонятно, где потом (при переходе к статьям) показывать посетителю название подраздела, в котором он находится. После выбора конкретного подраздела должен появиться список статей, которые в нем содержатся (рис. 5.17).

Этот вариант уже сам по себе не плох. Он напоминает каталоги на поисковых серверах. Посетитель сразу видит всю информацию о статьях и может отыскать нужное слово или фразу средствами браузера в содержимом страницы. Кроме того, становится понятнее смысл названий подразделов, т. к. из списка статей видно, что могло бы находиться в каждом их них. Посмотрим, можно ли найти более удачный вариант.

Такой вариант имеется (рис. 5.18), правда, он потребует немного больших знаний. В левом меню при помощи специального языка JavaScript можно сделать разворачиваемым список подразделов (когда посетитель "кликнет",

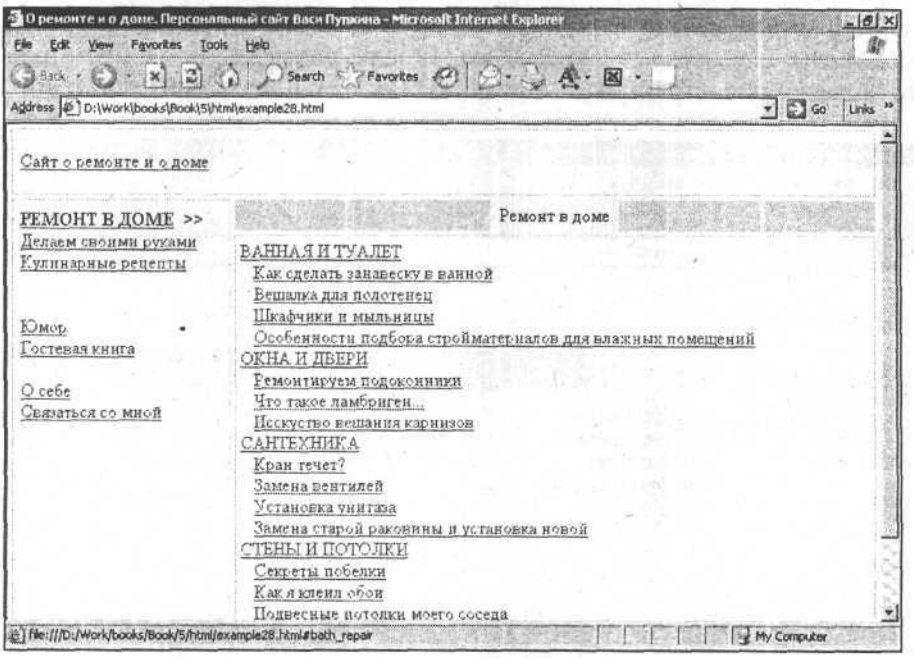


Рис. 5.17. Подразделы со списком статей внутри каждого из них

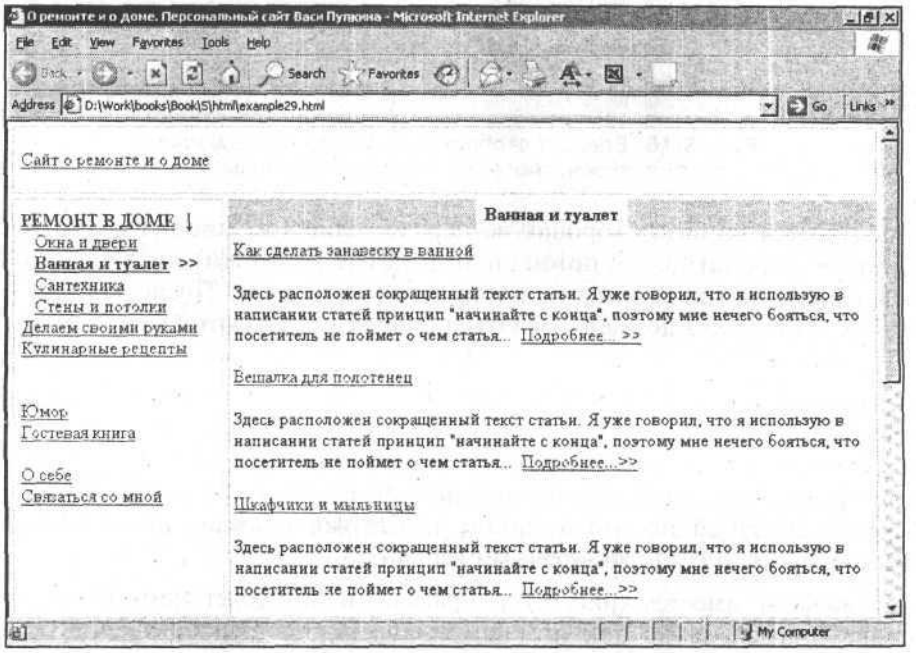


Рис. 5.18. Вариант модели навигации

например, на ссылке "Ремонт в доме", тут же появится под ней список подразделов, но при этом новая страница не будет загружаться). При выборе какого-то конкретного подраздела в части основного содержимого страницы появится список статей, причем не в виде названий, а в виде небольшой части статьи и ссылкой "Подробнее", чтобы посетитель мог понять, о чем статья и при желании посмотреть ее целиком.

Есть еще один элемент, которым я пользуюсь на своем сайте. Он называется "подвал". Такое название обусловлено расположением на странице. Подвал — это то, что находится в самом низу. В подвале чаще всего располагают описание авторских прав, ссылку на главную страницу сайта или ссылку "Отправить почту" с адресом электронной почты, а также список основных разделов сайта в виде нескольких строчек (рис. 5.19).

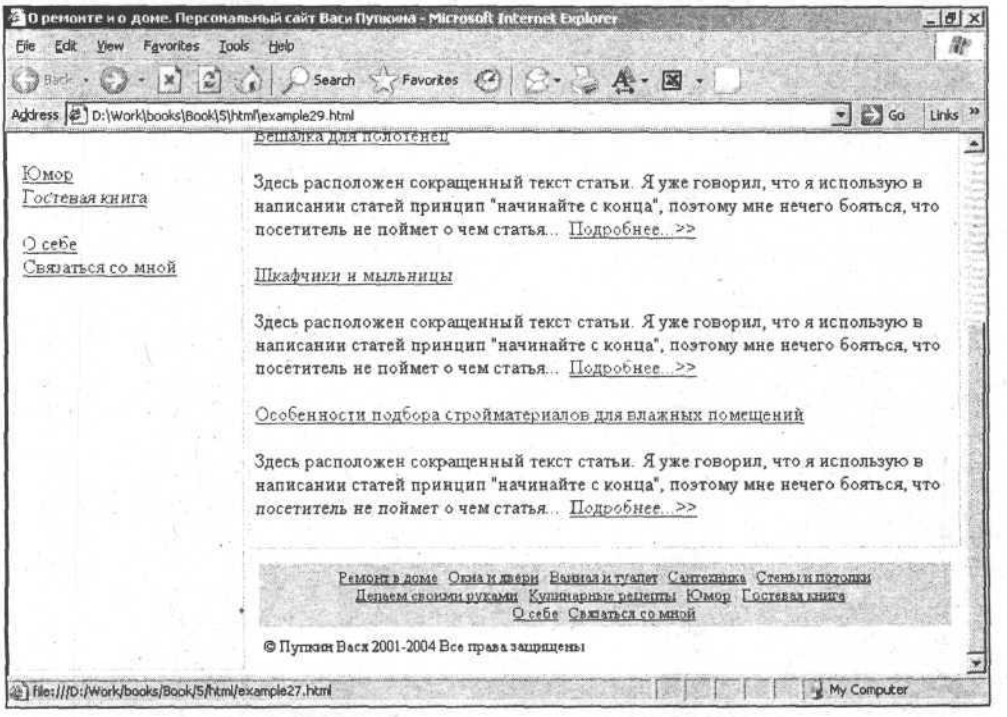
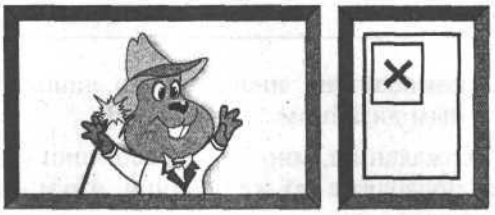


Рис. 5.19. Подвал на сайте

Для чего нужен подвал? Он присутствует (так же, как и ссылка на главную страницу) на каждой странице сайта и позволяет пользователю быстро перейти к нужному разделу, находясь внизу страницы (например, после прочтения большой статьи).



Глава 6

Использование графики на сайте

Графика в Интернете. Выбор наиболее подходящего формата

что толку в книжке, в которой нет ни картинок, ни разговоров...

Льюис Кэрролл. Алиса в Стране чудес

На подавляющем большинстве сайтов имеются графические изображения (это могут быть логотипы, фотографии, элементы дизайна и рекламные блоки). Чаще всего именно графика отличает один сайт от другого по внешнему виду, и такая индивидуальность зависит от дизайнера. Графику используют в самых разнообразных случаях. Например, часто пункты главного меню сайта делают не посредством текста, а при помощи картинок с надписями. К такому приему прибегают, например, если хотят использовать какой-то особый шрифт, которого может не оказаться на компьютере большинства пользователей. Кроме того, картинка с текстом — это уже не текст, поэтому ему не страшна неправильно установленная кодировка. С графическими изображениями можно сделать много визуальных эффектов (мигание, подсветка, объемность и т. д.), что можно просматривать только в специальных редакторах.

Однако все эти "красивости и удобства" обладают недостатками, присущими любому виду графики, расположенной на web-сайте:

- графические изображения часто имеют намного больший размер, чем текст, следовательно, загружаются на компьютер посетителя медленнее;
- существуют некоторые проблемы с загрузкой изображений с сервера, и тогда вместо красивой картинки посетитель увидит прямоугольник с крестиком в левом верхнем углу;
- чтобы на сайте хорошо составить сочетание графических изображений в единое целое, нужно обладать хотя бы минимальными знаниями основ

композиции, иначе можно лишь отпугнуть своих посетителей безвкусным дизайном.

К сожалению, многие разработчики сайтов к вопросу использования графики подходят с тех же позиций, что и Алиса в Стране чудес — что толку, если нет картинок.

В зависимости от интенсивности применения графики, сайты можно разделить на три типа (названия для них я придумал сам):

- *минималистский*, т. е. почти или вообще не использующий графику. Чаще всего это информационные сайты, которые быстро загружаются (т. к. содержат практически один текст), однако это достаточно редкое явление. Посетители видят все, что запланировал автор, но выглядят такие сайты слишком просто (рис. 6.1);

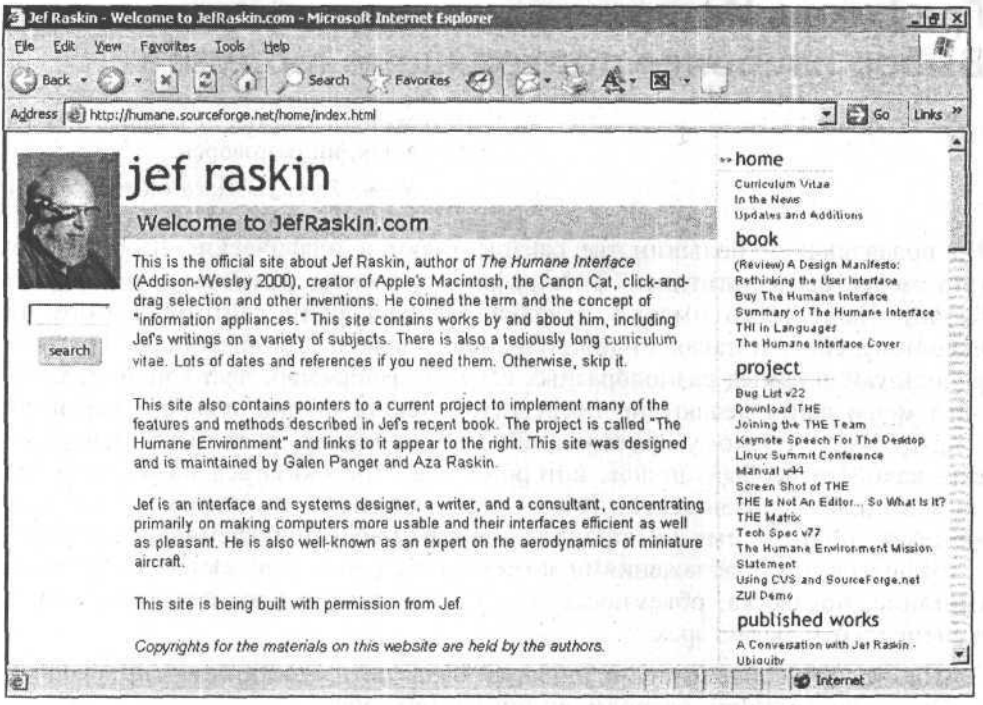


Рис. 6.1. Сайт разработчика графического интерфейса пользователя компьютеров Apple Mackintosh — Джефа Раскина

- *перегруженный графикой*. Такой сайт чаще всего выглядит пестро и едва ли не вульгарно (рис. 6.2). Посетители его постоянно сталкиваются с проблемой загрузки картинок с сервера и их исчезновением, поэтому они чаще всего, не дожидавшись окончательной загрузки, уходят на другой сайт. Если и навигация построена при помощи графики, то она тоже осложняется из-за указанных проблем;

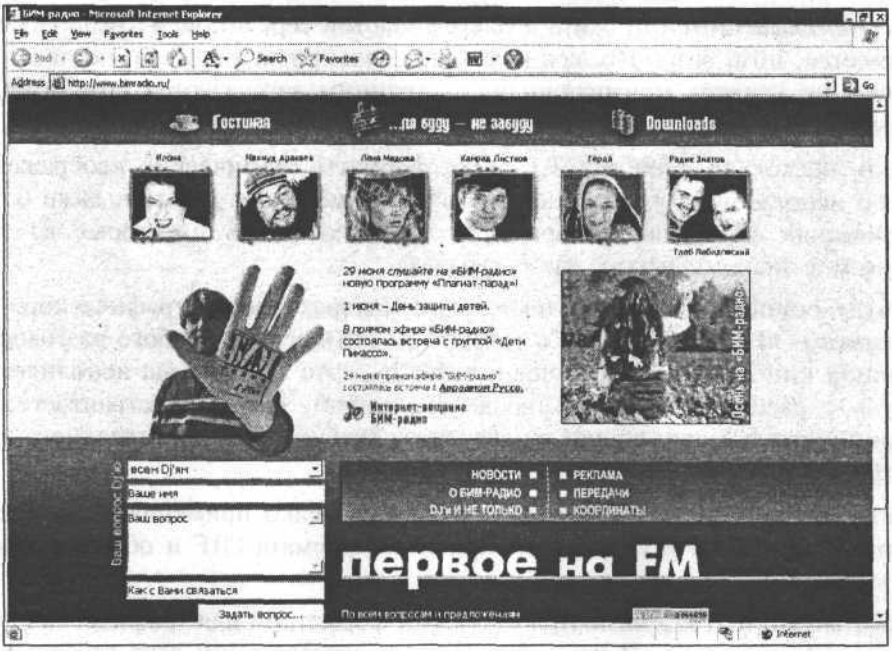


Рис. 6.2. Перегруженный графикой сайт "БИМ-радио"

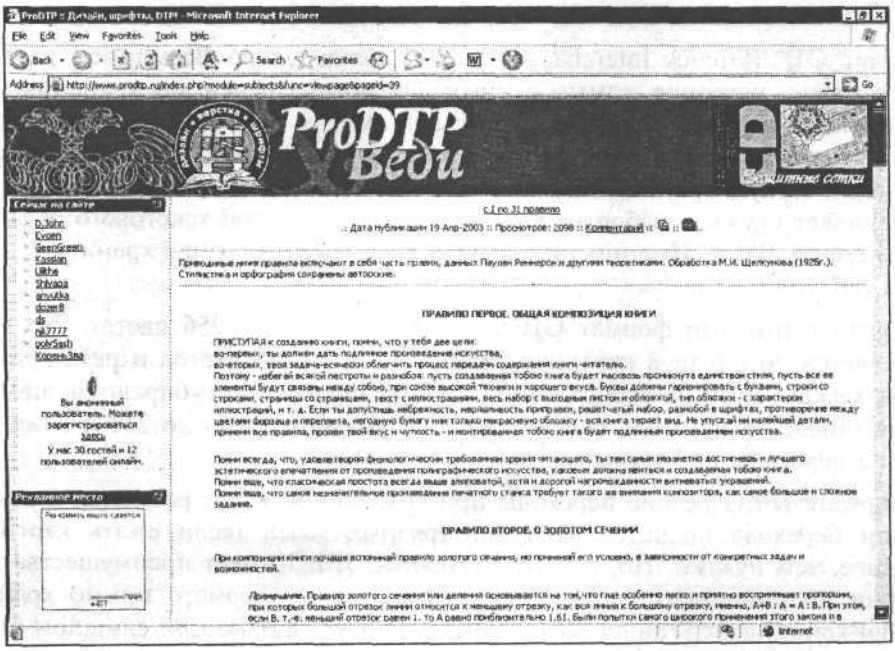


Рис. 6.3. Сайт с интернет-ресурса "Проект ВЕДИ" с усредненным количеством графики

- *усредненный*. Этот тип сайта является золотой серединой — графика здесь имеется, но в разумных количествах и только для удобства, поэтому дизайн не кажется примитивным, а страницы загружаются сравнительно быстро и приятно выглядят (рис. 6.3).

Теперь несколько слов о том, какие форматы графических изображений можно использовать в Интернете. Вообще существуют десятки, даже сотни графических форматов, но браузеры понимают лишь некоторые из них. Лично мне знакомы четыре таких формата.

Не буду останавливаться на технологии интерактивного графического содержимого "Macromedia Flash", потому что это предмет особого разговора и к данной книге не имеет отношения. Flash — это своего рода исполняемый модуль с расширением *.swf (вроде exe-файла), который встраивается на страницу при помощи специального тега и требует наличия отдельного просмотрщика для себя.

Пропускаю я и формат PNG, т. к. его очень редко применяют, поскольку он является как бы ухудшенным вариантом формата GIF и обладает рядом недостатков.

Форматы GIF и JPEG являются главными форматами для графических файлов на web-страницах. Я уже упоминал о существовании двух графических редакторов, предназначенных для векторной и растровой графики (вот именно здесь и используются эти два формата).

Формат GIF (Grafics Interchange Format) появился во Всемирной паутине очень давно, наверное, самым первым. Он предназначен для хранения изображений, близких к векторным по своей сути (т. е. изображениям, в которых нет плавных переходов от цвета к цвету, сами цвета ярко выраженные и имеют четкую границу на своем стыке). Примером графики векторного типа может служить набор клипов, вставляемых внутри текстового редактора Microsoft Word. Именно такого рода изображения лучше хранить в формате GIF (рис. 6.4).

Но дело в том, что формат GIF использует не более 256 цветов. Поэтому получается, что если в картинке значительное число цветов и резкие переходы между частями изображения, то при сохранении убираются лишние цвета (число может быть сокращено с нескольких тысяч до 256 и даже менее), а переходы не "размываются".

В формате JPEG резкие переходы при сохранении будут размыты, а вокруг линии перехода появится ареол из грязных точек (если сжать картинку сильнее, чем нужно). Но, с другой стороны, JPEG имеет преимущества при сжатии полноцветных изображений, ведь в этом формате можно хранить изображения, содержащие более 16 млн цветов. Однако при слишком большой степени сжатия начинает появляться тот самый грязный ареол вокруг отдельных частей (рис. 6.5).

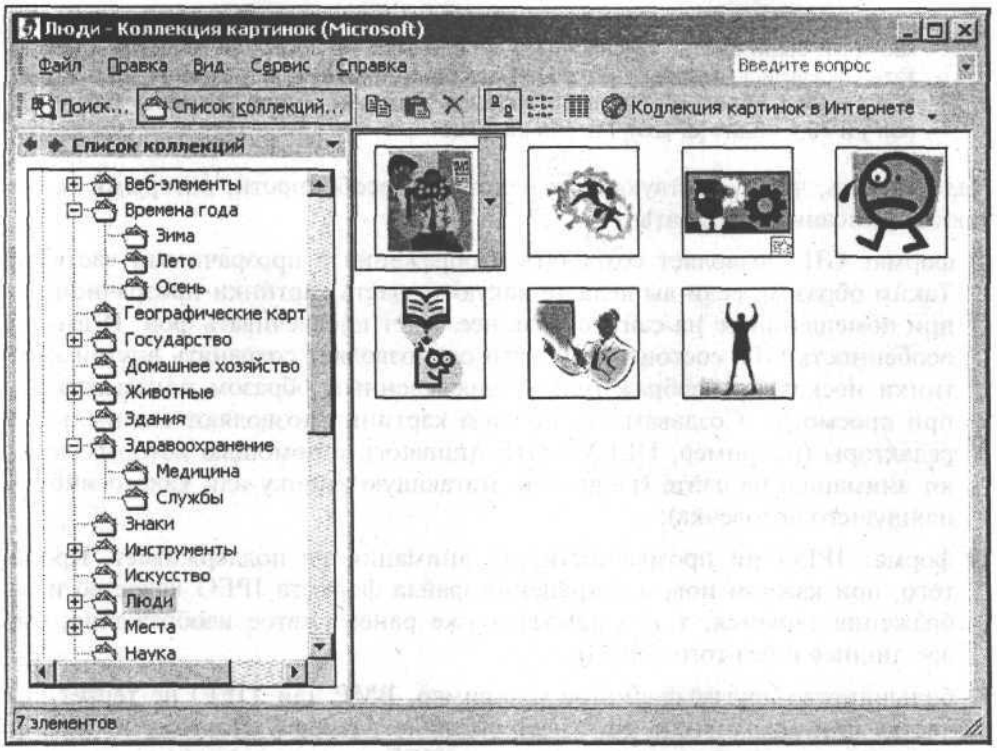


Рис. 6.4. Векторные изображения из набора стандартных клипов для MS Word

Рис. 6.5. Пример нормального формата JPEG и слишком оптимизированного

Примечание

В целях эксперимента я сохранил в буфере обмена изображение экрана (клавишей <Print Screen>) сначала из программы MS Word (т. е. аналог векторного изображения), а затем — с помощью просмотрщика картинок ACDSee (т. е. аналог растрового изображения). Потом эти изображения я вставил в графический

редактор Adobe Photoshop и сохранил их в форматах GIF и JPEG так, чтобы картинки были сжаты максимально, но при этом не потеряли качества. Результаты оказались таковы: векторное изображение имело размеры 28,5 Кбайт (GIF) и 97,4 Кбайт(JPEG). Растровое изображение имело размеры 120 Кбайт (GIF) и 70,1 Кбайт (JPEG). Какова разница!

Надо сказать, что существуют еще некоторые особенности, которыми отличаются описанные форматы:

- формат GIF позволяет сохранять изображения с прозрачными частями. Таким образом, если вы делаете какую-то часть картинки прозрачной, то при помещении ее на сайт, сквозь нее будет просвечивать фон. И вторая особенность GIF состоит в том, что он позволяет сохранять внутри картинки несколько изображений и определенным образом показывать их при просмотре. Создавать такого рода картинки позволяют специальные редакторы (например, ULEAD GIF Animator), с помощью которых делают анимацию на сайте (например, мигающую кнопку или какого-нибудь пляшущего человечка);
- формат JPEG ни прозрачности, ни анимации не поддерживает. Кроме того, при каждом новом сохранении файла формата JPEG качество изображения теряется, т. к. сжимается уже ранее сжатое изображение, где все лишнее и без того убрано;
- большинство других форматов (например, BMP или TIFF) не теряет качества при сохранении, но имеет большие размеры. Поэтому я обычно сохраняю все изображения в формате TIFF, затем редактирую их в графическом редакторе, сохраняю посредством команды **Сохранить как**, чтобы получить JPEG-изображение для сайта. В случае необходимости еще раз отредактировать изображение, я снова открываю TIFF-файл и снова пользуюсь командой **Сохранить как** и т. д.

Сканируем фотографии

Наверняка большинство читателей этой книги знают, что такое сканер, но, тем не менее, я немного поясню. Сканер — это устройство, подключаемое к компьютеру и позволяющее сохранять какие-либо изображения (фотографии, иллюстрации, напечатанный текст). И даже если кто-то захочет отсканировать свою ладонь или любимый брелок, то это тоже возможно. Сканер работает подобно ксероксу, но только делает копию не на другой лист бумаги, а в память компьютера (в графический файл).

Сканеры бывают самых разных размеров и конфигураций. Есть совсем маленькие сканеры, которые просто "протаскивают" через себя лист бумаги или фотографию, есть огромные промышленные сканеры, позволяющие сканировать без участия человека целые стопки документации (с автоматической подачей бумаги), есть обыкновенные, привычные для многих скане-

ры — в виде коробки с закрывающейся крышкой, размером чуть больше обычного листа бумаги. Но почти все они работают по одному принципу.

Чтобы отсканировать изображение, необходимо поместить его внутрь сканера, задать в специальной программе необходимые установки и область сканирования, затем, если необходимо, обработать изображение в графическом редакторе и сохранить результат в виде файла.

Как правило, с большинством сканеров поставляется программа, подобная той, что показана на рис. 6.6, позволяющая работать с ним без всякого графического редактора. Часто такие программы дают возможность работы как опытным пользователям, владеющим тонкостями сканирования, так и новичкам, которые только и знают, что нужно "сначала нажать на эту кнопку, а потом на ту".

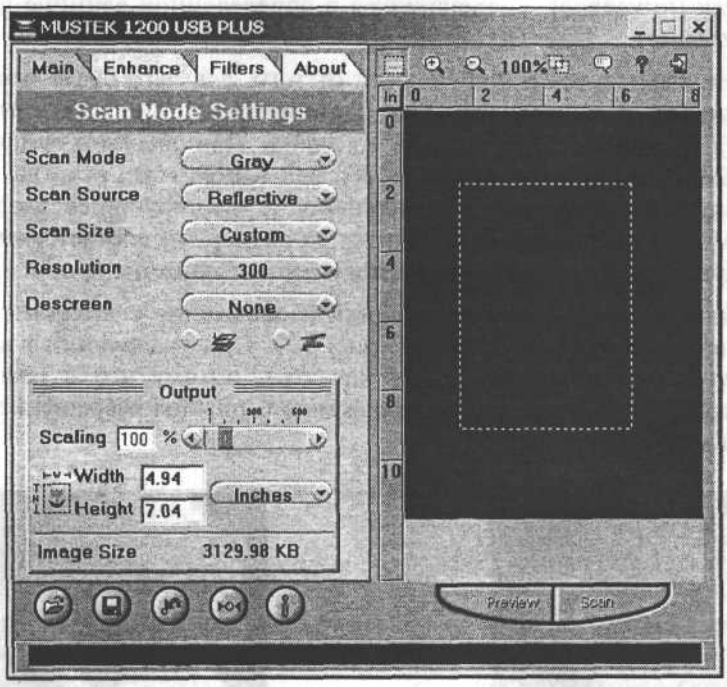


Рис. 6.6. Программа для управления сканированием на сканере Mustek 1200

Для автоматического подбора установок сканер может иметь всего лишь три-четыре кнопки (например, **Черно-белый текст**, **Черно-белое фото**, **Цветное фото**, **Журнал**). Однако чаще всего настройки, сделанные таким образом, не оптимальны, поскольку не могут быть, к примеру, у всех пользователей фотографии одного размера, которые нужно отсканировать с одинаковым качеством. Поэтому желательно, чтобы пользователь владел способами настройки.

Первым параметром, важным при сканировании изображения, является цветность. Чаще всего встречаются такие режимы:

- *grayscale* (или еще его называют black_and_white photo, BW-Photo) — это режим для сканирования черно-белых фотографий. Несмотря на то, что такой режим предназначен (судя по названию) для фотографий, сканирование текста (например, для последующего распознавания) позволяет получать более качественный результат;
- *color* (или, например, RGB, Color-Photo) — режим для сканирования цветных изображений;
- *lineart* (другие варианты: text, BW-Text) — режим для сканирования текста (например, книги или распечатки с принтера).

Вторым параметром является разрешение сканирования (речь идет о количестве точек (пикселов), умещающихся в определенной единице длины, например в дюйме).

Разрешение выбирается в зависимости от того, для чего будет предназначено изображение. Например, если вы сканируете что-либо для сайта, то вам хватит и 72 dpi (dot per inch – точек на дюйм). Для обычной бумаги необходимо 150-200 dpi, для типографии — 300 dpi. Кроме того, если вы сканируете маленькую фотографию, а в дальнейшем планируете ее увеличить, то нужно знать, во сколько раз увеличится ее размер (и, соответственно, потеряется разрешение) по сравнению с полученным на сканере оригиналом.

Чтобы лучше понимать, чем отличаются изображения с высоким и низким разрешением, приведу два примера (рис. 6.7). При увеличении изображение с высоким разрешением не теряет качества, потому что в нем хранится большое количество пикселов. Изображение с низким разрешением "рассыпалось" на мелкие квадраты.

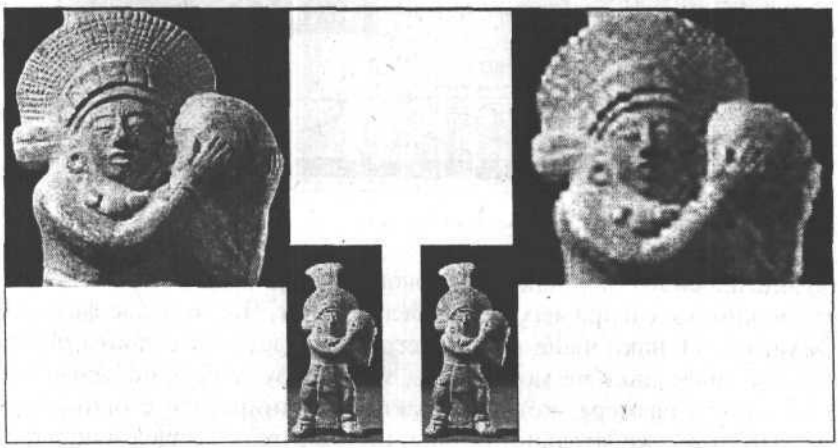


Рис. 6.7. Два изображения: с высоким разрешением (слева) и низким (справа)

Чтобы при сканировании убрать так называемый муар (проблемные места на изображении в виде лишних узоров, грязных или резких пятен и т. п.) пользуются опцией, которая называется *Descreen*. Она предлагает установить тип источника изображения (например, газета (*newspaper*), журнал (*magazin*) и т. п.), после чего (в зависимости от выбранного варианта) данное изображение будет специальным образом подправлено.

Я свои файлы с изображениями после сканирования обрабатываю в редакторе Adobe Photoshop, где необходимо выбрать **File** (Файл) | **Import** (Импорт), а затем — пункт в зависимости от используемого сканера (например, **Mustek 1200 VSB PLUS**) (рис. 6.8).

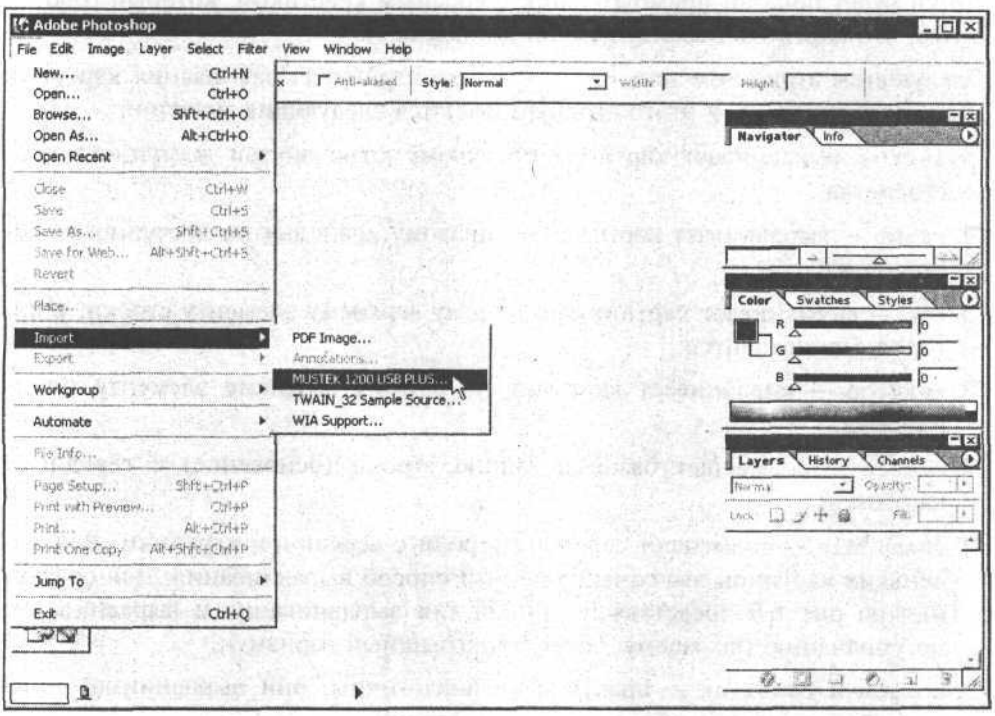


Рис. 6.8. Вызов программы сканирования из графического редактора Adobe Photoshop

Добавляем графику на web-страницу

Чтобы поместить на web-страницу какое-то изображение, необходимо использовать тег `<img>` и его атрибуты. Обращаю внимание на то, что данный тег не имеет закрывающей части. Минимальным набором является сочетание самого тега и атрибута `src`, который указывает путь к графическому файлу в файловой системе (аналогично путям в гиперссылках). Обычно для

того, чтобы не смешивать HTML-файлы и картинки, я в каждом каталоге с web-страницами создаю специальный подкаталог img. Тогда путь к графическим файлам, например, к файлу 14.gif, выглядит так, как представлено в листинге 6.1.

Листинг 6.1

```
<img src="img/14.gif">
```

Если вы ошиблись в имени файла или в написании атрибута, то вместо картинки будет показан прямоугольник с красным крестиком, который говорит о том, что картинка была загружена неудачно.

Следующим атрибутом тега `<img>` является атрибут выравнивания картинки `align` (листинг 6.2). У этого атрибута имеются следующие значения:

- `left` — выравнивает картинку по левому краю внутри доступного пространства;
- `right` — выравнивает картинку по правому краю внутри доступного пространства;
- `top` — выравнивает картинку по самому верхнему элементу строки, в которой она находится;
- `texttop` — выравнивает картинку по самому верхнему элементу текста в строке;
- `middle` — совмещает базовую линию строки (основание) с серединой картинки;
- `absmiddle` — совмещает середину строки с серединой картинки. Для маленьких картинок это самый удачный способ выравнивания. Для сравнения на рис. 6.9 представлен данный тип выравнивания и выравнивание по умолчанию (по-моему, более проигрышный вариант);
- `bottom` и `baseline` — практически аналогичны, они выравнивают нижний край картинки с базовой линией строки.

Листинг 6.2

```
<img src="img/14.gif" align="absmiddle">
```

Важными атрибутами являются `width` (ширина) и `height` (высота), которые играют существенную роль. Дело в том, что браузер, как правило, ничего не знает о размерах картинок, если эти атрибуты не указаны. Поэтому, например, таблица, в которой расположены большие картинки, сначала может

быть отображена в окне браузера как просто узкая полоска. По мере загрузки картинок она расширяется, все содержимое страницы начинает "плясать" и "прыгать", что для пользователя очень неудобно. В случае использования указанных атрибутов (т. е. когда размеры картинки уже заранее известны), браузер загружает страницу, отводя под нее необходимое место, что также позволяет создателю сайта не волноваться о том, что дизайн страницы может измениться, если у посетителя отключена загрузка графики.

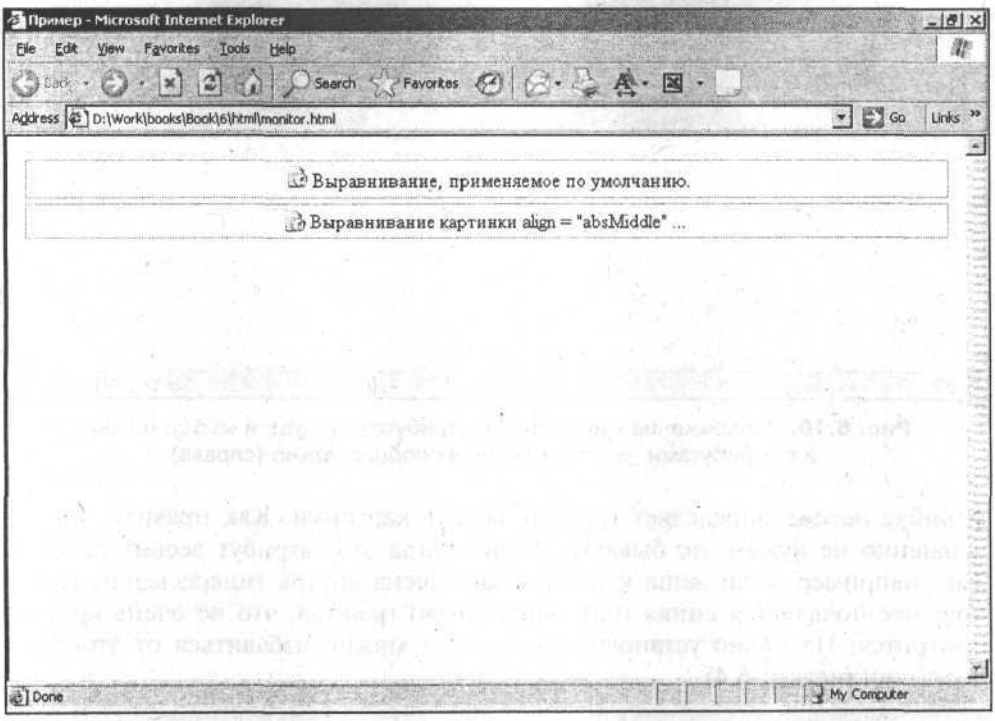


Рис. 6.9. Два способа выравнивания

Кроме того, посредством атрибутов `width` и `height` можно изменять размер отображения картинки (листинг 6.3). Если указать только один из атрибутов, то картинка будет пропорционально "подогнана" под указанный размер (рис. 6.10).

Листинг 6.3

```
<img src="img/14.gif" weight="430" height="200">
```

Рис. 6.10. Изображение картинки без атрибутов `height` и `width` (слева) и с атрибутами, указанными непропорционально (справа)

Атрибут `border` определяет границу вокруг картинки. Как правило, он совершенно не нужен, но бывают случаи, когда этот атрибут весьма полезен. Так, например, если ваша картинка заключена внутрь гиперссылки, то вокруг нее появляется синяя (или фиолетовая) граница, что не очень красиво смотрится. Насильно установив `border="0"` можно избавиться от этого недостатка (листинг 6.4).

Листинг 6.4

```
<img src="img/14.gif" border="0">
```

Атрибуты `vspace` и `hspace` имеют значения в пикселах и используются в случае выравнивания `left` или `right`. Они определяют пустые отступы вокруг картинки (`vspace` (vertical space) определяет отступы сверху и снизу, а `hspace` (horizontal space) — справа и слева) (листинг 6.5, рис. 6.11).

Листинг 6.5

```
<img src="img/14.gif" align="left" hspace="40" vspace="20">
```

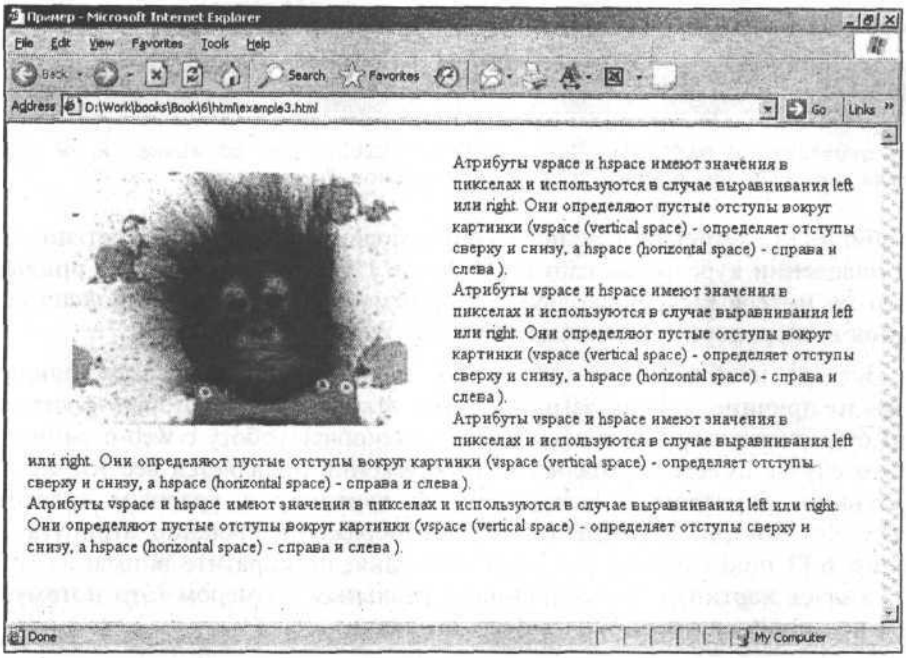


Рис. 6.11. Результат использования атрибутов hspace и vspace

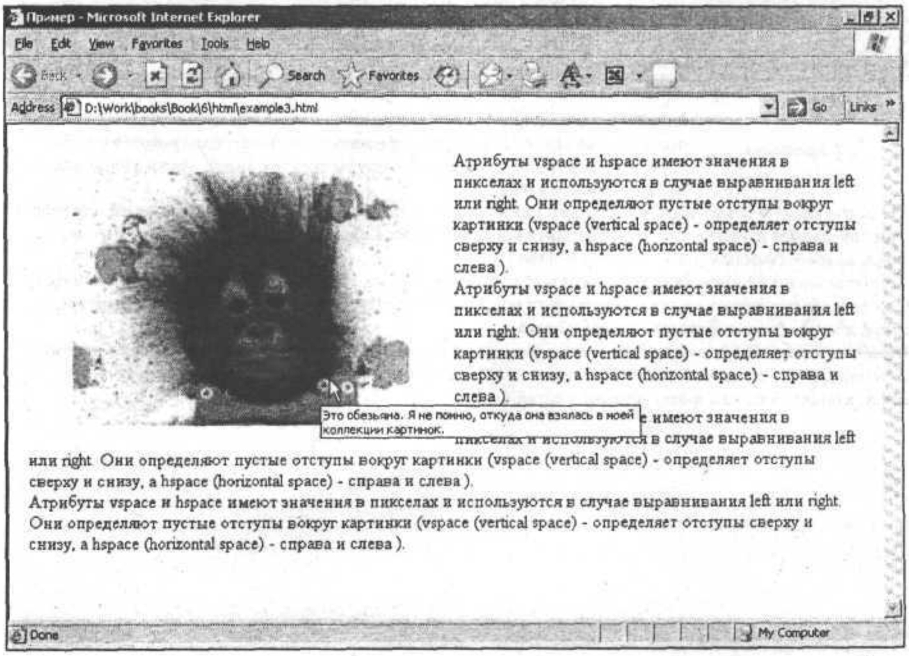


Рис. 6.12. Всплывающая подсказка к картинке

Также очень важными являются атрибуты `title` и `alt` (листинг 6.6).

Листинг 6.6

```
<img src="img/14.gif" alt="Это обезьяна" title="Это обезьяна. Я не помню,
откуда она взялась в моей коллекции картинок.">
```

Атрибут `title` описывает всплывающую подсказку, которая будет показана при наведении курсора на картинку (рис. 6.12). Пользователи уже привыкли к такому интерфейсу, поскольку подобным образом работает большинство иконок в операционной системе Microsoft Windows.

Атрибут `alt` необходим на тот случай, когда картинки не загрузились по какой-то причине или их загрузка вовсе отключена. Некоторые посетители сами отключают загрузку графики, чтобы ускорить работу с web-страницами. В этом случае вместо нормального изображения появляется все тот же прямоугольник размером с незагруженную картинку с красным крестиком в углу, но внутри его области будет отображено значение атрибута `alt`. На рис. 6.13 показана как раз такая ситуация, но обратите внимание на то, как сжалась картинка по сравнению с реальным размером (это потому что у нее не прописаны атрибуты `height` и `width`).

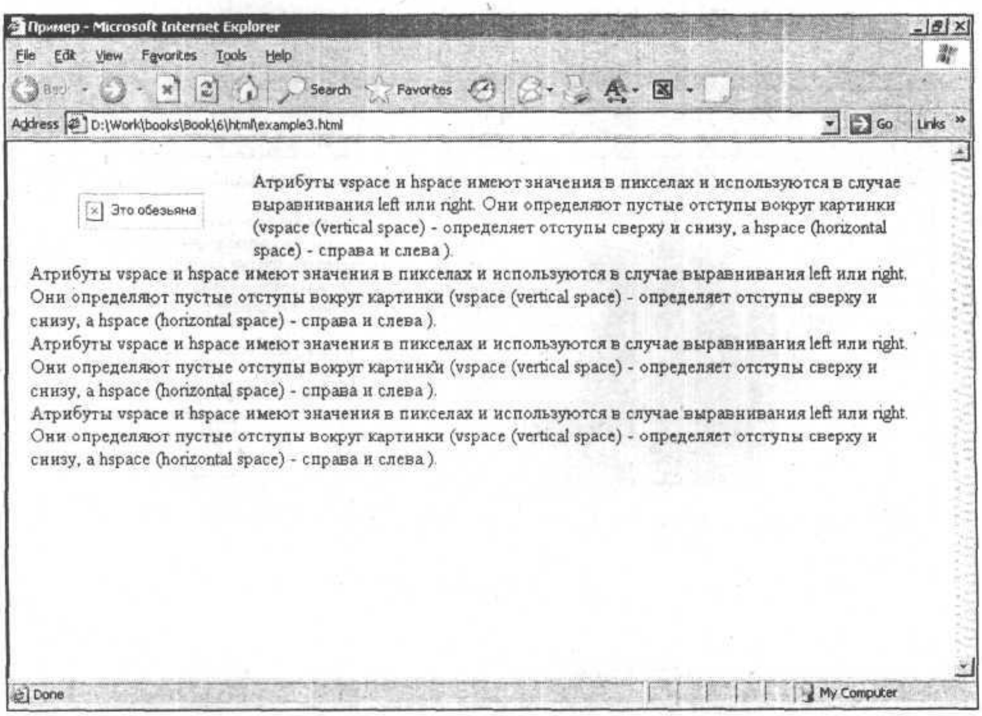


Рис. 6.13. Результат применения атрибутов `alt` и `title`

Создаем фотогалерею

Возможно, у вас имеется много фотографий или просто каких-то изображений, которые хочется выставить на своем сайте. Например, я коллекционирую фотографии из раздела "Юмор". У меня есть дешевый цифровой фотоаппарат, с помощью которого фиксирую всякие несуразности в жизни. Недавно сфотографировал новый магазин, над которым большими буквами красовалась надпись "Мы открылись", а на двери висела табличка "Закрыто". Мне хочется, чтобы такие "фото-приколы" стали достоянием общественности. Вот я и решил создать на своем сайте фотогалерею. На самом деле, чтобы создать ее, вовсе не обязательно разбираться в HTML. На интернет-серверах, которые бесплатно предоставляют место под сайт, часто встречаются специальные системы управления сайтом, предназначенные для новичков (рис. 6.14).

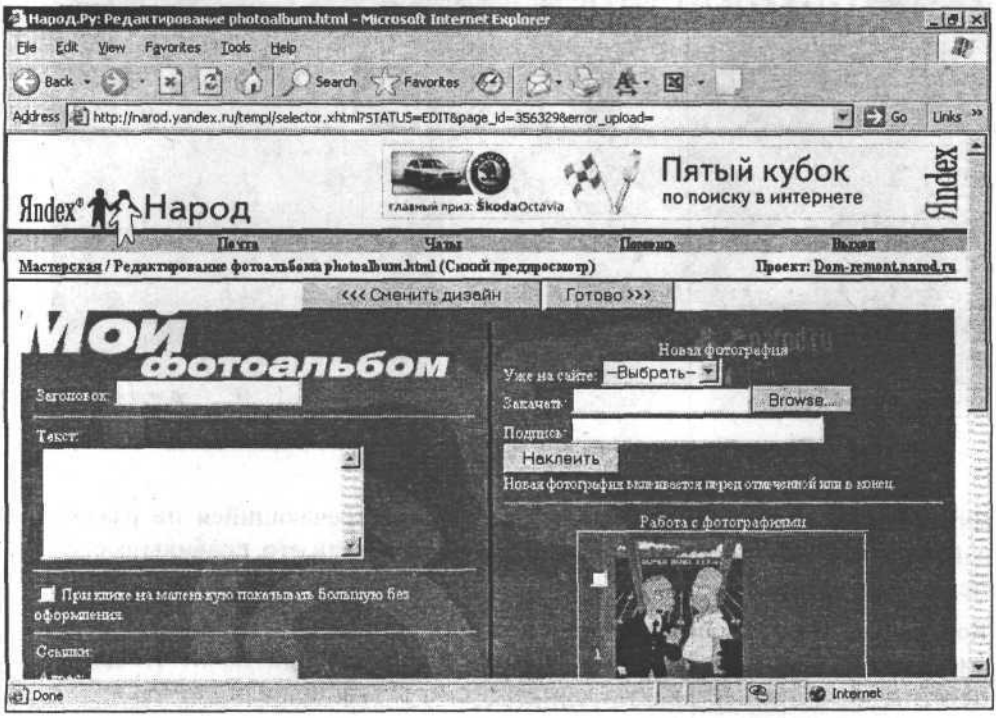


Рис. 6.14. Форма для создания фотоальбома на личном сайте (на сервере **narod.ru**)

Результат создания фотоальбома "чужими руками" показан на рис. 6.15. Но такой подход к созданию собственного сайта или его частей мне никогда не нравился. Во-первых, автор зависит от того, что ему предлагают, он не сможет никаким образом видоизменить свой фотоальбом (фотогалерею).

Во-вторых, если все пользователи, имеющие личные странички на **narod.ru**, станут делать одинаковые фотогалереи, то это будет, по меньшей мере, скучно. В-третьих, очень сомнительно, что предлагаемые разработчиками **narod.ru** (впрочем, как и любыми другими) варианты будут совпадать со стилевым дизайном вашего сайта.

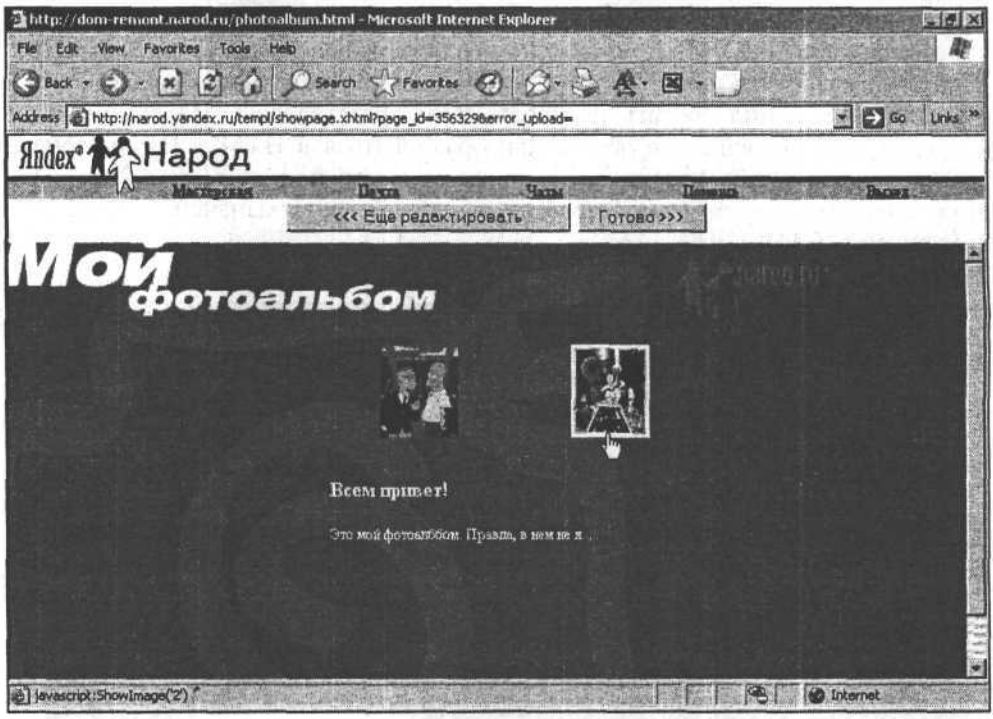


Рис. 6.15. Результат создания фотоальбома на **narod.ru** при помощи Мастера

Фотогалерея — элемент довольно-таки часто встречающийся на различных сайтах и поэтому проблем с тем, чтобы понять, как его правильно сделать, не должно возникать.

Во-первых, очень полезно и удобно, когда имеются уменьшенные копии рисунков или фотографий, находящихся в галерее. При таком отображении посетитель сам решает, какую картинку смотреть в полный размер.

Во-вторых, картинки загружаются медленно, поэтому нежелательно выдавать посетителю сразу большое количество, пусть даже и маленьких картинок. Да к тому же человеку трудно воспринимать весь объем информации. Можно выдавать картинки порциями по 10—15 штук, предоставляя ему право пользования навигацией между этими порциями. Порции обычно обозначают цифрами. Иногда для удобства делают еще и ссылки "Следую-

щая" и "Предыдущая", чтобы пользователю было комфортнее. На рис. 6.16 представлена заготовка страницы с порциями изображений.

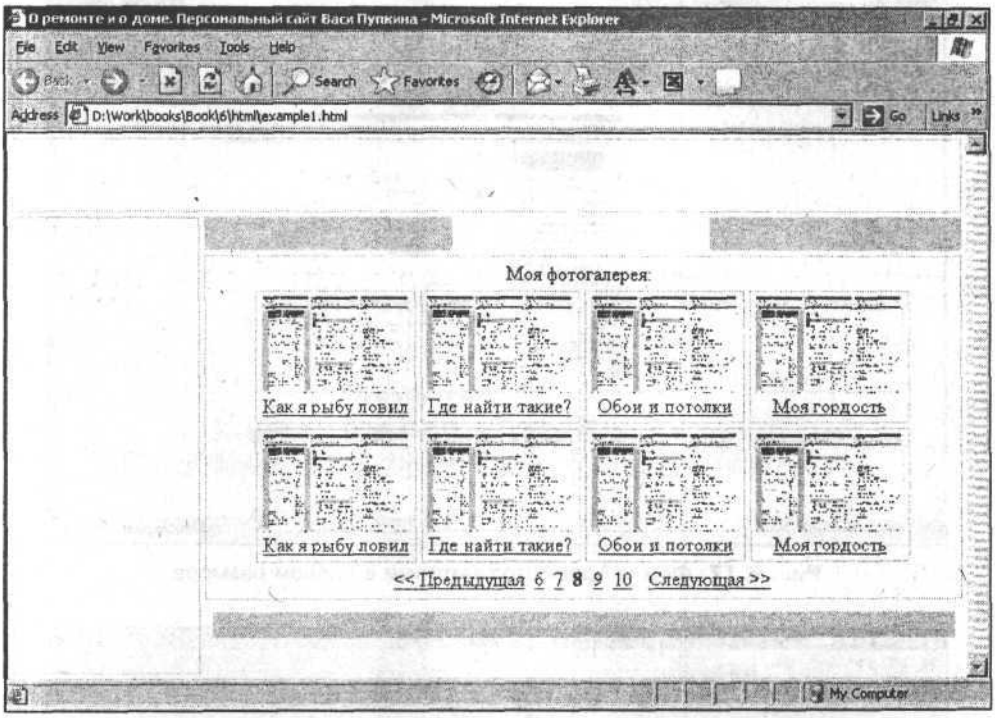


Рис. 6.16. Заготовка страницы для фотоальбома с порциями изображений

Рассмотрим, как устроена эта страница. На ней восемь уменьшенных картинок, под каждой из которых имеется надпись, характеризующая ее. В нижней части страницы расположен элемент навигации по порциям картинок, который содержит ссылки "Предыдущая" и "Следующая", а также номера-гиперссылки нескольких соседних порций. Номер текущей порции (цифра "8") представлен в виде обычного текста (не гиперссылки). При нажатии на ссылку "9" в этом элементе навигации исчезнет ссылка на порцию "6" и справа появится ссылка на порцию "11" (если она имеется). Ссылка "9" будет представлена уже не гиперссылкой, а просто текстом.

И сама картинка, и надпись являются гиперссылкой, при нажатии на которую вызывается просмотр картинки в полном размере. Форма просмотра картинки в полном размере схематично представлена на рис. 6.17.

Теперь посмотрим, как устроена эта страница. Здесь уже одна картинка в полный размер с надписью и тремя гиперссылками: "Предыдущая", "Вернуться к списку" и "Следующая". Понятно, что гиперссылка "Вернуться к списку" зависит от того, к какой порции принадлежит данная картинка.

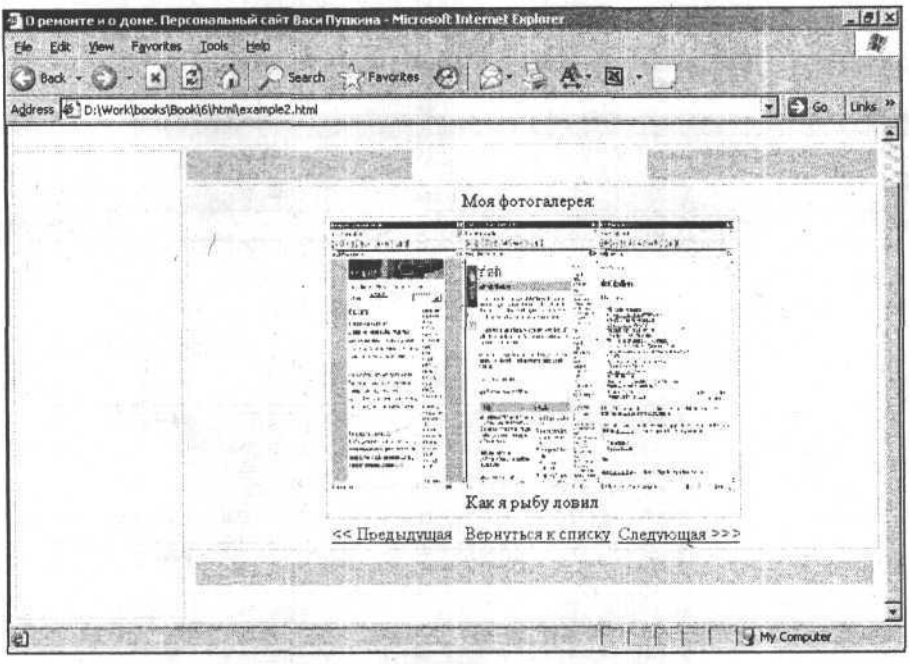


Рис. 6.17. Форма просмотра картинки в полном размере

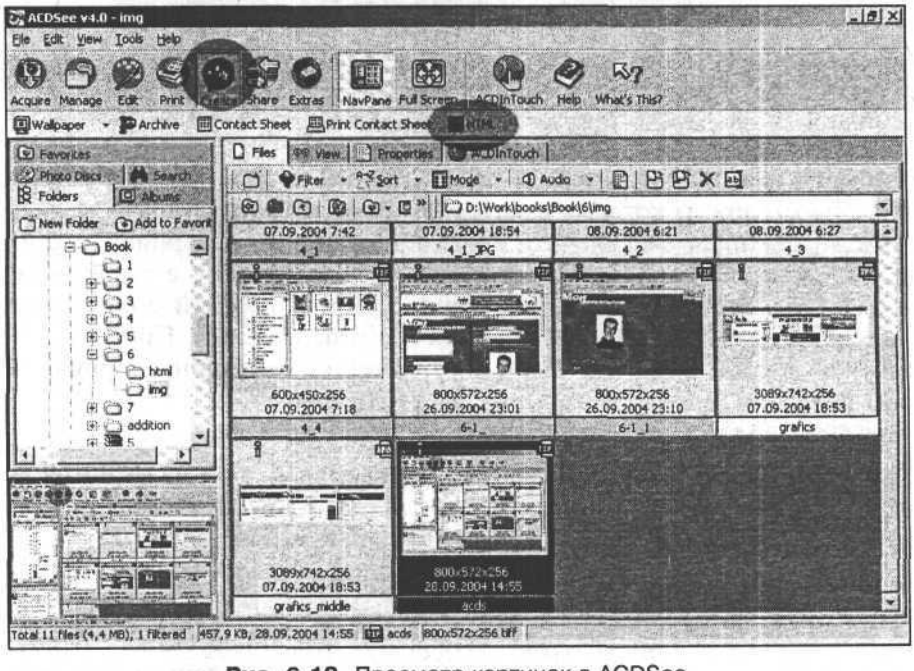


Рис. 6.18. Просмотр картинок в ACDSee.

Теперь одна тонкость: поскольку у меня должны быть и уменьшенные и полноразмерные картинки, то необходимо каким-то образом получить эти уменьшенные копии. Я знаю два относительно быстрых способа.

Способ первый заключается в применении тега `<img>`, имеющего атрибуты `height` и `width`, которые определяют, соответственно, высоту и ширину картинки. Если их указать, то картинка будет растянута/сжата под эти размеры, что довольно удобно и быстро, если бы не одна проблема: картинку приходится загружать с сервера всю. А когда таких картинок десять штук? Время загрузки значительно увеличивается, и поэтому нет никакого резона в показе маленьких картинок.

Второй способ основывается на использовании специально сделанных программ для подобных целей, которые автоматически генерируют действительно уменьшенные изображения. Я для генерации маленьких копий картинок обращаюсь к знаменитой программе ACDSee версии 4.0 (рис. 6.18).

При нажатии на кнопку **Create** появляется панель инструментов, на которой имеется кнопка **HTML**, нажатие на которую вызывает диалоговое окно для создания фотоальбома (рис. 6.19).

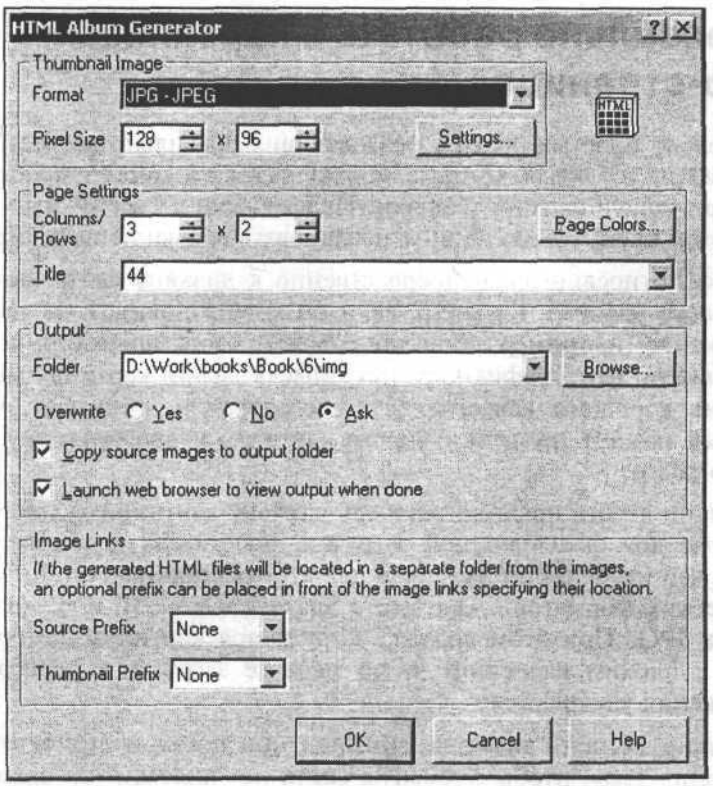


Рис. 6.19. Диалоговое окно **HTML Albom Generator** (из программы ACDSee)

Так как меня интересуют только уменьшенные картинки, а при большом уменьшении различия между форматами GIF и JPG практически отсутствуют, то мне достаточно просто нажать на кнопку **OK** (т. е. воспользоваться настройками по умолчанию).

Ускоряем загрузку картинок

Во-первых, понятно, что чем меньший объем файлового пространства занимает картинка, тем быстрее она грузится. Во-вторых, существуют специальные опции при сохранении картинок. Наверняка вы видели на сайтах, как большие картинки показывались постепенно: сначала картинка очень плохого качества, потом чуть получше, потом еще лучше и так до тех пор, пока не отобразится ее реальное состояние. Время уходит практически то же самое, но пользователь может уже заранее понять, хочет ли он дожидаться окончания или нет. Такой эффект достигается установкой специальных опций при сохранении файлов. Для формата GIF эта опция называется `interlaced`, а для формата JPEG — `progressive`.

Как правильно работать с картинками на web-странице

Есть несколько простых правил, позволяющих правильно работать с картинками на сайте, соблюдая которые можно избежать многих проблем, возникающих при использовании графики. Практически со всеми мы уже ознакомились в этой главе, нужно лишь акцентировать внимание на таких правилах:

- картинки, относящиеся непосредственно к дизайну сайта, должны быть небольшого размера. Конечно, если вы хотите показать на сайте полное изображение (например, обои для рабочего стола операционной системы или большую фотографию), то нет смысла делать картинку маленькой и не очень хорошего качества, ведь пользователь понимает, что у него в данный момент грузится большая картинка и она будет грузиться достаточно долго;
- желательно делать предварительную загрузку картинки худшего качества, чтобы, не дожидаясь полной загрузки, посетитель в общих чертах мог понять, что изображено на ней. Я для этого использую опции сохранения графических форматов: `interlaced` для формата GIF и `progressive` для формата JPG. При этом сначала загружается картинка, которая отображается с плохим качеством, и по мере ее загрузки с сервера качество изображения улучшается;
- желательно задавать размеры картинок при помощи атрибутов `height` и `width` тега `<img>`, чтобы элементы сайта не "плясали" во время загрузки (первоначально для картинок без заданных размеров отводится неболь-

шое место, а когда картинка начинает загружаться, это место расширяется, сдвигая все остальное вокруг себя, причем происходит это сдвигание буквально в течение нескольких секунд);

- всплывающую подсказку необходимо использовать всегда, если нужно, чтобы посетитель знал о картинке чуть больше, чем можно понять по ее внешнему виду. Например, если это картинка-гиперссылка, то необходимо в атрибуте `title` указать, куда эта ссылка ведет или какое действие произойдет при нажатии на нее;
- использование атрибута `alt` позволяет описать то, что должно быть изображено на картинке, даже если она по какой-то причине не загрузилась. Это правило может оказаться весьма полезным там, где используются изображения с надписями вместо обычного текста (так часто поступают с главным меню сайта). Если прописать атрибут `alt` у каждой такой картинки, то посетитель сможет производить навигацию по разделам сайта даже при отключенной графике. В случае отсутствия атрибута `alt` пользователь будет видеть лишь набор пустых прямоугольников с крестиками.

Создаем фон страницы

У самой HTML-страницы и у некоторых других элементов (например, у ячеек таблиц) может быть фон, который задается не просто цветом, а в виде изображения. Это может быть картинка, которая находится за текстом и создает на сайте определенный антураж. Например, в качестве фона личного сайта какого-нибудь писателя можно использовать изображение страницы, отпечатанной на пишущей машинке. Иногда для этих целей используют логотип компании или какой-то узор (в зависимости от фантазии). Самое главное, чтобы фон хорошо контрастировал с текстом, иначе текст будет трудно читать. Это важно.

Теперь о самом рисунке фона. Так как для фона используется некоторое изображение, оно обладает собственной шириной и длинной. В случае, когда размеры изображения фона меньше размеров элемента, к которому этот фон применяется, рисунок фона повторяется внутри этого элемента (по горизонтали и вертикали), как показано на рис. 6.20.

Если повторение рисунка фона нежелательно, необходимо прибегнуть к специальной инструкции, запрещающей такое повторение. Я узнал об этой инструкции подробнее, когда решил обойти проблемы с загрузкой графики на сайт. В верхней части моей web-страницы имеется картинка, которая не несет смысловой нагрузки, а служит в качестве дизайнерского украшения сайта. Она немаленькая, следовательно, будет долго загружаться при открытии каждой страницы моего сайта. Я решил, что если сделаю это украшающее изображение в виде фона, и с этим фоном возникнут проблемы, то посетитель просто решит, что так и надо.

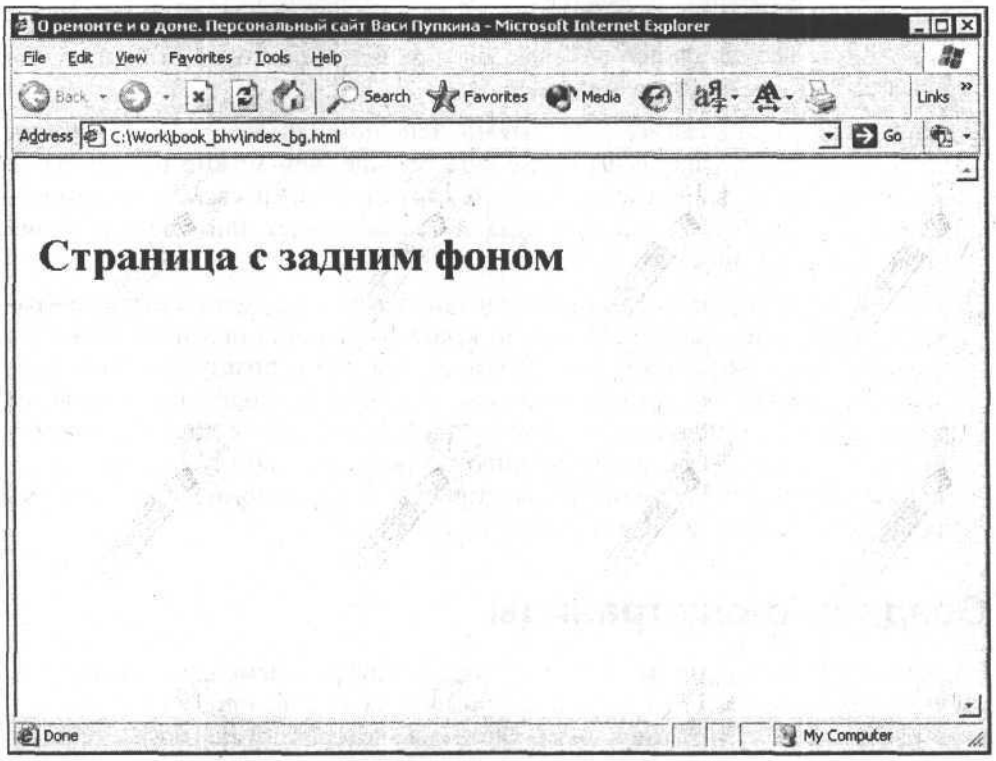


Рис. 6.20. Страница с рисунком для заднего фона

Довольный тем, что придумал такой замечательный выход из ситуации, я задал для ячейки таблицы, в которой должен быть этот дизайнерский рисунок, атрибут фона, открыл страницу в браузере и удивился: рисунок повторился внутри ячейки несколько раз.

Когда я стал искать информацию по созданию фона, то обнаружил, что есть средство для запрета повтора фона, которое реализуется при помощи стилей (нужно задать значение `no-repeat` у свойства `background-repeat`) (листинг 6.7).

Листинг 6.7

```
<td background="picture.gif" style="background-repeat: no-repeat;">
```

Кроме значения `no-repeat`, у свойства `background-repeat` существует еще несколько интересных его значений:

- `repeat-y` — повторять рисунок фона только по оси *Y*;
- `repeat-x` — повторять рисунок фона только по оси *X*.

Применение таких значений может оказаться полезным, когда нужно аккуратно задать фон для заголовка какой-нибудь таблицы, например, с плавным переходом цветов. В таком случае обычно берут заготовленное изображение, которое служит как бы макетом для этого фона, вырезают (например, при помощи Photoshop) картинку (полоску нужной высоты и шириной в 1 пиксел, чтобы картинка загружалась мгновенно) и делают ее фоном нужной ячейки таблицы, повторяя его только по горизонтали.

Также существует возможность управления способом прокручивания (рисунок фона может прокручиваться вместе с самой страницей, а может прокручиваться только содержимое страницы). Это задается стилями при помощи атрибута `background-attachment` со значением `fixed` (т. е. фиксированный фон) (листинг 6.8). По умолчанию принимается значение `scroll` (прокручивающийся фон).

Листинг 6.8

```
<body background="picture.gif"  style="background-attachment : fixed;">
```

Глава 7

Таблицы стилей CSS

Что такое таблицы стилей CSS и какая от них польза

Все сайты в большинстве своем отличаются друг от друга. На них используются различные шрифты, цвета, различные отступы и внешний вид одних и тех же элементов. Но если речь идет о каком-либо конкретном сайте, то его внешний вид от страницы к странице меняться не должен, иначе это будет просто неэстетично — равносильно тому, как если бы стены одной комнаты были оклеены разными обоями.

С первого взгляда кажется, что поддерживать все страницы в одном стиле сложно. Если вы используете шаблон-заготовку, а вам требуется что-то подправить не только в основном содержимом страницы, но и в других ее частях, даже в этом случае вы можете задеть часть шаблона, и тогда ваша страница уже перестанет быть похожей на остальные.

Нужен выход и он в свое время был найден. Его суть заключается в своеобразном обособлении описания внешнего вида элементов страницы от описания их расположения, размеров и состава.

Представьте себе такую ситуацию: вам нужно на странице (и далеко не на одной) сделать какой-то шрифт (например, заголовка таблицы) синего цвета. Вы создавали шаблон три недели тому назад, уже на его основе готова половина страниц сайта, а тут пришел ваш знакомый дизайнер и сказал, что заголовки таблиц должны быть не синие, как у вас, а оранжевые. Тогда, мол, ваш дизайн заиграет, все будет красиво. Вы посмотрели, попробовали на одной странице — и в самом деле лучше. Представляете, что вам предстоит сделать? Во всех уже готовых страницах (допустим, двадцати четырех) нужно отыскать заголовки таблиц и зафиксировать их цвет как оранжевый. Ну, подумаешь, скажете вы. Но если страниц не двадцать четыре, а значительно больше?

Представьте себе, что вместо описания цвета, размера, начертания заголовка вы в HTML-коде дали бы примерно такую инструкцию: "здесь текст показать как заголовок таблицы". А где-то в другом месте вы бы подробно описали, как именно нужно показывать заголовки таблиц. В таком случае вам достаточно было бы просто исправить синий цвет на оранжевый там, где описаны правила представления заголовков. А записи внутри страниц с инструкцией "здесь текст показать как заголовок" автоматически приняли бы оранжевый цвет. Таким образом, вам достаточно было бы сделать необходимое исправление всего лишь в одном месте. Одна строчка — и все! И такое возможно. Это делается при помощи так называемых *каскадных таблиц стилей CSS (Cascading Style Sheets)*.

Общий смысл стилей в том, что они задают какому-то конкретному элементу страницы или нескольким элементам внешний вид.

Использование таблиц стилей имеет много преимуществ. Во-первых — это средство соблюдения единства стиля на всем сайте, во-вторых — возможность быстрого изменения всего внешнего вида сайта без изменения самого содержимого страницы. Еще одно преимущество заключается в том, что уменьшается размер файлов всех страниц (ведь в таком случае предполагается только однократное описание стиля, который нужно применить). Представьте себе, насколько меньше кода будет в HTML-странице, если во всех заголовках таблиц сослаться на конкретное описание стиля вместо того, чтобы каждый раз подробно описывать цвет, размер, шрифт, фон и т. д.

Таблицы стилей (так же, как и сам язык HTML) имеют версии, а в последних версиях есть свойства, которые могут поддерживаться, а могут и не поддерживаться некоторыми браузерами. Не все свойства элементов страницы, которые можно задать при помощи простого HTML-кода, могут быть заданы таблицами стилей (но такие свойства уже редкое явление). В случае, когда таблицы стилей обособлены от основного HTML-файла, может возникнуть ситуация, при которой пользователь будет видеть отображение вашего HTML-кода без применения стилей. Вот, пожалуй, один из тех недостатков, с которым можно столкнуться при использовании стилей, однако имеющиеся недостатки с лихвой перекрываются достоинствами.

При грамотном подходе к созданию сайта и использованию CSS можно полностью изменить дизайн в считанные минуты. Кроме того, если учесть обилие устройств, которые понимают HTML-код (начиная от обычных персональных компьютеров и заканчивая мобильными телефонами), то очень полезно иметь свои HTML-страницы не загроможденными описанием внешнего представления. Пусть устройство пользователя отвечает за то, как оно будет отображать ваш сайт. Если оно сможет показать информацию с примененными к ней таблицами стилей, то отлично. Если нет — пользователь увидит простенькую страничку безо всяких стилей, но она загрузится быстро и будет показана корректно.

Как подключать таблицы стилей

Каскадные таблицы стилей — это отдельная от HTML-кода информация, которую, как и любую внешнюю сущность, необходимо подключить к этому коду.

Существуют три способа подключения таблицы стилей:

- непосредственное описание в коде HTML-страницы (inline-описание). С таким способом вы уже знакомы — это атрибут `style` у какого-либо тега. Подобный способ не дает имеющихся преимуществ в использовании CSS, поскольку описание стиля находится в том же месте, что и сам элемент web-страницы и его код. Все, что позволяет такое указание стилей, — это в первую очередь обращение к свойствам элементов, которые недоступны в обычном HTML-коде. Указанные свойства в этом случае действуют только в переделах тега, в котором они описаны. Пример описания стилей в коде HTML-страницы приведен в листинге 7.1 (здесь ячейке таблицы задается ширина 20 % и граница — сплошная, серого цвета, толщиной в 1 пиксел);

Листинг 7.1

```
<td rowspan="3" style="width:20%;border:1px solid #d4d4d4" valign="top">
```

- описание внутри заголовка `<head>` HTML-страницы. При этом способе внутрь тега `<head>` вставляется тег `<style>`. Свойства элементов внутри него оформляются несколько иначе, нежели в первом случае. Запись представляет собой сложную конструкцию (подробное описание будет представлено чуть позже). Суть этого способа заключается в том, что на странице имеется описание стилей не для конкретного элемента, а для целого набора элементов. Одним из недостатков такого способа является то, что он может быть применен только к одной странице (другие страницы этих стилей не видят). Пример описания стилей в заголовке HTML-страницы представлен в листинге 7.2. Сначала записывается при помощи некоторого выражения элемент, к которому будет применяться стиль, а затем в фигурных скобках указывается набор стилевых свойств элементов и их значений. При указании стилей, свойство элемента отделяется от его значения двоеточием. Если стилевых свойств несколько, то они указываются через точку с запятой;

Листинг 7.2

```
<head>
...
...
```

```
<style>
body {
  color: white;
  font-family: Georgia, Times New Roman, Times, Microsoft Serif;
  background-color: silver;
}
p.text {
color:darkblue;
}
</style>
...
</head>
```

- описание стилевых свойств во внешнем файле. В этом случае каскадные таблицы стилей выносятся во внешний файл с расширением CSS. Внутри такого файла они записываются аналогично тому, как записываются внутри тега `<style>` заголовка страницы (листинг 7.3). Кроме самого описания стилей в отдельном файле, необходимо указать в HTML-странице на тот факт, что стилевые свойства нужно применять именно из этого файла. Делается это в заголовке конкретной HTML-страницы при помощи тега `<link>` (листинг 7.4), в котором указывается ссылка на файл со стилями (`href`), тип содержимого (`type`) и то, чем является указанный файл относительно текущего HTML-файла (`rel`). Недостаток такого описания состоит в том, что стили помещены в отдельный файл и, следовательно, отдельно подгружаются с сервера. Страница может быть показана вообще без применения стилей, если вы случайно ошиблись в имени CSS-файла или произошел какой-то сбой во время его загрузки. К счастью, это довольно редкое явление.

Листинг 7.3

```
body {
  color: white;
  font-family: Georgia, Times New Roman, Times, Microsoft Serif;
  background-color: silver;
}
...
```

Листинг 7.4

```
<head>
...
<link href="css/common.css" type=text/css rel=stylesheet>
```

```
...
</head>
```

Теперь расскажу чуть подробнее о подключении стилей через описание в заголовке страницы или через внешний файл.

Тот элемент, к которому относится конкретное стилевое правило (то, что записано в фигурных скобках), называется селектор. На основании селектора из кода HTML-страницы выбираются те элементы, к которым нужно применять стиль. Имеются следующие типы селекторов:

- селектор для тега;
- селектор для класса;
- селектор для псевдо-класса;
- селектор по атрибуту `ID`.

Если нужно задать селектор для какого-либо конкретного тега (например, тега `<p>`), то селектор называется просто именем тега (листинг 7.5).

Листинг 7.5

```
P{
  font-size:12px;
  font-weight:bold;
}
```

Я не случайно написал имя селектора для тега `<p>` с большой буквы. Некоторые специалисты советуют называть селекторы для тегов именно таким образом. Разницы для браузера никакой (он не обращает внимания на регистр букв), а вот мне удобнее сразу визуально выделять такие селекторы. Обнаружив внутри HTML-страницы любой тег `<p>`, браузер применит к нему указанный стиль.

Селекторы для класса применяются только тогда, когда есть для этого специальное указание в теге. Если абзацы, например, могут быть разного вида, то необходимо разделить на несколько частей описание их стилей и, соответственно, обращение к этому описанию, что и делается при помощи селекторов для классов и атрибута `class`, который может быть задан практически у любого тега (листинг 7.6).

Листинг 7.6

```
...
p.annotation {
```

```
  font-size:12px;
  text-align:right;
}
.text {
  font-size:14px;
  color:black;
}
...
```

Примечание

Два селектора в листинге 7.6 отличаются присутствием описания тега <p>. Первый селектор (p.annotation) выберет и применит указанный для него стиль ко всем тегам <p>, внутри которых есть атрибут class равный annotation. Второй селектор (text) применит указанный для него внешний вид к любому тегу с атрибутом class="text".

Если мы хотим, например, абзацу (тегу <p>), содержащему аннотацию к статье, присвоить описанный в каскадных таблицах стилей внешний вид, то должны написать так, как представлено в листинге 7.7.

Листинг 7.7

```
<p class="annotation">Текст абзаца-аннотации</p>
```

Запись class="annotation" — это именно то описание, которое я условно называл в начале главы "здесь текст показать как заголовок таблицы".

Селекторы для псевдокласса используются не очень часто и ограниченно. Я с их помощью фиксирую различные стили для гиперссылок. Такие селекторы определяют внешний вид самих гиперссылок (активных, посещенных и текущих). В листинге 7.8 приведен пример, в котором представлена запись внутри файла CSS, делающая все гиперссылки на странице размером 14 пикселов, полужирным шрифтом и темно-синего цвета.

Листинг 7.8

```
A:link {
  font-size: 14px;
  font-weight:bold;
  color: darkblue;
}
```

Всего имеется семь псевдоклассов, из которых я использую только четыре: link, visited, hover, active. Остальные три: focus, first-child и lang мне не нужны, поскольку некоторые из них не поддерживаются нормально.

Теперь рассмотрим селекторы для атрибута ID. Практически у каждого тега на web-странице может быть атрибут ID, который его однозначно выделяет среди других таких же тегов, т. е. в HTML-коде не может быть два тега с одинаковым атрибутом ID. Селекторы для ID похожи на селекторы для классов, но классов может быть много в документе, а ID – уникален. Такие селекторы даже описываются так же, как и селекторы для класса, только вместо точки используется знак решетка # (листинг 7.9).

Листинг 7.9

```
p#annotation {
  font-size:12px;
  text-align:right;
}
#text {
  font-size:14px;
  color:black;
}
```

Для применения стилей, указанных в листинге 7.9, в HTML-коде должна быть запись, подобная указанной в листинге 7.10 (обратите внимание на атрибуты id).

Листинг 7.10

```
<p id="annotation">Здесь аннотация</p>
...
  <td id=text>
     ...Здесь текст...
   </td>
...
```

В зависимости от необходимости, могут использоваться все три способа подключения таблиц стилей (in-line-описание, внутри заголовка <head>, с помощью внешнего файла). Например, если необходимо описать внешний вид какого-то элемента, отличающегося от остальных, я использую in-line-описание. Если мне нужно использовать исключительно HTML-код, а внешние файлы не могу подключить, то я прописываю стили внутри тега

<head>. И, наконец, в обычных обстоятельствах при создании сайта я использую внешние файлы со стилями. Кроме того, можно применять одновременно несколько способов и тогда в зависимости от приоритета описания будет подключаться какой-то один из нескольких (именно поэтому они и называются каскадными).

При одновременном использовании разных способов подключения браузер должен точно знать, какие именно стили надо применять для конкретного элемента. В листинге 7.11 приведен фрагмент файла с таблицами стилей — common.css. Обратите внимание на это имя файла и то, как он подключается в теге <link> (листинге 7.12).

Листинг 7.11

```
.text1 {
  text-decoration: none;
  text-align:left;
  color:#b4b4b4
}
```

В листинге 7.12 представлен сам HTML-файл, с которым работаем. В нем используются все три способа подключения стилей. Я специально показал в этом примере стили с одинаковыми свойствами, но разными их значениями, чтобы четче понять, каким образом их интерпретирует браузер (рис. 7.1).

Листинг 7.12

```
<html>
<head>
<!-- Подключение стилей из внешнего файла -->
<link type="text/css" rel="stylesheet" href="common.css">
<META http-equiv="Content-Type" content="text/html; charset=windows-
1251">
<!-- Подключение стилей в секции заголовка при помощи тега "style" -->
<style>
  p {
    text-decoration: none;
    font-size:140%;
    text-align:right;
  }
  p.text1 {
    text-decoration: underline;
    text-align:right;
```

```
    line-height:10;
  }
</style>
</head>
<body>
<table border="1">
<tr>
 <td>
 <!-- Подключение стилей в самом теге при помощи атрибута style -->
 <p class="text1" align="left" style="text-align:center;line-height:2;">
      Текст первого абзаца. Текст первого абзаца.
      Текст первого абзаца. Текст первого абзаца.
  </p>
  <p> Текст второго абзаца. Текст второго абзаца.  </p>
 </td>
</tr>
</table>
</body>
</html>
```

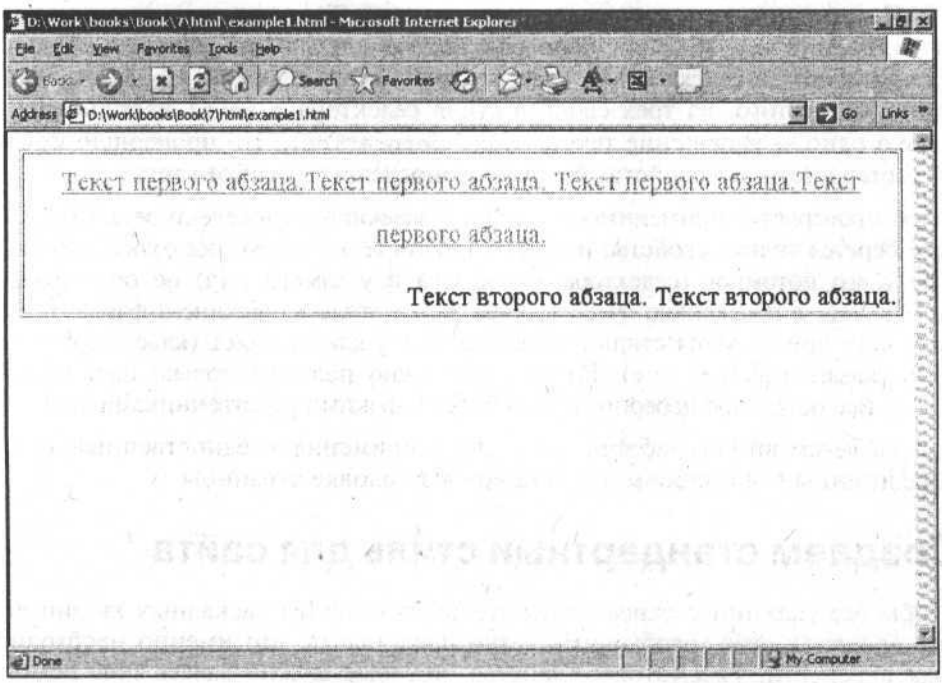


Рис. 7.1. Внешний вид кода из листинга 7.12 после применения каскадных таблиц стилей

Рассмотрим, почему страница получилась именно такой. Во-первых, необходимо понять механизм наследования, который используется в таблицах стилей. Представьте себе отца и сына. У отца кучерявые волосы, нос "крючком" и прыгающая походка. Сын унаследовал от него все эти свойства — он тоже кучерявый, с крючковатым носом и подпрыгивает, когда идет. Но вот сын вырос и ему перестал нравиться собственный нос. Он пошел к пластическому хирургу и сказал: "Сделайте мне другой нос". Хирург сделал. Тем самым он переопределил свойства носа у парня. А все остальное, что не трогали, осталось как у отца. В таблицах стилей роль такого отца и сына играют различные способы описания стилей для элементов и различные способы задания селекторов.

Сначала применяются стилевые свойства самого элемента (в данном случае — тега <p>). Вы видите строчку:

```
<p class="text1" align="left" style="text-align:center;line-height:2;">
```

В ней описано выравнивание текста внутри абзаца (`text-align`) и расстояние между строчками (`line-height`). Обратите внимание, кстати, что применился стиль `text-align:center`, а не атрибут `align="left"`, хотя по смыслу они одинаковые. Надо отметить, что стили главнее атрибутов. Как видите, в описании стилей нет никакой информации о том, что текст должен быть показан серым цветом и подчеркнут. Но у тега <p> есть атрибут `class`, который нужен для работы селекторов таблиц стилей. В заголовке HTML-страницы есть второе описание стилей. Там записан стиль для класса `text1` тега <p> и стиль непосредственно для любого тега <p>. Первым воспринимается селектор для класса у тега (он главнее селектора для самого тега). Кроме того, из трех свойств стиля селектора `p.text1` использовалось только одно — украшение текста (`text-decoration`). Это произошло потому что оставшиеся два свойства были переопределены у самого тега <p>.

Затем проверяется применимость свойств, задающихся в селекторе для тега. От него берется только свойство размера шрифта (`font-size`), поскольку это свойство у его потомков (селектора для класса и у самого тега) не описывалось. И, наконец, в последнюю очередь проверяется стиль из внешнего файла. К данному тегу применяется стиль с селектором для класса `text1` (класс этот может быть указан в любом теге). Из него возможно получить только цвет шрифта `color`. Все остальное переопределено более близкими родственниками тега <p>.

Что касается второго абзаца, то к нему применился единственный стиль, определенный селектором для тега <p> в заголовке страницы.

Создаем стандартный стиль для сайта

Чтобы все указанные ранее положительные качества каскадных таблиц стилей проявили себя, необходимо четко представить, что именно необходимо туда перенести. Рассмотрим свойства, которые можно задать при помощи таблиц стилей.

Основные свойства шрифта

К основным свойствам шрифта относятся следующие:

- `font-family` — показывает браузеру, каким именно шрифтом отображать текст, к которому применен этот стиль. Можно указать только один шрифт, а можно и несколько, через запятую. В последнем случае браузер показывает текст первым видом шрифта из указанного списка, если же такой шрифт на компьютере пользователя отсутствует, то используется второй шрифт и т. д. Если у пользователя нет ни одного шрифта из указанного списка, то браузер применяет какой-либо стандартный шрифт, существующий по умолчанию;
- `font-size` — задает размер шрифта. В CSS применимы все стандартные размеры шрифтов, начиная от процентного отношения к базовому шрифту и заканчивая абсолютными значениями в пискелах, миллиметрах и т. д. Имеются также и стандартные значения — ключевые слова (они перечислены в порядке увеличения шрифта: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`);
- `font-style` — характеризует начертание шрифта: обычное (`normal`) и курсив (`italic`). Наклонное начертание (`oblique`) — это тот же курсив;
- `font-weight` (weight в переводе с английского означает нажим). Представьте себе мягкий карандаш: чем сильнее на него нажимаешь, тем толще и жирнее получается черта. Так же и со значением свойства `font-weight`. Оно может находится в пределах от 100 до 900 (от тонкого к очень жирному) и быть кратным 100 или просто иметь значение `bold` (так называемое полужирное начертание).

Основные свойства текста

Для характеристики текста также существуют некоторые свойства:

- `line-height` — указывает на расстояние между строками текста. Если вы зададите `line-height:3`, то браузер поймет это как утроенное расстояние между строками, а если зададите: `line-height:3px`, то увидите сплошную "кашу" из налезающих друг на друга строчек;
- `text-align` определяет способ выравнивания текста. Как уже подробно рассматривалось ранее, возможно выравнивание:
  - по левому краю (`left`), что является выравниванием по умолчанию;
  - по центру (`center`);
  - по правому краю (`right`);
  - по ширине (`justify`);
- `text-decoration` — украшение текста:
  - подчеркивание (`underline`);

- надчеркивание (overline);
- перечеркивание (line-through);

- text-indent — отступ красной строки, который по умолчанию равен нулю;
- text-transform — задает регистр букв для содержимого тега. Это свойство может принимать значения:
  - capitalize (первая буква каждого слова заглавная);
  - uppercase (все буквы заглавные);
  - lowercase (все буквы строчные);
  - none (текст отображается в таком виде, в каком он введен).

Основные свойства цвета и фона

- color — устанавливает цвет основного содержимого тега. Свойство может применяться как к тексту, так и к различным элементам. Например, относительно абзаца оно задает цвет текста, а относительно горизонтальной черты — цвет самой черты;
- background-color — задает цвет фона элемента. Помимо любого возможного цвета фон может быть и прозрачным (transparent);
- background-image — подключает рисунок фона элемента из указанного файла;
- background-attachment — предоставляет возможность прокрутки рисунка фона вместе с основным содержимым страницы, может принимать значения относительно окна браузера:
  - scroll (прокручивать);
  - fixed (не прокручивать);
- background-position — служит для задания координат начала фонового рисунка (по умолчанию принимается левый верхний угол). Если нужное место задается при помощи одного числа, то координаты этой начальной точки равны между собой, если при помощи двух чисел, идущих через пробел, то первое число означает отступ по оси *X* от верхнего левого края элемента, а второе — отступ по оси *Y*. Также можно задавать начальную точку, как бы выравнивая рисунок фона при помощи ключевых слов: left, right, center (по оси *X*) и ключевых слов: top, center, bottom (по оси *Y*);
- background-repeat — определяет способ повторения рисунка фона:
  - размножается во все стороны (по умолчанию);
  - размножается только по горизонтали (repeat-x);
  - размножается по вертикали (repeat-y);
  - не размножается вообще (no-repeat).

Основные свойства обрамления элементов

У элементов, которые могут обладать обрамлением (границами), существуют свойства, определяющие внешний вид границ таких элементов и различных отступов вокруг этих границ и элементов. Для большей наглядности на рис. 7.2 приведен пример некоторого абстрактного элемента и свойств его обрамления.

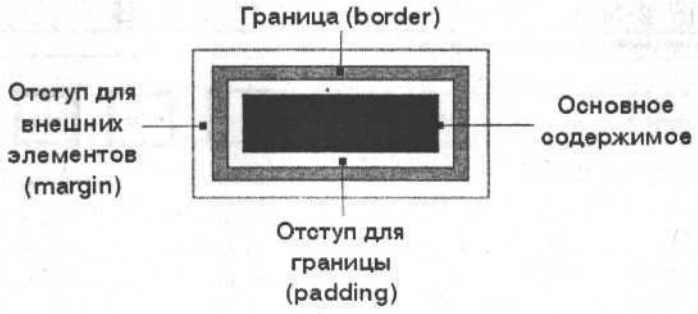


Рис. 7.2. Элемент (содержимое тега) и его обрамление

Рассмотрим эти свойства:

- `margin` и `padding` похожи между собой, они отличаются только положением относительно границы элемента:
  - `padding` — это отступ между содержимым и его обрамлением (есть вариант перевода этого слова как "набивка для подушки");
  - `margin` — это отступ от внешних элементов (в переводе с английского — кайма, грань).

  Оба эти свойства задаются двумя способами: либо просто `margin` и `padding`, либо отдельно для каждой из четырех сторон: `left`, `right`, `top`, `bottom` (например, `margin - top:8px`; `padding - top: 6px`);
- `height` и `width` — это высота и ширина элемента (например, картинки или таблицы);
- `border-style` — определяет внешний вид обрамления (границы) элемента. Существуют следующие разновидности стиля обрамления:
  - `none` (нет обрамления);
  - `dotted` (точечная линия);
  - `dashed` (пунктирная);
  - `solid` (сплошная);
  - `double` (двойная);
  - `groove` (вдавленная граница);

- `ridge` (выпуклая граница);
- `inset` (вдавленный элемент);
- `outset` (выпуклый элемент).

На рис. 7.3 представлены все перечисленные разновидности.

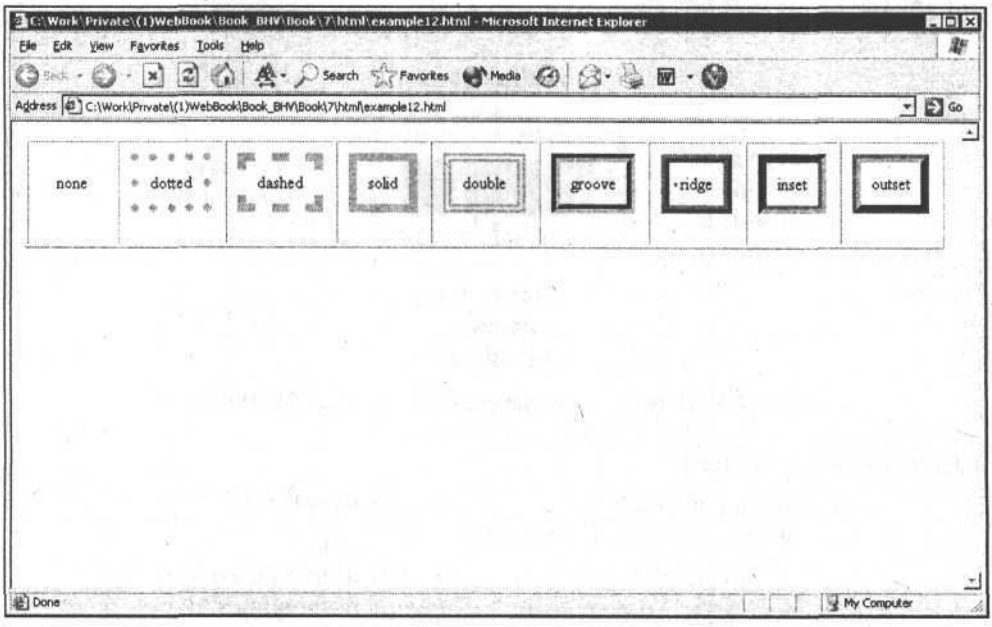


Рис. 7.3. Различное оформление внешнего вида границы элемента

Примечание

Следует внимательно следить за тем, к какому элементу применяется стиль. Например, когда я готовил HTML-файл с примером, который использовал для этого рисунка, то каждую запись (название стиля внешнего вида) поместил в отдельную таблицу с единственной ячейкой. Получилось 9 таблиц с различным видом обрамления, вставленные в ячейки одной большой таблицы. Однако они оказались "склеенными" друг с другом, и чтобы их отделить, я задал в секции заголовка `<head>`, параметры `padding:8px` и `margin:2px` с селектором `table` (т. е. для каждой таблицы), однако применился только один параметр — `margin`. Оказывается, я должен был задавать `padding` не для таблиц, а для ячеек `<td>`, т. к. именно от ячеек отсчитывается отступ до обрамления.

- `border-width` — определяет толщину границы элемента и задается точно так же, как и любая другая ширина: в процентах, пикселах, сантиметрах и т. д.;
- `border-color` — цвет границы.

Все свойства обрамления могут быть заданы не для всей границы сразу, а для каждой отдельной стороны элемента (`top` (верхняя), `right` (правая), `bottom` (нижняя), `left` (левая)). Можно указывать свойства для какого-либо элемента по отдельности (листинг 7.13), а можно и подряд через пробел в определенной последовательности (листинг 7.14).

Листинг 7.13

```
TD {border-width:3px;
    border-style:solid;
    border-color:black;
   }
```

Листинг 7.14

```
TD {border :3px solid black;}
```

Основные свойства списков

У списков свойств не так уж и много, причем вы помните, что существуют упорядоченные и неупорядоченные списки (соответственно `<ol>` и `<ul>`):

- `list-style-type` — это свойство, аналогичное атрибуту `type` для `<ol>` и `<ul>`, может принимать различные значения, в зависимости от того, упорядоченный список или нет.

  Для неупорядоченных списков существуют значения:

  - disc;
  - circle;
  - square.

  Для упорядоченных списков существуют следующие значения:

  - `decimal` (цифры);
  - `lower-roman` (маленькие римские);
  - `upper-roman` (большие римские);
  - `lower-alpha` (маленькие латинские);
  - `upper-alpha` (большие латинские);
  - `none` (нет). В этом случае перед элементом списка ничего не выводится;
- `list-style-image` — это свойство, позволяющее выводить перед элементами списка картинку. Значением такого свойства является запись типа

`url` (путь к картинке). Пример приведен в листинге 7.15 (в этом случае перед элементами списка будет выведена картинка с относительным путем img/list_pic.gif).

Листинг 7.15

```
list {
list-style-image: url(img/list_pic.gif);
}
```

Свойства полосы прокрутки

Сразу хочу заметить, что во-первых, свойства полосы прокрутки работают только в MS Internet Explorer, во-вторых, специалисты не советуют менять привычный внешний вид полосы прокрутки (рис. 7.4).

Рис. 7.4. Полоса прокрутки и ее составляющие

Существуют следующие свойства полосы прокрутки:

- *scrollbar-face-color* — основной цвет ползунка полосы прокрутки (по умолчанию светло-серый);
- *scrollbar-highlight-color* — подсветка ползунка (по умолчанию белый);
- *scrollbar-shadow-color* — тень от ползунка (темно серый);
- *scrollbar-darkshadow-color* — тень подложки, по которой двигается ползунок, (черный цвет);
- *scrollbar-arrow-color* — цвет стрелок полосы прокрутки;
- *scrollbar-track-color* — фон подложки полосы прокрутки.

Единицы измерения размеров

Единицы для задания размеров бывают относительные (в процентах) и абсолютные:

- em (ширина буквы *m* в используемом шрифте);
- ex (высота буквы *x* в используемом шрифте);
- px (пикселы);

- mm (миллиметры);
- cm (сантиметры);
- in (дюймы = 2,54 сантиметра);
- pt (пункты = 1/72 дюйма);
- pc (пики, двенадцать пунктов).

Создание сквозного стиля для всех страниц

Итак, вспомним подготовленную нами ранее "рыбу" (рис. 7.5) и модель навигации. Теперь надо решить, какие из элементов можно описать в каскадных таблицах стилей.

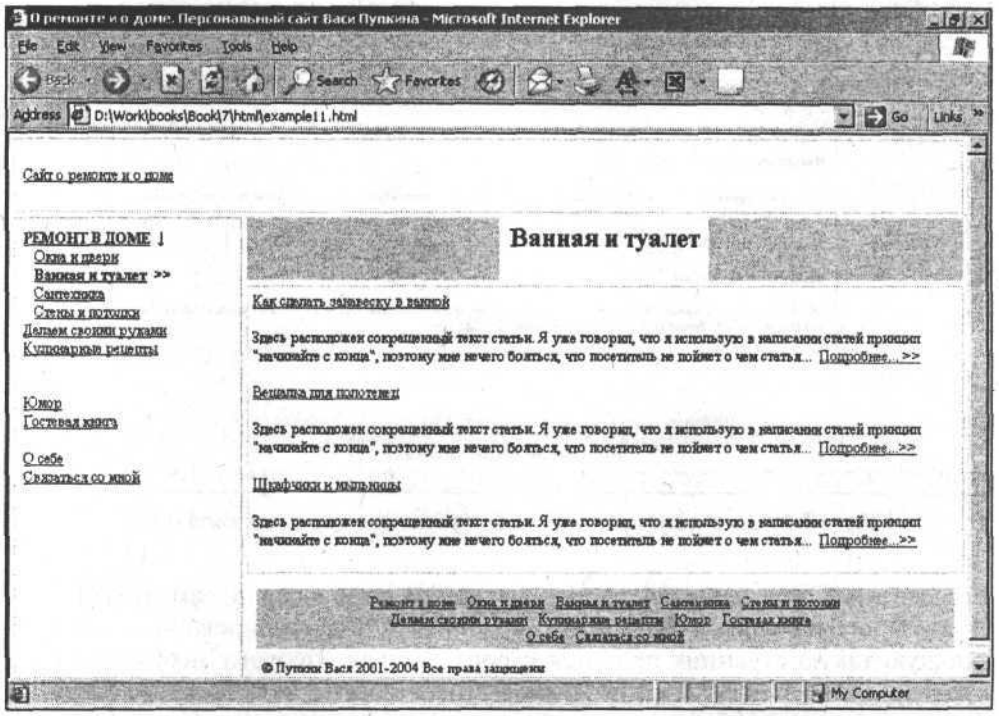


Рис. 7.5. Основа будущего сайта

Внимательно посмотрев на внешний вид своего будущего сайта и его структуру, я пришел к выводу, что нужно его упростить. Вся суть останется прежней, но мне совершенно разонравились серые области вокруг заголовка для основного содержимого страницы. Кроме того, я решил растянуть подвал на всю ширину. Это потребует незначительного изменения структуры таблицы каркаса сайта. Думаю, вы уже хорошо понимаете, как строятся таблицы,

помните, как я убирал лишние ячейки, исправлял и удалял атрибуты `rowspan` и `colspan` и т. д.

Внеся соответствующие изменения, я всерьез принялся за создание окончательного варианты внешнего вида своего будущего сайта (на рис. 7.6 представлена его заготовка).

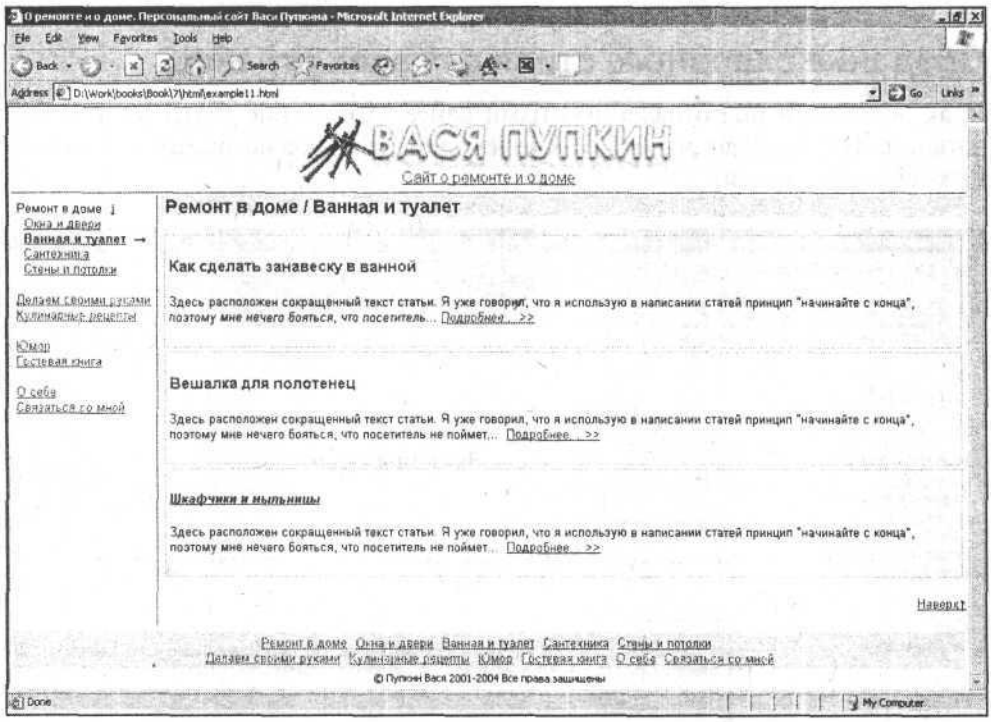


Рис. 7.6. Заготовка окончательного внешнего варианта вида сайта

Естественно, что большая часть изменений вносилась в сам HTML-код. В итоге он получился большим, сложным, неструктурированным, и следующую такую страницу придется строить с нуля. Поэтому необходимо аккуратно и правильно перенести всю информацию о стилях во внешний файл, чтобы можно было воспользоваться всеми преимуществами CSS.

Начнем с самых глобальных элементов:

- основной стиль самой страницы и стили полосы прокрутки;
- основные стили базовой таблицы, формирующей скелет таблицы и ее ячеек;
- стиль общих элементов для всей страницы (картинки, горизонтальные полоски, гиперссылки по умолчанию и т. д.).

Основой стиль самой страницы — это стили для тега <body>. Они могут определять отступы основного содержимого страницы от границ окна, фон страницы (его цвет или рисунок), способ прокрутки рисунка фона, основной шрифт страницы и т. д. (листинг 7.16).

Листинг 7.16

```
body {
   background-color: white;
   margin-top:0 ;
   margin-left:0;
   margin-right:0 ;
   margin-bottom:0;
   font-size: 11 pt;
   font-family: arial;
   scrollbar-face-color: #D1d1d1;
   scrollbar-highlight-color: #FFFFFF;
   scrollbar-shadow-color: #DEE3E7;
   scrollbar-3dlight-color: #D1D1D1;
   scrollbar-arrow-color:  gray;
   scrollbar-track-color: #EFEFEF;
   scrollbar-darkshadow-color: #98AAB1;

}
```

Напомню некоторые свойства:

- `background-color` — это цвет фона (обычно я пишу, что он белый, даже если его нет вовсе);
- `margin` — это отступы (как видите, их четыре — с каждой стороны; вместо этих четырех записей можно написать просто `margin:0`);
- `scrollbar` — это стили различных частей полосы прокрутки;
- `font-size: 11 pt` и `font-family: arial` — указывает на то, что любой, не переопределенный другим стилем текст будет выводиться шрифтом `arial` и размером `11 pt`.

Примечание

Естественно, что я убираю в самом HTML-файле те стилевые описания, которые перенес в таблицы стилей (в CSS-файл), в результате чего HTML-страница становится легче и более аккуратной.

Далее рассмотрим основные стили базовой таблицы, формирующей скелет дизайна.

Они состоят из стилей для таблицы и стилей для ее ячеек.

Селектор для базовой таблицы назовем `table.main`. Обратите внимание на то, что я назвал стиль по смыслу, а не по внешнему виду (написал о сути таблицы, а не о ее внешнем представлении). Стили предназначены для смены внешнего вида элементов. Если я назову стиль, к примеру, "большой красный текст", а потом захочу изменить цвет таких элементов на оранжевый, то у меня будет либо несоответствие в названии стиля и его внешнего вида, либо придется переделывать все описания классов для CSS, где я указывал этот стиль. В листинге 7.17 описаны стили основной таблицы-каркаса для страницы (заданы длина, ширина и нулевая граница).

Листинг 7.17

```
table.main {
  width:100%;
  height:100%;
  border:0;
}
```

Представьте себе, что у меня имеется 50 страниц, построенных одинаковым образом (как вы помните, я делаю свои страницы при помощи резинового дизайна). И вдруг по какой-то причине мне понадобилось использовать жесткий дизайн. Мне не надо переделывать все 50 страниц. Поскольку я вынес стиль основной таблицы в CSS-файл, то смогу исправить описание стилей для селектора `table.main` таким образом, как представлено в листинге 7.18.

Листинг 7.18

```
table.main {
  width:770px;
  height:100%;
  border:0;
}
```

Ширину я ограничил размером в 770 пикселов, а высоту оставил размером 100 %. Изменение дизайна с резинового на жесткий произойдет со всеми страницами одновременно.

Внимание

Применять свойства позиционирования (т. е. указывать ширину, длину, положение на странице) при помощи CSS не желательно, потому что в случае

каких-нибудь проблем с загрузкой таблиц стилей, ваша страница может "поплыть". Если шрифт из CSS не загрузится, браузер сам подберет ему замену. А вот если какой-то элемент размером 20 × 20 пикселов из-за проблем с загрузкой стилей браузер вдруг решит растянуть на 100 %, ваша страница будет изуродована.

Следующий элемент, который надо рассмотреть, — это описание стиля для ячеек таблицы-каркаса. `Td.menu` — это стиль для всех ячеек основной таблицы каркаса (листинг 7.19).

Листинг 7.19

```
td.menu {
 border-right: 1px solid #345678;
 vertical-align:top;
 padding:6px;
}
td.header {
  text-align:center;
  align:center;
  vertical-align:bottom;
  padding:4px;
  height:80px;
  border-bottom:1px solid #345678;
  background-repeat: no-repeat;
  background-position:center;
  background-image: url(../img/header2.gif);
}
```

Теперь вернемся к стилю общих элементов страницы (гиперссылки по умолчанию, картинки, горизонтальные полоски и т. д.).

В листинге 7.20 представлено, как эти стили должны быть описаны в файле CSS.

Листинг 7.20

```
/* Стили для основного вида гиперссылок (по умолчанию) */
A:active {
  font-size: 10 pt;
  color:#4e4593;
  font-family: arial;
```

```
  text-decoration: underline;
}
A:link {
  font-size: 10 pt;
  color:#4e4593;
   font-family: arial;
  text-decoration: underline;
}
A:visited {
  font-size: 10 pt;
  color:maroon;
  font-family: arial;
  text-decoration: underline;
}
A:hover {
  font-size: 10 pt;
  color:#4e4593;
  background-color:#FFFFAA;
  font-family: arial;
}
/* Общий стиль для абзаца*/
P {font-size: 10 pt;}

/* Стиль для горизонтальной полоски */
HR {color: black;}

/* Общий стиль для картинок*/
IMG {border:0px}
```

Обратите внимание на то, как оформлены стили для гиперссылок. По умолчанию гиперссылки определены как подчеркнутый текст (только для гиперссылки, находящейся под наведенным курсором, нет подчеркивания, что необходимо для того, чтобы отличать текущие гиперссылки от обычных). У гиперссылки с псевдоклассом `visited` не следует переопределять цвет, либо делать его близким к лиловому (по умолчанию — цвету посещенных ссылок).

Стили для бокового меню состоят из набора гиперссылок. Причем, одна из этих гиперссылок должна отличаться от других (например, цветом) чтобы пользователь мог быстро определить, в каком именно разделе он находится. Если вы посмотрите еще раз на заготовку ("рыбу" страницы), по которой я

делаю таблицы стилей для своего сайта, то увидите, что там имеются две отличающиеся гиперссылки. Одна из них написана обычным текстом (бывшая ссылка на раздел "Ремонт в доме"), чтобы было понятно, какой раздел сейчас развернут со всеми подразделами. Вторая гиперссылка ("Ванная и туалет") — это текущий подраздел, который я хочу сделать полужирным начертанием, чтобы он бросался в глаза пользователю, когда тот вернется к этому меню для перехода к другому разделу или подразделу. Стиль гиперссылок для текущего подраздела описывается в листинге 7.21. Гиперссылку с текста "Ремонт в доме" я должен снять, т. к. это текущий раздел верхнего уровня и все его подразделы доступны.

Листинг 7.21

```
/* Стили для гиперссылок текущего подраздела*/
A:link.current {
  color:maroon;
  font-weight:bold;
}

A:visited.current {
  color:maroon;
  font-weight:bold;
}

A:hover.current {
  color:maroon ;
  font-weight:bold;
}

A:active.current {
  color:#ffffaa ;
  font-weight:bold;
}
```

И наконец, стиль для ячеек, в которых будут располагаться сведения об авторских правах и текстовые гиперссылки подвала. Текст об авторских правах, находящийся внутри ячейки, будет небольшого размера, черный, написанный шрифтом `tahoma` (листинг 7.22).

Листинг 7.22

```
.copyright{
  font-size: 8 pt;
  font-family: tahoma;
  color: black;
  text-align:center;
}
```

Для гиперссылок в подвале уже есть описание — это стиль гиперссылок по умолчанию.

Переходим к основному содержимому страницы — списку статей и самим статьям.

Стиль для основного текста страницы в принципе отсутствует, поскольку он находится внутри каких-либо других тегов, к которым будут применяться собственные стили.

Приведу перечень компонентов, из которых состоит текст основного содержимого большинства страниц моего сайта:

- заголовок страницы;
- заголовок статьи;
- аннотация;
- абзац;
- гиперссылка в основном тексте страницы (также гиперссылки "Подробнее" и "Наверх");
- списки.

Заголовок страницы и заголовки статей я задаю при помощи тегов `<h1>` и `<h2>` соответственно, поэтому и стили для них описываю с указанием селекторов для тегов (листинг 7.23).

Листинг 7.23

```
H1{
  font-size: 14 pt;
  color:#4e4593;
  background-color:lightyellow;
}
H2{
  font-size: 12 pt;
  color:#4e4593;
}
```

Аннотацию описываю при помощи стилей для селектора по классу тега `<p>`, ведь аннотация — это всегда отдельный абзац (листинг 7.24).

Листинг 7.24

```
p.annotation{
  font-style:italic;
}
```

Абзацы и гиперссылки в основном тексте страниц я оставляю без изменений, поскольку их уже определял ранее (для основного содержимого всего сайта). Также мне необходимо задать стиль для контейнера (листинг 7.25), внутри которого помещаю краткую информацию по статьям (когда вывожу их список). Это необходимо, чтобы визуально отделять информацию о разных статьях. При помощи иных тегов я не могу задавать стили, потому что использую и заголовки <h2> и абзацы <p> и гиперссылки.

Листинг 7.25

```
DIV{
margin-top:0;
border:1px solid #d4d4d4;
padding:4px;
margin:4px
}
```

Превращаем обычную страницу в страницу с таблицами стилей

Прежде чем начать превращать страницу-шаблон в страницу с внешними таблицами стилей, я всегда делаю резервную копию страницы. Это опыт. Не один раз я попадал в капкан тонкостей языка HTML или собственной невнимательности, и тогда приходилось восстанавливать шаблон едва ли не с нуля.

Обычно для переноса лишней информации из HTML-файлов в таблицы стилей, я поступаю следующим образом: мне известны элементы, которые будут описаны в CSS, поэтому могу удалить из таких тегов информацию о стилях, а если ее не хватает в CSS, то перенести туда. Я отыскиваю по всему коду страницы-шаблона с заготовкой дизайна схожие элементы и очищаю их от стилей. Затем проверяю, не изменилось ли отображение страницы. Полезно при такой работе держать браузер все время открытым. Когда я вношу изменения в код, а потом перехожу к окну браузера, то вижу там сначала старый вариант, после чего нажимаю в браузере клавишу <F5> (перечитать содержимое страницы) и передо мной появляется уже откорректированная версия. При правильном переносе стилей из HTML-кода во внешний CSS-файл изменений во внешнем виде страницы быть не должно.

В листинге 7.26 представлен конкретный пример переноса.

Листинг 7.26

```
...
<title>О ремонте и о доме. Персональный сайт Васи Пупкина</title>
</head>
<body link="#4e4593" vlink="maroon" alink="#4e4593">
<!-- Начало таблицы-каркаса-->
<table border="0" width="100%" height="100%">
 <tr>
   <td style="border-bottom:1px solid #345678;"
       valign="top" cellpadding="4px" colspan="2"
       height="80px">
 <!-- Начало верхнего блока страницы-->
   <table style="width:100%;height:80px">
    <tr>
      <td background="../img/header2.gif" align="center"
          valign="bottom"
          style="background-repeat: no-repeat; background-position:center"
          height="80">
          <a href="http://dom-remont.narod.ru" style="font-size:12pt">
          Сайт о ремонте и о доме
          </a>
      </td>
    </tr>
   </table>
<!-- Конец верхнего блока страницы -->
...
...
```

Листинг 7.27 иллюстрирует ситуацию, которая получилась после вынесения всех стилевых свойств из этого кода во внешний CSS-файл.

Листинг 7.27

```
...
<title>О ремонте и о доме. Персональный сайт Васи Пупкина</title>
<link type="text/css" rel="stylesheet" href="style.css">
</head>
<body>
```

```
<!-- Начало таблицы-каркаса-->
<table class="main">
 <tr>
   <td colspan="2">
   <!-- Начало верхнего блока страницы-->
   <table style="width:100%;">
   <tr>
   <td class="header">
    <a href="http://dom-remont.narod.ru" style="font-size:12pt">
     Сайт о ремонте и о доме
   </a>
   </td>
   </tr>
   </table>
   <!-- Конец верхнего блока страницы -->
...
```

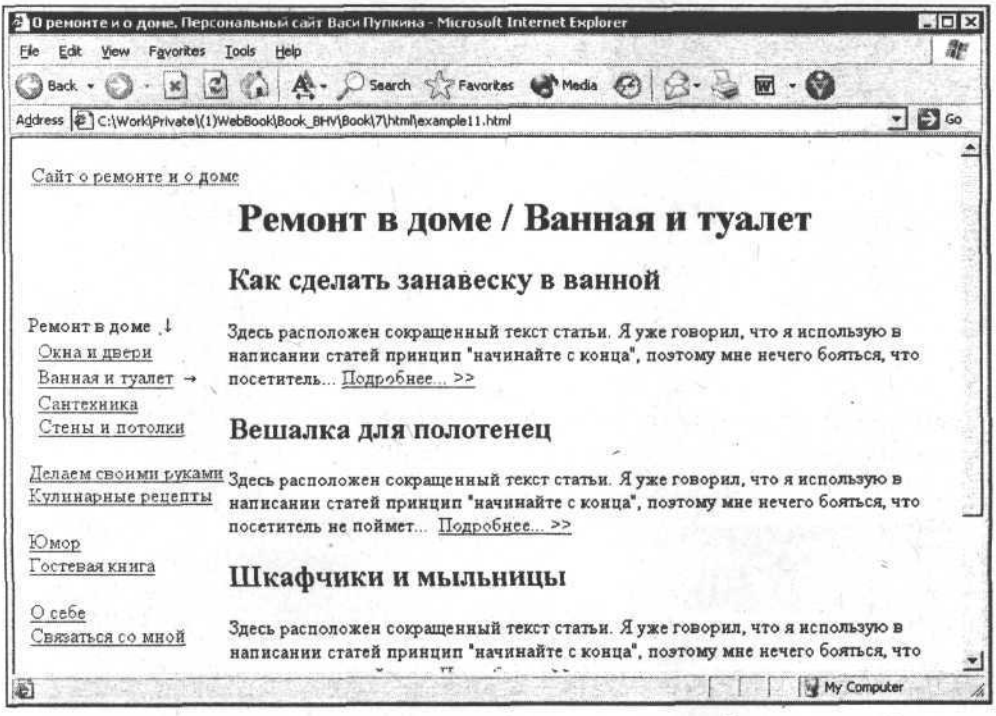


Рис. 7.7. Шаблон страницы макета с отключенными таблицами стилей

Видите разницу? Насколько меньше стало текста даже на таком небольшом фрагменте кода. Но все же имеются некоторые моменты, которые нельзя вынести в каскадные таблицы стилей (во всяком случае, без ущерба для размеров таблиц стилей). Например, ради одной единственной гиперссылки "Сайт о ремонте и о доме" переопределять стиль ссылок для класса (в данном случае — для четырех псевдоклассов) мне кажется просто нецелесообразным.

И, наконец, необходима проверка отображения страницы с отключенными таблицами стилей на тот случай, если вдруг произойдет сбой в загрузке стилей из внешнего файла (что, в общем-то, маловероятно, но случается). Как выглядит результат, можно увидеть на рис. 7.7. Чтобы сделать такую проверку, я всего лишь изменяю на время путь к файлу CSS в теге `<link>` с правильного на неправильный.

Глава 8

Типичные разделы сайта

Создаем раздел "Новости"

Начнем с того, что необходимо показывать в разделе новостей. Дело в том, что в некоторых случаях раздел новостей не имеет смысла. Например, если бы я создавал просто персональный сайт с историями о себе и своей семье, то в новостях мог бы давать информацию примерно такого типа: "я сходил в магазин" или "мы купили миксер". Кроме меня самого это никому не интересно. Если бы я создавал сайт для небольшой домашней сети, которую используют соседи по подъезду, то наоборот — раздел новостей был бы едва ли не самым посещаемым (у кого-то появился новый фильм, у кого-то — альбом любимого исполнителя, кто-то узнал, что отключат горячую воду).

На информационном сайте (вроде моего "О ремонте и доме") в разделе новостей можно помещать сведения, касающиеся обновления содержимого самого сайта (добавление новых статей и разделов, появление новых сервисных возможностей типа рассылки по электронной почте и т. п.). Новостей, конечно, будет не так много, но все же они будут.

Теперь о том, как будут выглядеть такие новости. Возможно несколько вариантов:

- первый — это просто список новостей (например, перечень статей подраздела) (рис. 8.1). В этом случае новости находятся в отдельном разделе, с ними можно познакомиться по желанию, они не мешают основному содержимому сайта;
- второй — это размещение раздела новостей (тоже в виде некоторого списка) таким образом, чтобы они были видны на каждой странице, например, где-нибудь в правой части (рис. 8.2). Такая полоса новостей всегда будет на виду. Однако не всегда имеет смысл загружать лишней информацией основные страницы;
- третий — это сменяющие друг друга новости, расположенные в каком-то одном месте и не занимающие много места, например, созданные по принципу бегущей строки или небольших рекламных блоков (рис. 8.3).

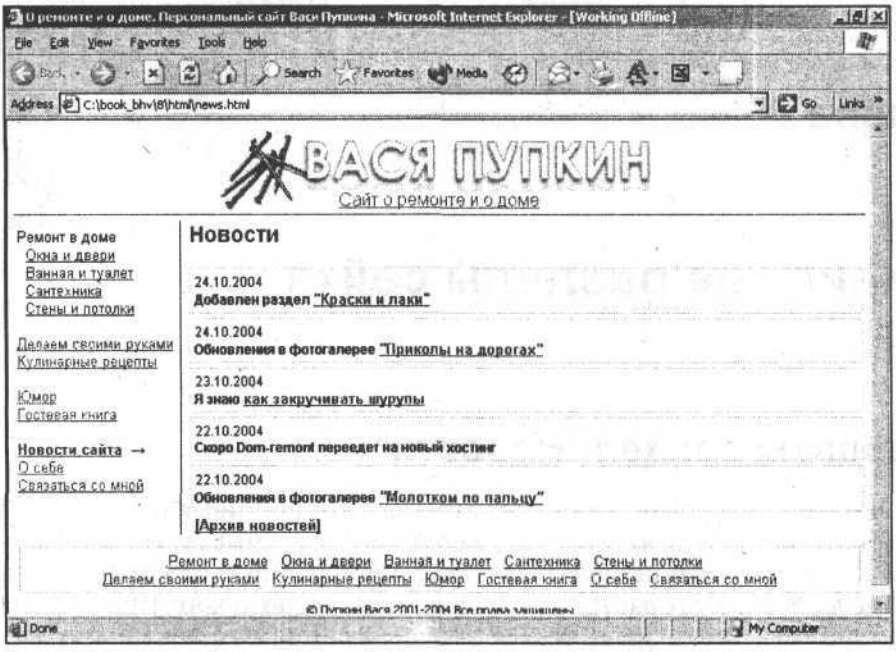


Рис. 8.1. Вариант размещения новостей на сайте в виде отдельного подраздела

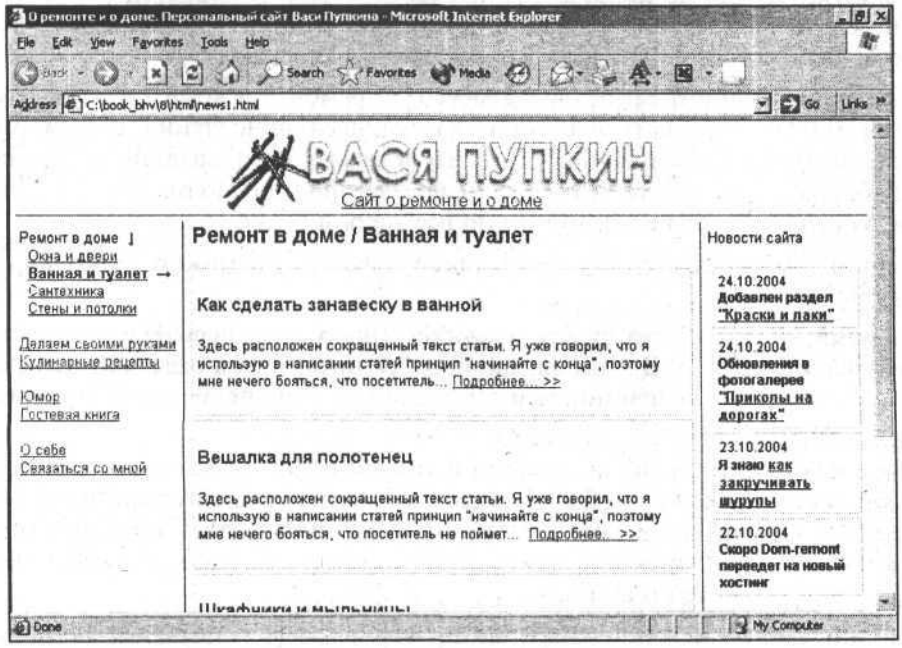


Рис. 8.2. Вариант размещения новостей на сайте в виде постоянно доступного меню с правой стороны

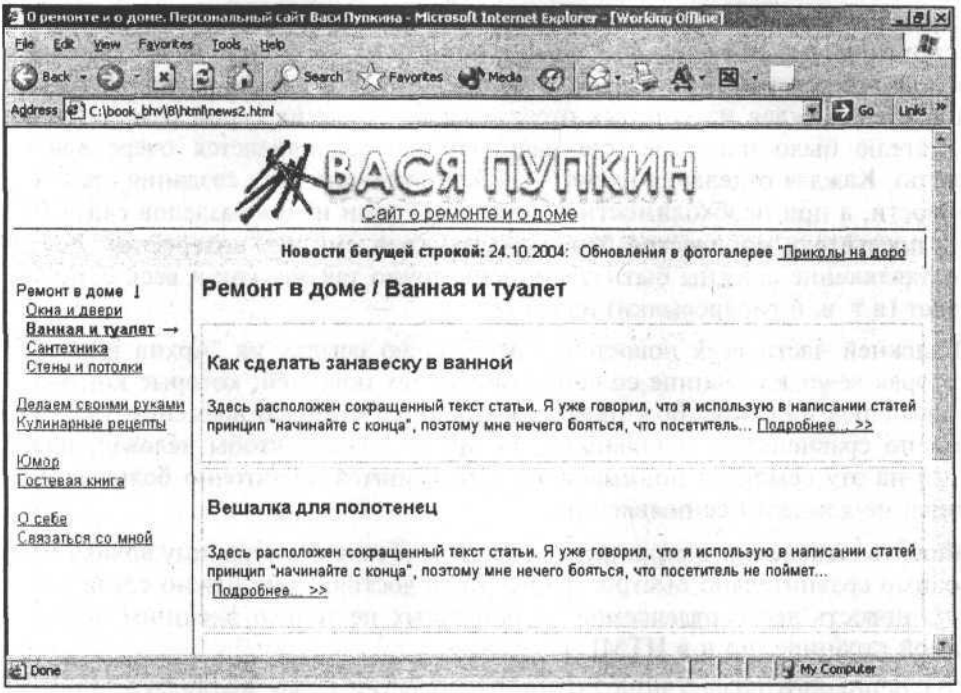


Рис. 8.3. Вариант размещения новостей на сайте в виде бегущей строки с меняющимися в цикле новостями

Несмотря на то, что третий вариант кажется интересным и необычным, у него есть некоторые недостатки. Все, что движется на страницах сайта (мигает, летает или прыгает), отвлекает внимание и уводит пользователя от основного содержимого, не давая ему сосредоточиться.

Большинство других вариантов организации новостей на сайте в основном похожи на какой-то из этих трех. Например, вместо бегущей строки можно использовать просто небольшой блок, в котором новости сменяют одна другую (прежняя исчезает, новая появляется на ее месте). Или, например, новости, организованные по принципу форума (это немного видоизмененный первый вариант с отдельным подразделом для новостей).

Я предпочитаю первый вариант. Он хорош тем, что не занимает лишнее место на всех страницах, не отвлекает от действительно важной для посетителя информации. Кроме того, эти новости можно поместить в отдельный файл. Слишком уж неблагодарное занятие обновлять все страницы, когда появилось что-то новое из новостей.

Теперь о самих новостях и о том, как я их организую.

Прежде всего, необходимо сразу же внести изменения в страницу-шаблон. Затем я добавляю в главное меню сайта раздел "Новости сайта" со ссылкой на файл news.html и создаю этот файл.

Страницу новостей, которая находится в файле news.html, я формирую на основании уже продуманной конструкции списков статей. В верхней части основного содержимого делаю заголовок "Новости", ниже располагаю набор новостей, каждая из которых отделяется от соседних серой рамкой (чтобы читателю было понятнее, где начинается и заканчивается очередная новость). Каждая отдельная новость должна включать дату создания, сам текст новости, а при необходимости и ссылки на один из подразделов сайта (чтобы посетитель мог быстро перейти туда, если ему это интересно). Все эти составляющие должны быть оформлены точно так же, как и весь остальной текст (в т. ч. и гиперссылки) на сайте.

В нижней части всех новостей я располагаю ссылку на "Архив новостей", которая ведет к странице со списком всех тех новостей, которые когда-либо появлялись на сайте. Возможно, эта страница будет со временем тяжеловатой по сравнению с остальными, но не настолько, чтобы человек, нажавший на эту ссылку и понимающий, что грузится достаточно большая страница, не дождался ее появления.

Вносить изменения в страницу новостей и обновлять страницу архива необходимо сравнительно быстро. Чтобы этого достичь, мне нужно сделать каждую новость легко отделяемой от остальных не только внешним видом на самой странице, но и в HTML-коде.

Код основного содержания страницы новостей будет выглядеть примерно таким образом, как представлено в листинге 8.1 (вставляется он между соответствующими комментариями):

Листинг 8.1

```
<!-- Начало основного содержимого страницы-->

<!-- Заголовок основного содержимого  страницы -->
<h1 class="header">Новости</h1>

<!-- Начало блока новостей-->

<div>
   24.10.2004<br>
   <b>Добавлен раздел <a href="../repair/paint/index.html">"Краски и
   лаки"</a></b>
</div>

<div>
    24.10.2004<br>
```

```
    <b>Обновления в фотогалерее <a href="../humour/gallery1.html">
    "Приколы на дорогах"</a></b>
</div>

<div>
    23.10.2004<br>
    <b>Я знаю <a href="../repair/window_door/howto.html">как закручивать
    шурупы</a></b>
</div>

<div>
    22.10.2004<br>
    <b>Скоро Dom-remont переедет на новый хостинг</b>
</div>

<div>
    22.10.2004<br>
    <b>Обновления в фотогалерее <a href="../humour/gallery3.html">
    Молотком по пальцу</a></b>
</div>

  <b><a href="news_arch.html">[Архив новостей]</a></b>
<!-- Конец блока новостей-->
<!-- Конец  основного содержимого страницы -->
```

Таким образом, чтобы безболезненно добавить или удалить какую-нибудь новость, мне достаточно убрать из кода все, что заключено в конкретный тег `<div>`, т. е. фрагмент кода, представленный в листинге 8.2.

Листинг 8.2

```
<div>
    22.10.2004<br>
    <b>Обновления в фотогалерее <a href="../humour/gallery3.html">
     Молотком по пальцу</a></b>
</div>
```

Кстати, даже готовить новости к следующему обновлению я могу именно в таком виде: не полная HTML-страница новостей, а просто текстовый

файл с набором тегов , содержащих какие-либо новости, после чего можно лишь копировать содержимое этого файла и вставлять в нужное место страницы новостей.

И наконец, нужно решить, как долго старые новости можно хранить на основной странице новостей и когда их необходимо переносить в "Архив новостей". Мне кажется целесообразным менять страницу с новостями раз в месяц. Окончательный вариант страницы новостей представлен на рис. 8.4.

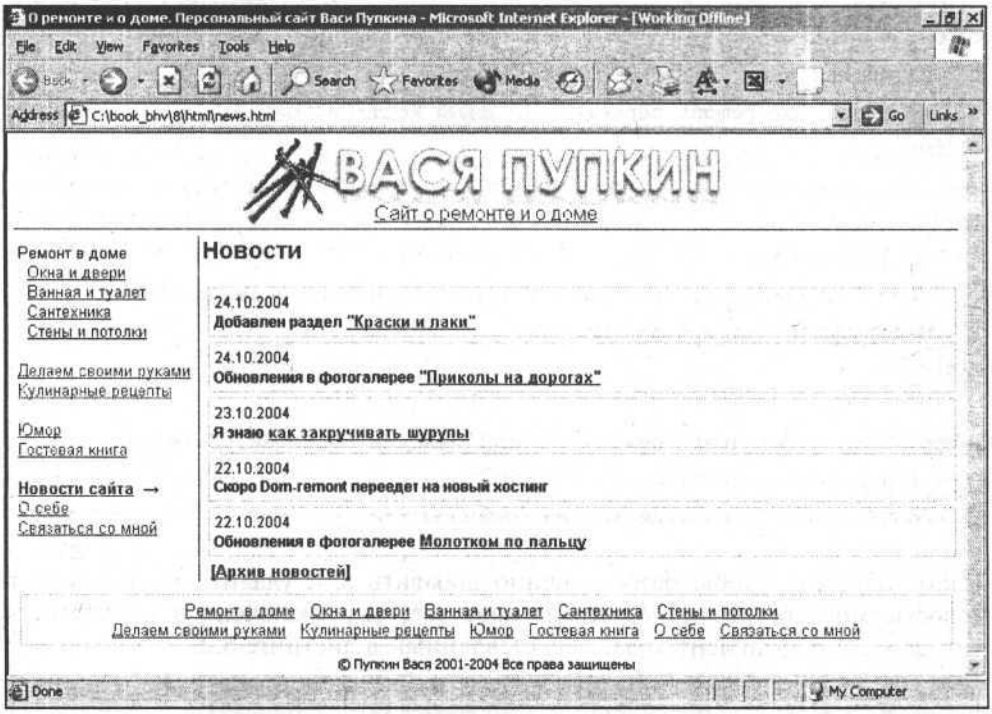


Рис. 8.4. Окончательный вариант внешнего вида страницы новостей моего сайта

Часто в Интернете предлагают использовать для своего сайта чужие новости. Например, какой-то известный и крупный сайт предоставляет сервис импорта его новостей. Происходит это примерно так: вы в определенном месте на своих страницах вставляете несколько строчек кода, которые при загрузке вашей страницы обращаются на сервер владельца этого крупного и известного сайта и получают оттуда список новостей. Новости отображаются на вашей странице. В следующий раз, когда новости обновятся на том сервере, они обновятся и на ваших страницах.

Создаем раздел "Моя гостевая книга"

Что такое гостевая книга? Вспомните "Книгу жалоб и предложений", которую вы наверняка видели в различных магазинах, ателье, парикмахерских. Гостевая книга сайта — это нечто подобное. В гостевой книге любой человек, попавший на сайт, может выразить свое мнение, попросить поменять структуру сайта, поблагодарить или покритиковать. Как правило, гостевая книга — это последовательный набор сообщений с указанием даты сообщения и некоторых сведений об авторе.

К сожалению, с гостевой книгой уже не обойдешься так просто, как с разделом новостей. Однако в Интернете существует много сайтов и компаний, которые предоставляют бесплатные гостевые книги. Например, я поначалу сделал свою гостевую книгу, воспользовавшись услугами сервера **www.narod.ru** компании "Яндекс". На **http://www.narod.ru** есть такой раздел, как "Мастерская", предназначенный для владельцев сайтов. Он позволяет использовать заранее заготовленные разработчиками компании "Яндекс" шаблоны и примеры. Одним из таких примеров является фотоальбом (мы уже рассматривали его), который можно создавать, вообще ничего не понимая в HTML. Создание гостевой книги тоже является одним из подобных сервисных моментов.

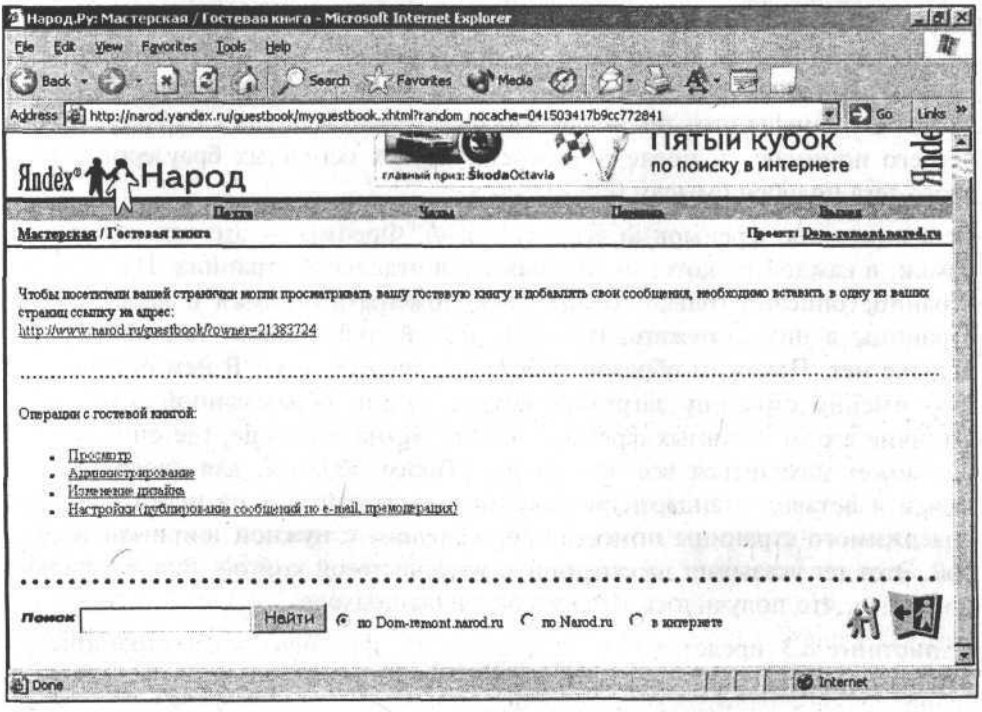


Рис. 8.5. Работа с гостевой книгой на **www.narod.ru**

При вызове сервиса создания гостевой книги и указании всех нужных сведений, вам выдается гиперссылка на страницу, на которой будет расположена ваша гостевая книга. Эта ссылка никоим образом не связана с именем вашего сайта, поэтому с такой гостевой книгой могут возникнуть небольшие проблемы. У вас есть два варианта присоединения такой страницы к вашему сайту.

Первый вариант напрашивается сам собой — надо открывать ссылку на гостевую книгу в новом окне браузера. Это делается при помощи атрибута `target="_blank"` у гиперссылки на вашу гостевую книгу. Ничего плохого в таком решении нет, если не считать того, что посетитель как бы отрывается от вашего сайта, а опытные пользователи Интернета могут еще и закрыть такое всплывающее окно, решив по привычке, что это очередной рекламный "pop-up".

Второй вариант предполагает открывать гиперссылку с гостевой книгой в том же окне. Тогда вернуться непосредственно к вашему сайту можно, всего лишь нажав в браузере кнопку **Назад**. И все же этот вариант еще хуже предыдущего, потому что неопытный пользователь просто потеряется и не поймет, куда делся сайт.

Я много думал над тем, какой же вариант выбрать, а потом нашел неплохой выход. Моя гостевая книга должна открываться практически так же, как и все другие страницы моего сайта, поэтому рядом с ней надо расположить и список основных разделов сайта. Это я могу сделать при помощи тега `<iframe>`. Раньше этот тег поддерживался только в Internet Explorer, но сейчас его понимают и последние версии других основных браузеров. Смысл этого тега подобен смыслу тега `<frame>`.

Вспомните про фреймовую верстку (*гл. 4*). Фреймы — это так называемые рамки, в каждой из которых отображается отдельная страница. На основной странице описано только то, как расположены эти рамки и какие именно страницы в них загружать. Никакой другой информации там практически больше нет. Похожим образом работает и тег `<iframe>`. В нем описано, какую именно страницу загрузить внутрь рамки, образованной этим тегом. Отличие его от обычных фреймов в том, что на странице, где описана рамка, может находиться все что угодно. Таким образом, для своей гостевой книги я оставил стандартную страницу моего сайта, а на место основного содержимого страницы поместил тег `<iframe>` с нужной шириной и высотой. Этот тег указывает на страницу с моей гостевой книгой. Рис. 8.6 иллюстрирует то, что получилось у меня при таком подходе.

В листинге 8.3 представлено, как выглядит фрагмент кода страницы для отображения гостевой книги внутри моего сайта.

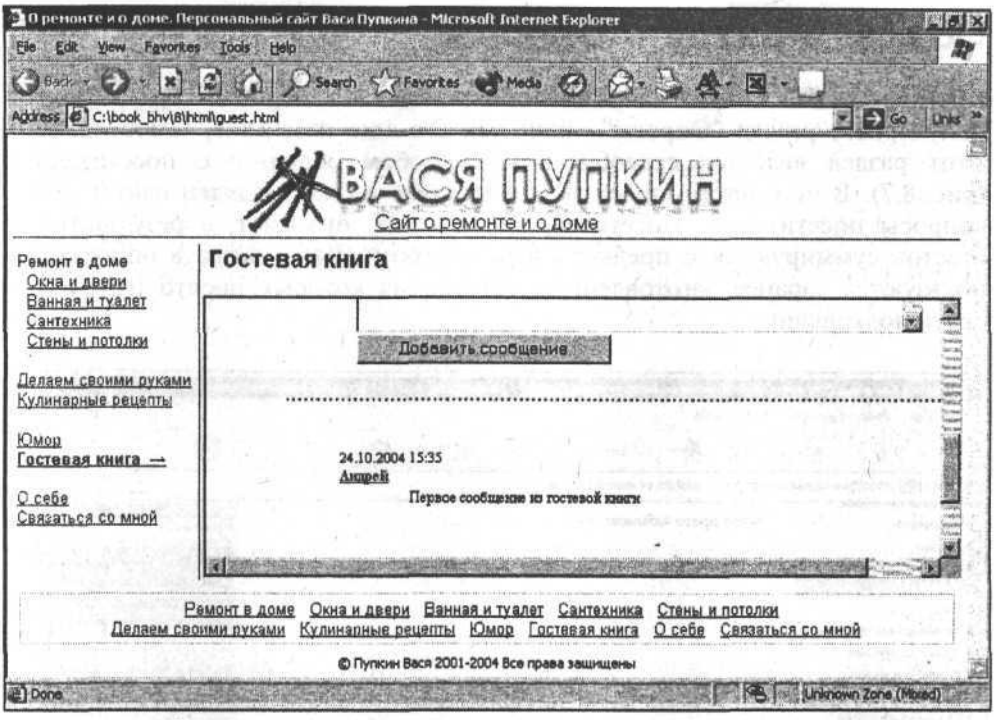


Рис. 8.6. Так будет выглядеть подключение моей гостевой книги (она пролистана к первому сообщению) на моем сайте на **narod.ru**

Листинг 8.3

```
<!-- Начало основного содержимого страницы-->
<!-- Заголовок основного содержимого  страницы -->
  <h1 class="header"> Гостевая книга</h1>
  <iframe src="http://www.narod.ru/guestbook/?owner=21383724"
              style="width:100%;height:85%">
  </iframe>
<!-- Конец  основного содержимого страницы -->
```

Как вы, наверное, обратили внимание, на сайте **www.narod.ru** можно не только просматривать свою гостевую книгу, но и администрировать ее (т. е. изменять/удалять ошибочные или нецензурные сообщения, можно изменять некоторые настройки и установки). Например, установив опцию уведомления вас по электронной почте при добавлении кем-то сообщения в книгу, возможно изменение интерфейса вашей гостевой книги.

Создаем раздел "Опросы"

Кроме гостевой книги, на различных сайтах часто присутствует довольно популярный раздел "Опросы" или, как его еще называют, "Голосование". Этот раздел является своеобразным способом общения с посетителями (рис. 8.7). В нем автор опроса (в моем случае он же владелец сайта) задает вопросы посетителям. Посетители, если хотят, отвечают, а результаты их ответов суммируются с предыдущими ответами. Чаще всего в опросах используются заранее заготовленные ответы, из которых просто нужно выбрать подходящий.

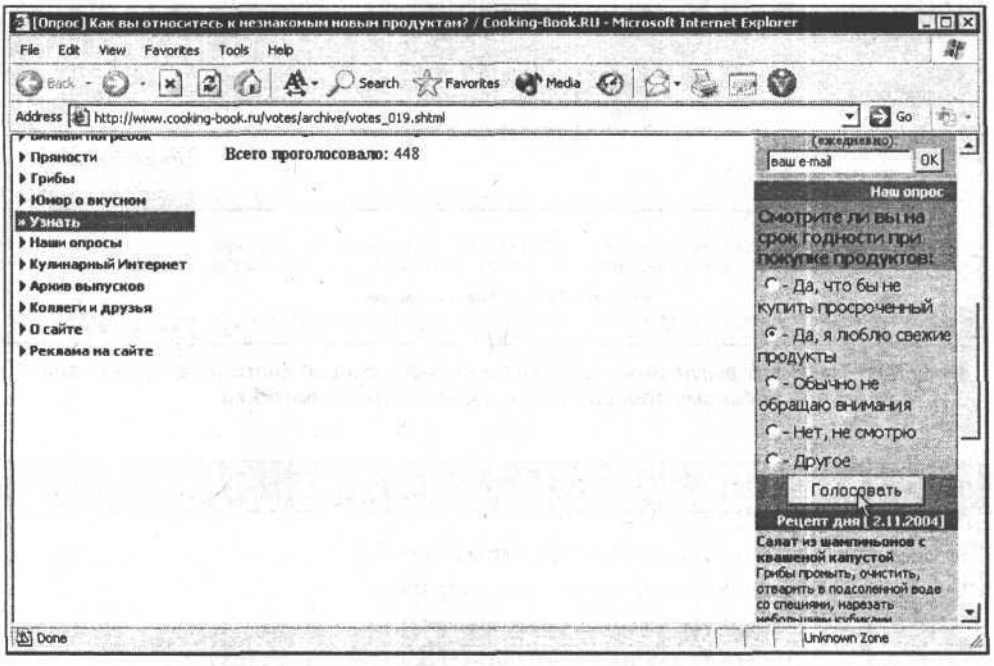


Рис. 8.7. Опрос на сайте "Кулинарная книга" (в правой части рисунка)

Посетителю интересно, как проголосовали другие, поэтому он может даже не голосуя, просто посмотреть на результаты опроса, а может увидеть их после того, как выбрал ответ и нажал на кнопку **OK** или **Голосовать**. Результаты опроса в большинстве случаев показываются в виде процентного соотношения, а также при помощи визуальных индикаторов.

Понятно, что продуктивно опросы и голосование можно проводить только на сайтах, которые часто посещаются и обновляются. Если вы еще не раскрутили свой сайт и его посещают 1–2 человека в неделю, то нормальных результатов опроса, по которым можно составить мнение о каких-либо предпочтениях среднестатистического пользователя, вам придется ждать годами.

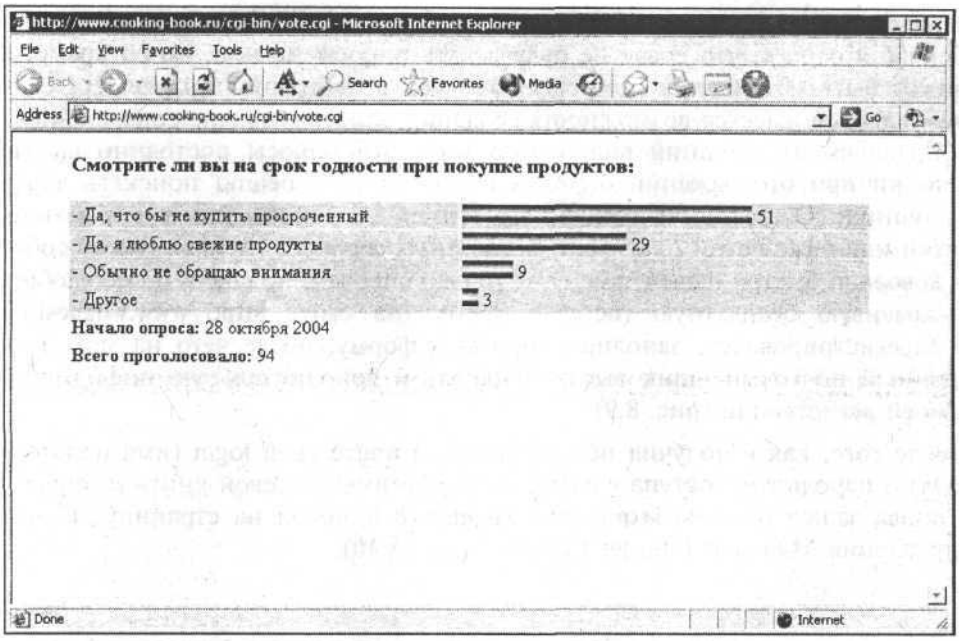


Рис. 8.8. Результаты опроса

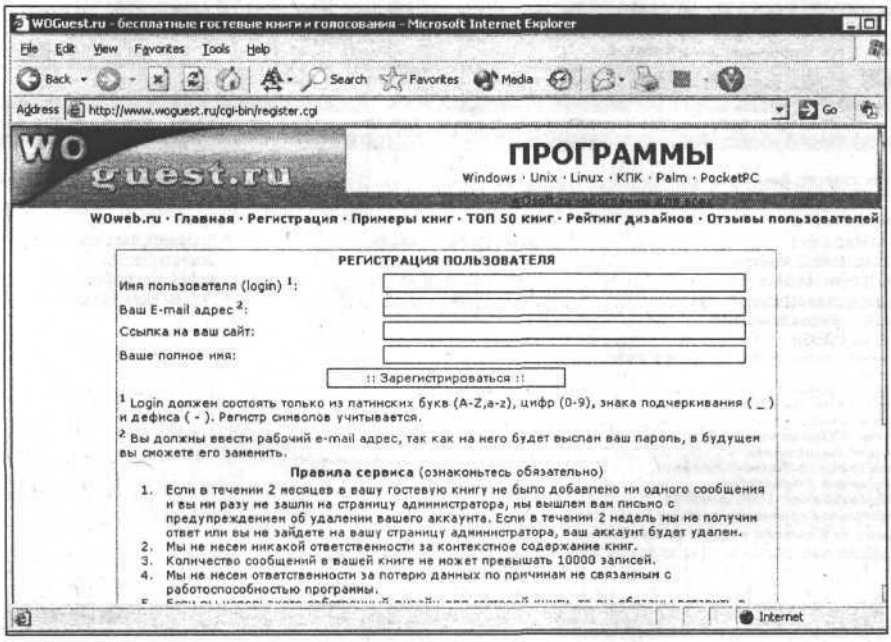


Рис. 8.9. Форма регистрации нового пользователя бесплатных гостевых книг и опросов на сайте **www.woguest.ru**

Я надеюсь, что мой сайт будут посещать примерно 20–30 человек в неделю. Сейчас я решил, что сразу не буду делать опросы на нем, но со временем может быть и придется, поэтому надо все подготовить заранее. На сайте **www.narod.ru** имеется возможность создания опросов, но мне совершенно не понравился их внешний вид, кроме того, эти опросы постоянно выдают ошибки при отображении страниц результатов. Я решил поискать другие источники. Особенность нынешнего Интернета такова, что в нем есть десятки или даже сотни платных и бесплатных сервисов, схожих между собой. Я довольно быстро нашел замену не только опросам, но еще и более удобную и красивую бесплатную гостевую книгу (на сайте **http://www.woguest.ru**). Я зарегистрировался, заполнив короткую форму, после чего на мой электронный почтовый ящик выслали пароль и дополнительную информацию о моей регистрации (рис. 8.9).

После того, как я получил по электронной почте свой login (имя пользователя) и пароль для доступа к администрированию гостевой книги и опросов, я снова зашел на сайт **http://www.woguest.ru** и попал на страницу "Панель управления Microsoft Internet Explorer" (рис. 8.10).

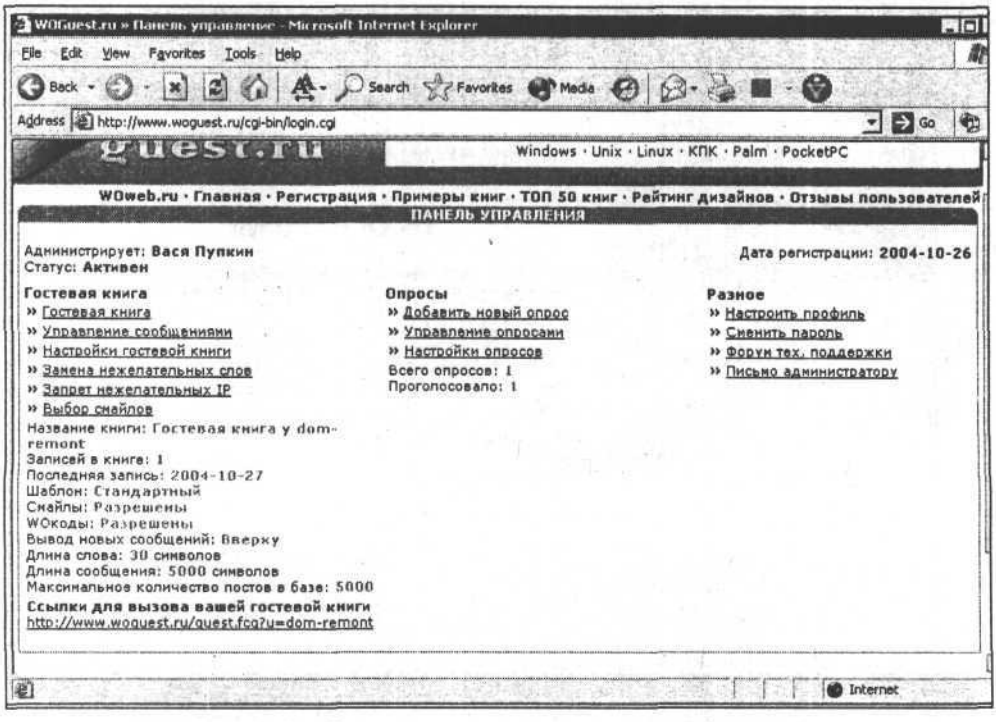


Рис. 8.10. Страница для управления гостевой книгой и создания опросов и голосований

Как видите, сервис по управлению гостевой книгой и опросами богат и многообразен. Опрос состоит из самого вопроса, набора вариантов ответа на него и результатов. Результаты формируются автоматически, а вопрос и варианты ответов задает автор сайта. Форма добавления нового опроса показана на рис. 8.11.

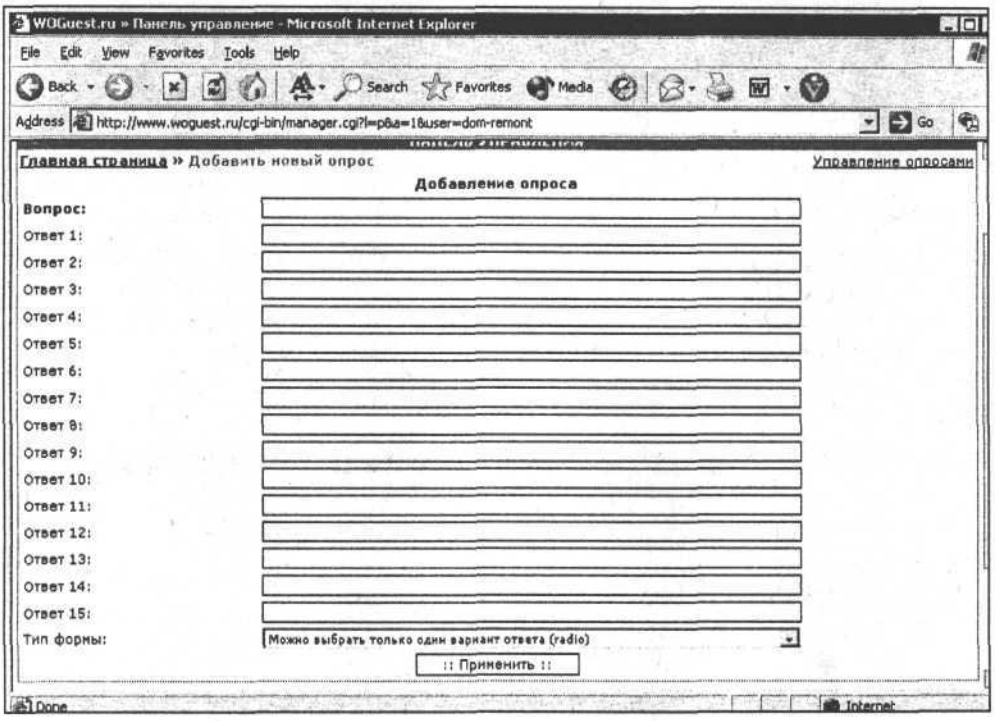


Рис. 8.11. Форма добавления нового опроса

Глава 9

JavaScript и нестандартные элементы сайта

Что такое JavaScript и как он работает?

JavaScript — это своеобразный язык программирования, который можно использовать прямо на HTML-страницах. Для Всемирной паутины разработано немало языков программирования, их часто называют скриптами или скриптовыми языками. К таким относятся языки PHP, ASP, Perl, JSP и т. д. (они используются, когда нужно создать сложный, объемный сайт или интернет-магазин или еще что-то подобное). Такие языки требуют специального программного обеспечения, установленного на сервере. JavaScript (в отличие от перечисленных языков) такого программного обеспечения не требует и поэтому его иногда даже называют расширением HTML.

На web-странице JavaScript подключается практически в любом месте, где это необходимо, при помощи тега `<script>`, внутри которого либо указывается имя внешнего файла с программой, написанной на JavaScript, либо сама JavaScript-программа. Если программа находится во внешнем файле, то у этого файла должно быть расширение js (например, menu.js). Однако чаще всего функции JavaScript помещают прямо в код HTML-страницы. В листинге 9.1 представлен фрагмент HTML-кода, показывающий пример включения JavaScript в web-страницу.

Листинг 9.1

```
...
<title>О ремонте и о доме. Персональный сайт Васи Пупкина</title>
<link type="text/css" rel="stylesheet" href="style.css">
</head>
 <script language="JavaScript">
 <!--
function hideMenu()
```

```
{
document.getElementById(curent_element).style.visibility = 'hidden';
}
//-->
</script>
<body>
...
```

Как и многие другие языки программирования, конструкции JavaScript позволяют создавать ветвления, циклы, математические операции и многое другое. Однако основной идеей этого языка является то, что он взаимодействует с объектами web-страницы и браузера, некоторым образом изменяя их состояние и свойства.

В листинге 9.2 приведен простой пример работы JavaScript, где используются вымышленные русские названия тегов, атрибутов, объектов и т. п.

Листинг 9.2

```
...
<Здесь начало JavaScript>
функция Поменять_цвет_для (Идентификатор){
   Документ.Элемент[Идентификатор].Цвет  =  красный;
}
 </Здесь конец JavaScript>

...
<Квадрат Идентификатор=1 Цвет=Зеленый  Если_Кликнут=Поменять_цвет_для(1)>
    Текст внутри квадрата
</Квадрат>
...
```

Как такой скрипт может работать? Все просто. При помощи самого JavaScript создается функция, которая меняет цвет конкретного элемента страницы, используя его идентификатор. Сама по себе эта функция ничего не делает. Она начинает работать только тогда, когда ее вызывают из HTML-кода и передают ей значение идентификатора элемента. То есть внутри тега `<квадрат>` (в случае, если пользователь "кликает" мышью на этом элементе на HTML-странице) вызывается функция `Поменять_цвет_для (1)` (где `1` — это идентификатор элемента для замены цвета). Для обработки берется элемент с нужным идентификатором и его свойству `цвет` задается значение `красный`.

Приведу еще несколько примеров JavaScript, которые я использовал на своем сайте, а вы уже сами решите — хотите ли изучать этот язык программирования или вам достаточно просто знать, что он есть. А вообще-то, чтобы использовать на своих страницах JavaScript, чаще всего достаточно просто понимать общий смысл его работы и иметь готовый пример.

Делаем простое выпадающее меню

Как вариант организации навигационного элемента я решил попробовать сделать выпадающее меню. Я еще не знаю, понравится ли мне такой вариант, но ведь для того, чтобы это понять, нужно попробовать.

Чтобы сделать заготовку такого меню и при этом не отвлекаться на остальные элементы страницы-шаблона, я решил ставить свои эксперименты на отдельной HTML-странице.

Смысл выпадающего меню в том, что при наведении курсора мыши на какой-либо элемент (например, основной раздел) появляется список подразделов. При перемещении курсора за пределы этого элемента, список исчезает. Выпадающее меню позволяет сэкономить место на странице (правда,

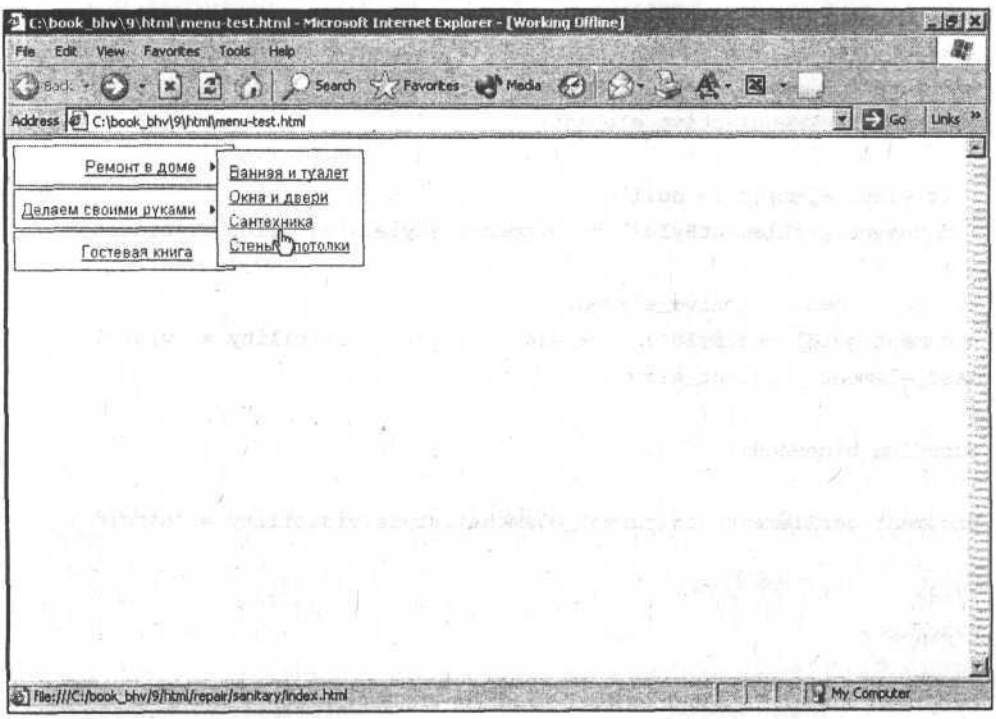


Рис. 9.1. Внешний вид фрагмента выпадающего меню

при этом же оно некоторым образом скрывает от пользователя структуру сайта). Такое меню используется в большинстве программ, практически во всех операционных системах (только там оно функционирует при щелчке мыши, а не при наведении курсора). Так или иначе, но для web-страниц это не совсем стандартный элемент, поэтому необходимо дополнительно сообщать пользователям о том, что это не просто ссылка, а именно выпадающее меню. Для этого достаточно рядом с каждым пунктом выпадающего меню изобразить стрелку (тогда пользователь будет знать, что если подвести к ней курсор или "кликнуть" на ней, то появится выпадающее меню). Понятно, что там, где нет выпадающих подразделов, такая стрелка не нужна.

На рис. 9.1 показан внешний вид фрагмента выпадающего меню. Код, реализующий это выпадающее меню, представлен в листинге 9.2.

Листинг 9.2

```
<html>
<meta content="text/html; charset=windows-1251" http-equiv="content-type">
<head>

<script language="JavaScript">
<!--
var curent_element, last_element;
function showMenu(active_element)
{
  if (last_element != null){
  document.getElementById(last_element).style.visibility = 'hidden';
  }
curent_element = active_element;
document.getElementById(active_element).style.visibility = 'visible';
last_element = curent_element;
}
function hideMenu()
{
document.getElementById(curent_element).style.visibility = 'hidden';
}
//-->
</script>

<style>
A {
```

```
font-size:10pt; color:#4e4593; font-family:arial; text-
decoration:underline;
}
.expand{
border:1px solid #345678; vertical-align:top; background-color:white;
padding:6px; text-align:right;
}
</style>

</head>
<body style="margin:3px">

<!-- Таблица с основными разделами -->
<table border="0" align="left">
   <tr>
     <td onmouseover="showMenu('repair')" onmouseout="hideMenu()"
         class="expand">
         <a href="#">Ремонт в доме</a>
           <img src="img/arrow.gif">  </td>
   </tr>
   <tr>
     <td onmouseover="showMenu('handmade')" onmouseout="hideMenu()"
         class="expand">
         <a href="#">Делаем своими руками</a>
           <img src="img/arrow.gif">  </td>
   </tr>
   <tr>
     <td class="expand">
         <a href="guest_book.html">Гостевая книга</a>
               </td>
   </tr>

</table>
<!-- Конец таблицы с основными разделами -->

<!-- Выпадающие меню - списки подразделов-->
<div id="repair" class="expand"
      onmouseover="showMenu('repair')"
      onmouseout="hideMenu()"
```

```
      style="position:absolute; width:120px; height:20px;
      z-index:1; visibility: hidden;
      top:9px; left: 170px">
<table width="100%" height="76" bgcolor="white">
 <tr><td nowrap>
     <a href="repair/bath_toilet/index.html">Ванная и туалет</a>
</td></tr>
<tr><td nowrap>
     <a href="repair/window_door/index.html">Окна и двери</a>
</td></tr>
<tr><td nowrap>
    <a href="repair/sanitary/index.html">Сантехника</a>
</td></tr>
<tr><td nowrap>
    <a href="repair/wall_floor/index.html">Стены и потолки</a>
</td></tr>
</table>
</div>

<div id="handmade" class="expand"
     onmouseover="showMenu('handmade')"
     onmouseout="hideMenu()"
     style="position:absolute; width:120px; height:20px;
     z-index:1; visibility: hidden;
     top: 44px; left: 170px">
<table width="100%" height="76" bgcolor="white">
<tr><td nowrap>
     <a href="hand_made/wood/index.html">Изделия из дерева</a>
</td></tr>
<tr><td nowrap>
    <a href="hand_made/iron/index.html">Изделия из металла</a>
</td></tr>
<tr><td nowrap>
    <a href="hand_made/build/index.html">Строительство </a>
</td></tr>
</table>
</div>
<!-- Конец выпадающих меню - списков подразделов-->
</body>
</html>
```

Вся конструкция состоит как бы из трех частей. Первая часть — сам JavaScript, вторая — таблица с главными разделами сайта, которая видна всегда, и третья — блок с подразделами, появляющийся на экране только после наведения курсора на соответствующий пункт основного меню.

Рассмотрим, каким образом достигается эффект выпадающего меню. Блок с основными разделами всегда виден на странице. В JavaScript есть такое понятие, как событие (т. е. некоторое действие пользователя на странице). В данном случае при наведении пользователем курсора на ячейки таблицы с основными разделами наступает событие `onmouseover` (наведение курсора мыши на данный элемент). Когда курсор уводится за пределы элемента, происходит событие `onmouseout`. В первом случае вызывается функция `showMenu (active_element)` (меню_подразделов), отвечающая за отображение конкретной части выпадающего меню с подразделами. Во втором случае вызывается функция `hideMenu()`, скрывающая меню подразделов. Рассмотрим теперь подробнее сам скрипт (листинг 9.3).

Листинг 9.3

```
<script language="JavaScript">
<!--
var curent_element, last_element;
function showMenu(active_element)
{
  if (last_element != null){
  document.getElementById(last_element).style.visibility = 'hidden';
  }
curent_element = active_element;
document.getElementById(active_element).style.visibility = 'visible';
last_element = curent_element;
}
function hideMenu()
{
document.getElementById(curent_element).style.visibility = 'hidden';
}
//-->
</script>
```

Сначала мы объявляем переменные `curent_element` (текущий элемент) и `last_element` (последний элемент), которые используются при работе с выпадающими меню подразделов.

Затем описываем функцию `showMenu(active_element)`, параметром которой является передаваемое ей значение переменной `active_element` (текущий элемент). Значение это передается в скрипт при событии наведения курсора мыши на ячейку основного меню. Например, в коде страницы есть запись `onmouseover="showMenu('repair')"`, которая означает, что при наведении курсора на тот элемент, к которому эта запись относится, следует вызвать функцию `showMenu` и передать ей в качестве значения переменной `active_element` строку `'repair'`.

Далее функция `showMenu` анализирует: если последний показанный элемент выпадающего меню существует (т. е. значение переменной `last_element` не равно 0), то необходимо получить этот элемент по его уникальному идентификатору (атрибут `"id"`) и присвоить ему стиль `hidden` (невидимый). Эта проверка делается для того, чтобы не оказалось два одновременно выпадающих элемента.

Далее переменной `curent_element` присваиваем значение `active_element` (т. е. делаем активный элемент текущим), потом присваиваем этому элементу стиль `visible` (видимый) и после этого переменной `last_element` (последний элемент) присваиваем значение текущего.

Таким образом, мы узнали, на какой элемент указал курсор мыши, сделали этот элемент текущим, изменили его свойства и после успешного изменения свойств стали считать его последним.

При перемещении курсора за пределы пункта меню основных разделов, при помощи функции `hideMenu()` мы полученному ранее текущему элементу выпадающего меню (который виден на экране) присваиваем свойство `hidden` (невидимый).

Делаем простое "дерево"

Под "деревом" понимается древовидная структура, пример которой можно видеть в любой современной операционной системе. Это отображение файловой структуры каталогов, которые могут сворачиваться и разворачиваться, показывая внутри себя другие каталоги и файлы. Иногда в таком виде организуют структуру разделов сайта. Но бывает, что разработчики используют сложные и непонятные алгоритмы создания древовидной структуры и ее отображения. Зачастую "дерево" сделано так, что при разворачивании какого-то элемента перечитывается вся страница и приходится подолгу ждать результата.

Я хочу попробовать оформить структуру основных разделов сайта и их подразделов не только как статичную структуру или выпадающее меню, но и в виде "дерева" (рис. 9.2, листинг 9.4). Мне нужно подобрать необходимые картинки для раскрытых, свернутых, а также для конечных подразделов и написать небольшую HTML-страницу со скриптом.

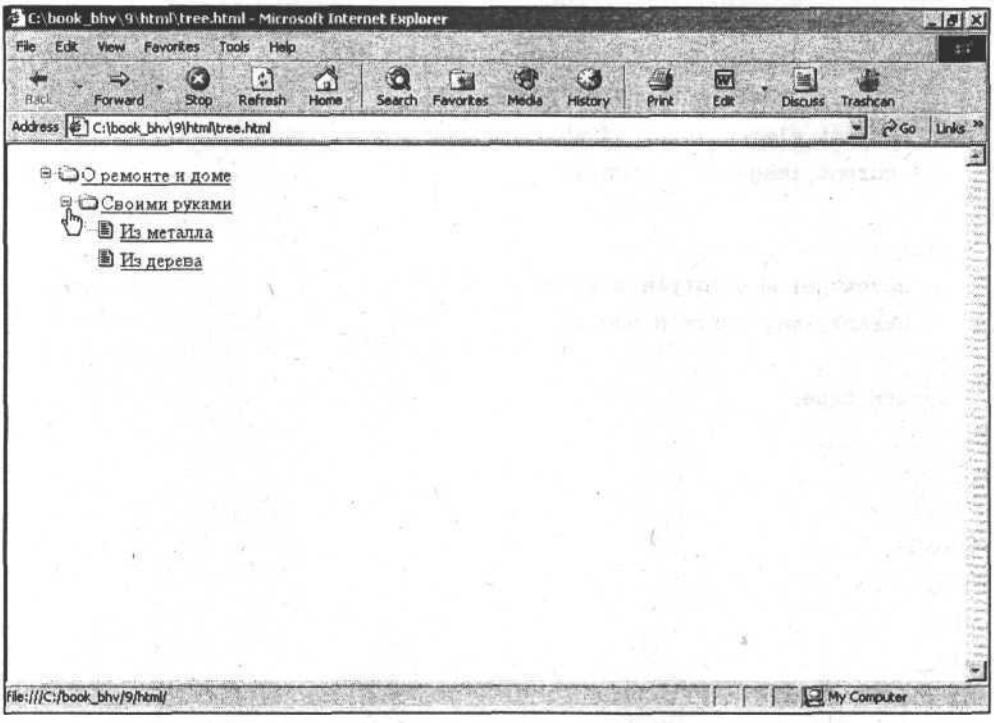


Рис. 9.2. Древовидная структура на HTML-странице

Листинг 9.4

```
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=windows-
1251">
<style>
  ul {margin-left : 16px; margin-top:4px; list-style: none;}
  li {margin-top:4px}
  img {border: 0;}
</style>
<script>
<!--
function openTree(id)
{
   var curent_element = document.getElementById("child"+id);
   var curent_image= document.getElementById("img"+id);
```

```
    if( !curent_element ) return false;
    if( curent_element.style.display == "none"
        || curent_element.style.display == "" ) {
        curent_element.style.display = "block";
        curent_image.src='open.gif';
    }
    else {
       curent_element.style.display = "none";
       curent_image.src='close.gif';
    }
    return true;
}
-->
</script>
</head>
<body>
<div>
<ul>
<li class="folder">
  <a onclick="return !openTree('1_1');" href="">
  <img id="img1_1" src="close.gif" align="absmiddle">О ремонте и доме</a>
</li>
<div id="child1_1" style="display: none;">
  <ul>
  <li>
  <a onclick="return !openTree('1_1_1');" href="">
  <img id="img1_1_1" src="close.gif" align="absmiddle">Своими руками
  </a>
  </li>
  <div id="child1_1_1" style="display: none;">
    <ul>
      <li class="list">
      <a href="">
      <img id="img1_1_1" src="list.gif" align="absmiddle">Из металла
      </a>
      </li>
      <li class="list">
      <a href="">
```

```
      <img id="img1_1_1" src="list.gif" align="absmiddle">Из дерева
      </a>
      </li>
    </ul>
  </div>
  </ul>
</div>
</ul>
</div>
</body>
</html>
```

Этот скрипт работает достаточно просто. У меня есть некоторый элемент, который является гиперссылкой. Этот элемент показан вместе с картинкой открытой или раскрытой папки. Когда пользователь щелкает на нем мышью, то происходит вызов функции (открыть "дерево"), в нее передается идентификатор элемента (текущего), который нужно показать (открыть). Также изменяется текущая картинка для элемента, на котором щелкнул пользователь. Для элемента, который является текущим, производится проверка его состояния (если он в данный момент показан, то после щелчка мыши исчезает и наоборот).

Меняем картинку при наведении курсора мыши

Часто возникает необходимость менять (или добавлять) картинку при наведении курсора мыши, например, на гиперссылку. В листинге 9.5 представлен скрипт, который делает это достаточно легко. Я привожу здесь код страницы, которая содержит только то, что нужно для работы скрипта, а потом этот код можно будет подключить к любой странице.

Листинг 9.5

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
<title>О ремонте и о доме. Персональный сайт Васи Пупкина</title>
<script language="JavaScript">
```

```
<!--
  function select(hover_obj){
  document[hover_obj].hover_img=document[hover_obj].src;
  document[hover_obj].src=document[hover_obj].lowsrc;
 }
 function unselect(hover_obj){
  document[hover_obj].src=document[hover_obj].hover_img;
 }
//-->
</script>
</head>
<body>
 <table border="0">
    <tr>
    <td>
        <a href="index.html"
           onMouseOver="select('refresh')";
           onMouseOut="unselect('refresh');">
           <img name="refresh" border="0" align="absmiddle"
                src="img/refresh_bw.jpg"
                lowsrc="img/refresh_color.jpg"
                alt="Обновить данные">
                Обновить данные
        </a>
    </td>
    </tr>
 </table>
</body>
</html>
```

В данном примере используется существующая в HTML возможность присваивать тегу `<img>` две различные картинки. Это делается при помощи атрибутов `src` (для обычной картинки) и `lowsrc` (для картинки с низким качеством), которые служат для предварительной загрузки картинки. Таким образом, в зависимости от события `onMouseOver` (курсор мыши наведен на элемент) или `onMouseOut` (курсор мыши вне элемента), подключается картинка то из атрибута `src`, то из `lowsrc`.

Как находить и использовать бесплатные программы на JavaScript

JavaScript язык достаточно простой, а фантазия у людей развита прилично, поэтому в Интернете содержится огромнейшее количество информации и по самому JavaScript и по примерам его использования, но не всегда эти примеры могут служить образцом для подражания.

Обычно для того, чтобы найти примеры использования JavaScript для какой-то конкретной ситуации, достаточно набрать в строке поиска на поисковом сервере что-нибудь вроде "`выпадающее меню JavaScript`" или "`бесплатные скрипты`".

Но прежде, чем встраивать найденный скрипт в свою страницу, подумайте хорошенько, а действительно ли он вам так необходим. Почему я говорю об использовании только необходимых скриптов? Да потому что слишком много расплодилось сейчас чересчур "навороченных" скриптов, которые, кроме утяжеления страницы, не приносят никакой пользы.

Глава 10

Тестирование сайта

Простые способы тестирования

Тестирование — это контроль некоторой системы (например, web-сайта) с целью выявления ошибок и проверки определенных показателей.

Поскольку тестирование очень важный процесс в производстве компьютерных систем, то со временем он превратился в настоящую науку; появились огромные фирмы, занимающиеся разработкой программного обеспечения для тестирования. Такие программы позволяют специалистам автоматизировать свою работу, регистрировать в специальных системах найденные ошибки, заранее придумывать и записывать тесты и т. д.

Существует понятие автоматизированного и ручного тестирования. Для автоматизированного тестирования необходима специальная программа. Для ручного тестирования никакая дополнительная программа не нужна, здесь все зависит от человека, который запускает тестируемую систему и начинает с ней работать, проверяя, все ли в ней в порядке.

Автоматизированное тестирование используют лишь для монотонных, достаточно простых действий или, например, для имитации одновременной работы с системой большого количества пользователей. Программы для автоматизированного тестирования не всегда могут заменить человека и его восприятие системы, они не видят грамматических неточностей, неудачных сочетаний цветов, не могут по ошибке случайно закрыть окно с программой во время ее активной работы и т. д.

Лично я использую для тестирования своего сайта простенькую бесплатную программу, которую когда-то случайно обнаружил, путешествуя по Интернету (о ее возможностях можно почитать на английском языке по адресу **http://home.snafu.de/tilman/xenulink.html** и скачать с адреса **http://home.snafu.de/tilman/XENU.ZIP**). Эта программа называется Xenu's Link Sleuth и служит в основном для проверки и поиска "битых" гиперссылок (т. е. таких, которые указывают на несуществующие файлы). Например, если я забыл про

какую-то HTML-страницу и не положил ее вместе с остальными или просто ошибся в имени файла в атрибуте "href" гиперссылки, то такая программа быстренько отыщет, какие ссылки у меня не работают и почему. Согласитесь, это гораздо удобнее, чем вручную проверять все гиперссылки на всех страницах. Также эта программа собирает и анализирует некоторые полезные сведения о сайте.

Прежде чем рассказывать о простых способах, которыми я могу быстро протестировать свой сайт, хочу заострить внимание на одном немаловажном факте. Начинать тестирование сайта (впрочем, как и любой другой программы вообще) лучше с самого начала. Когда вы напишете страницу-шаблон, необходимо провести ее быстрое тестирование. После внесения в работу сайта каких-либо серьезных изменений необходимо снова протестировать его. Таким образом вы сможете сэкономить время и нервы, а ваш сайт будет работать надежно и правильно.

Существуют пять простых способов тестирования:

- общая поверхностная проверка работы сайта на вашем компьютере. Дело в том, что в любом случае вам когда-нибудь придется разместить свой сайт в Интернете, и только после этого сможете качественно протестировать такие его показатели, как скорость загрузки страниц и графики, корректность определения кодировки ваших страниц, работу внешних разделов сайта (таких как "Гостевая книга", "Опросы и голосование"). Также вам придется снова проверить, не появились ли на сайте "битые" гиперссылки, ведь вы могли пропустить какой-нибудь файл во время загрузки сайта со своего компьютера в Интернет, или мог произойти какой-то сбой, в результате чего страница загрузилась не полностью. Возможна и такая ситуация, что какие-то настройки на интернет-сервере, на который вы загрузили свои страницы, мешают сайту нормально работать и т. д. Все эти проверки в любом случае придется сделать, но прежде многие моменты необходимо проверить прямо на вашем компьютере, возможно даже не имея никакой связи с Интернетом;
- внимательный просмотр внешнего вида всех до единой страниц, при котором вы сможете увидеть неожиданно возникшую проблему, например, с кодировкой или отображением какой-то картинки (если вы спутали жестко заданные размеры картинки и она стала сплюснутой) или, возможно, исчез какой-то шрифт, есть ошибка в указании цветов элементов и т. д. Существует множество примитивных ошибок, которые можно легко заметить, просто просмотрев весь сайт;
- проверка сайта при отключенной графике и таблицы стилей. Это можно сделать, временно переименовав файл с таблицами стилей и каталог с картинками либо отключив поддержку этих элементов в настройках браузера (если вы это умеете). Тогда обращение к этим частям сайта в HTML-коде будет ошибочным (только не забудьте все вернуть на место).

Если ваш сайт будет выглядеть достаточно нормально (т. е. изменится только шрифт, цвет, границы, станут пустыми места под картинки), но сама структура и содержимое останутся на месте, значит, он выдержал такое испытание, иначе вам необходимо браться за исправление всех ваших страниц;

- просмотр сайта в основных популярных на данный момент браузерах (Internet Explorer, Mozilla и Opera). Даже если учесть то обстоятельство, что, по статистике, наиболее часто пользователи работают в Internet Explorer 6.0, нельзя игнорировать остальные браузеры только потому, что вам лень проверить работу сайта, например в браузере Mozilla;
- проверка работы вашего сайта на различных разрешениях монитора (надо воспользоваться настройками рабочего стола). На данный момент наиболее частыми являются разрешения монитора 800×600 и 1024×768 пикселов. Дело в том, что у пользователей может встречаться и то, и другое разрешение.

Быстрое общее тестирование web-сайта

После проверки работы сайта при помощи простейших способов тестирования, необходимо провести общее тестирование, которое также включает некоторые правила:

- если вы используете JavaScript, то необходимо включить в вашем браузере так называемую отладку сценариев. Это одна из настроек свойств браузера, позволяющая видеть, когда в работе JavaScript происходят сбои. В случае этих сбоев на каждую ошибку будет выдаваться диалоговое окно, где будет высвечиваться причина ошибки. Если вы не включите эту опцию, то вполне возможно, что не заметите ошибок, которые могут возникнуть у пользователя. А пользователь, столкнувшийся с такими сайтами, изобилующими ошибками JavaScript, как правило, навсегда уходит с его страниц;
- обязательно проверять все функции, которые вы предоставляете пользователям (от отправки формы по e-mail до вывода на принтер). Проверьте, действительно ли отправляется почта, действительно ли скачиваются файлы, которые вы предлагаете посетителям, добавляются ли сообщения в гостевую книгу и т. д.;
- если вы вдруг решили использовать на своем сайте аудио- или видеооформление, обязательно проверьте его работу на нескольких компьютерах (хотя бы у своих друзей). Часто бывает, что из-за различий в настройках и конфигурации аудио- и видеоустройств на разных компьютерах возникают непредвиденные проблемы;
- просмотрите внимательно все страницы, не нарушилось ли что-нибудь в той части кода, которая принадлежит шаблону. Это может произойти

случайно во время разработки. Стиль оформления страниц должен быть, по возможности, одинаковым на всех страницах;

- обязательным условием является проверка работы сайта в "боевых условиях", т. е. уже после загрузки на сервер в Интернете. Дело в том, что даже большие тяжеловесные страницы сайта на вашем домашнем компьютере будут загружаться в браузер почти мгновенно. Однако стоит им попасть в Интернет, как скорость загрузки может упасть в десятки раз. Если окажется, что ваши страницы грузятся очень долго, то придется переделывать весь сайт или, как минимум, самые большие по размеру страницы, в противном случае ваш сайт никогда не станет популярным. Есть один универсальный прием для оценки не только скорости загрузки страниц, но и их оформления — посмотрите, как это выглядит у конкурентов (т. е. у сайтов, подобных вашему по тематике). Словом, оцените разницу, хотя бы просто на глаз;
- проведите так называемое тестирование по принципу "белого ящика". Это означает, что протестировать сайт надо не только с внешней стороны, но и со стороны его кода. Проверьте, везде ли используются одинаковые решения (например, везде ли тег `<h1>` выступает в качестве заголовка), везде ли оптимально написан код (нельзя ли что-то упростить). Результаты подобной работы сделают ваши страницы еще более легкими и быстро загружаемыми.

Тестирование удобства использования

Теперь надо кратко рассказать о том, что такое удобство использования. Его еще называют хитрым английским словом usability. Usability — это свойство, определяющее комфортность работы посетителя вашего сайта (простота, скорость, удовлетворенность результатами). Трудно привести какой-то глобальный пример, показывающий, что такое usability, тем более что этот показатель складывается из большого количества различных факторов. Для понимания общего смысла usability можно сказать просто: читать черный текст на белом фоне удобнее и приятнее, чем красный на оранжевом; читать текст, разделенный на абзацы и обозначенный заголовками, удобнее, чем сплошную "простыню" текста без разделителей и переводов строк. Прочитав всплывающую подсказку к иконке, проще понять, что делает система при щелчке по этой иконке и т. д.

Перечислю вкратце те факторы, которые влияют на usability сайта и которые нужно протестировать в первую очередь:

- важная информация должна быть выделена визуально (цветом, размером, шрифтом и т. д.). Это необходимо, чтобы пользователь мог сразу сориентироваться на новой странице и не отвлекаться на второстепенные моменты. Примером такой важной для пользователя информации может быть заголовок страницы или название формы;

- текст должен быть хорошо читаемым (т. е. иметь контраст с фоном);
- шрифт должен быть достаточно простым и не мелким ("фигурные" шрифты читать намного труднее, чем классические типа Arial, Times, Verdana, Sans Serif);
- на каждой странице должна быть установлена правильная кодировка текста (например, кодировка Windows-1251). Если вы принудительно не установили кодировку на своих страницах, то браузер пользователя попытается сделать это сам, и зачастую неудачно;
- текст не должен содержать грамматических ошибок. Скопируйте ваши страницы в текстовый редактор с проверкой орфографии (например, Microsoft Word) и исправьте грамматические ошибки;
- всем разделам сайта, заголовкам, надписям на кнопках необходимо давать четкие и понятные названия, чтобы пользователи не гадали, что означает та или иная надпись (например, такие несуразные названия, как "Применить к текущему", "Информация" или "Отменить почту");
- не должно быть очень мелких элементов (иконок, гиперссылок в виде многоточия и т. п.);
- необходимо всегда давать пользователю возможность понять, в каком разделе сайта он находится (также неплохо показывать, с какой страницы или из какого раздела он пришел). Это позволяет легче ориентироваться в вашей навигационной модели и понимать структуру сайта;
- нежелательно использовать на информационных сайтах динамические элементы (бегущие строки, плавающие или мелькающие изображения), т. к. они отвлекают пользователя от основного содержания сайта и не дают ему сосредоточиться;
- необходимо предусматривать возможность наличия у пользователей различных браузеров и мониторов с разным разрешением;
- нежелательно, чтобы на какой-то из ваших страниц появилась горизонтальная полоса прокрутки. Пользователям будет очень неудобно перемещаться по странице, прокручивая ее и по вертикали, и по горизонтали.

Если все указанные правила будут выполняться, можете надеяться на то, что ваш сайт принесет кому-то пользу.

Кроме самостоятельной проверки, можно еще устроить настоящий usability тест, как это делают серьезные фирмы. Для такого теста вам нужно просто показать свой сайт нескольким друзьям или знакомым и попросить их высказать свои замечания. Попросите их выполнить какую-нибудь задачу на вашем сайте (например, найти конкретную статью), таким образом вы сможете проверить, понятно ли разделили свой сайт на разделы и подразделы. Попросите человека, тестирующего ваш сайт, чтобы он комментировал вслух свои действия. Наверняка вы после подобного испытания внимательно обдумаете те моменты, на которых споткнулся ваш "контролер".

Web-этикет

Web-этикет — это правила хорошего тона во Всемирной паутине, соблюдение которых позволит вам заслужить уважение и доверие ваших посетителей:

- старайтесь не допускать в текстах на своем сайте грамматических ошибок и неточностей;
- если вы пообещали посетителю какие-то сведения или новые возможности, не обманывайте его. Не давайте ссылок на раздел "Масса информации по ремонту кухни", если в разделе всего одна или две статьи;
- не заставляйте пользователей заполнять большую форму — не всегда они хотят заполнять формы и анкеты, даже чтобы получить демонстрационную версию какой-нибудь программы;
- не предлагайте пользователям заплатить вам денег за какой-нибудь пустяк, который они могут бесплатно найти на других сайтах;
- всегда ссылайтесь (указывайте источник информации) на сайты, с которых вы взяли какие-либо статьи или информацию;
- не размещайте на своем сайте никакой навязчивой рекламы, а тем более не создавайте "pop-up" окон с рекламой, не пытайтесь насильно оставить посетителя на своем сайте, подсовывая ему окно браузера, которое невозможно закрыть;
- в текстах будьте вежливы.

Глава 11

Размещение и раскрутка сайта в Интернете

Регистрация сайта

Итак, сайт создан, собран, настроен и протестирован. Пришло время для его размещения в Интернете. Существует большое количество интернет-компаний, предоставляющих доменные имена для различных сайтов (я говорил об этом *в гл. 1*). Доменное имя (адрес сайта во Всемирной паутине) представляет собой англоязычный идентификатор, используемый для более легкого запоминания адреса человеком (фактический адрес компьютера в Сети — это всего лишь набор цифр).

Многие компании, предоставляющие услуги так называемого хостинга (т. е. размещения и регистрации в Интернете доменного имени вашего сайта) чаще всего работают с двумя видами услуг:

- позволяют любому пользователю Интернета бесплатно создавать на сервере компании сайт и получать для него имя. Однако это доменное имя имеет третий уровень (например, **www.homyak.iCompany.ru**). В этом адресе фигурирует доменная зона **ru** (указывает на то, что это сайт российский), **iCompany** — это название компании, которая предоставляет услуги бесплатного хостинга, **homyak** — это непосредственное имя вашего сайта. Вот таким образом и получается домен третьего уровня. Хостер (тот, кто предоставляет хостинг) добавляет к доменному имени бесплатных сайтов название своей компании в целях рекламы и привлечения дополнительных посетителей;
- платный хостинг — это предоставление домена второго уровня (например, **www.homyak.ru**) и достаточно большого объема файлового пространства на своем сервере за определенную плату.

Я пока пользуюсь лишь бесплатным хостингом.

Как самый настоящий Вася Пупкин, я решил зарегистрировать свой сайт на сервере **www.narod.ru**, принадлежащем компании Яндекс. Для этого я набрал в адресной строке своего браузера **www.narod.ru** и нажал клавишу <Enter>.

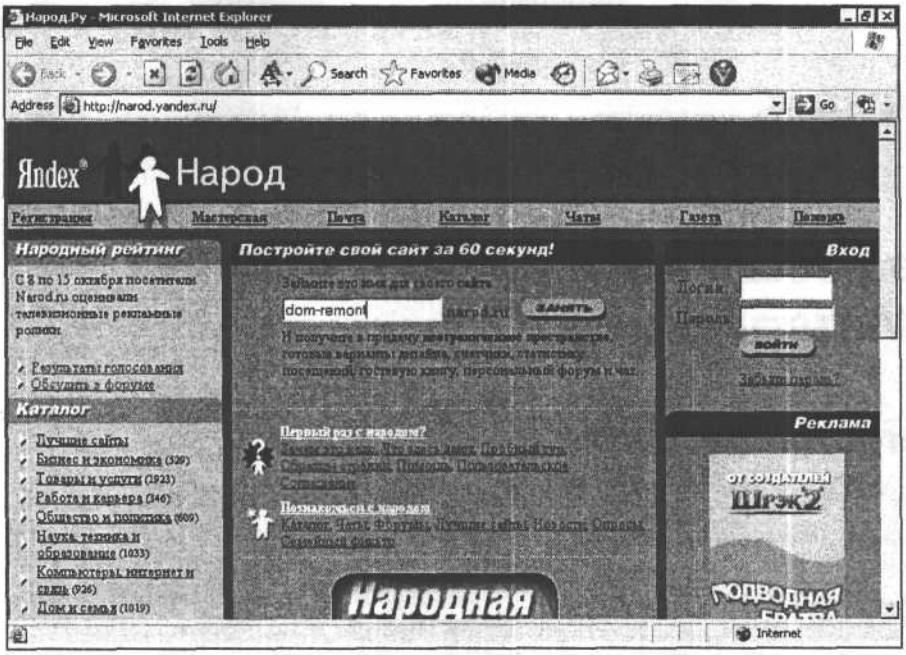


Рис. 11.1. Главная страница "Народ.Ру", с которой можно войти на страницу управления своим сайтом или зарегистрировать новый сайт

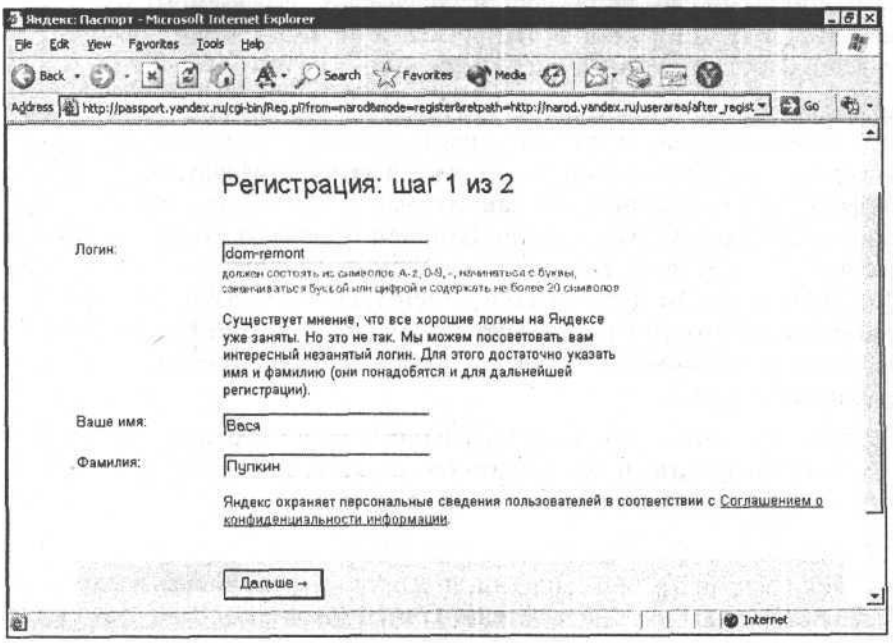


Рис. 11.2. Первый шаг регистрации сайта на "Народ.Ру" (регистрация имени нового сайта)

На открывшейся странице (рис. 11.1) мне сразу же предложили вход (в том случае, если я уже зарегистрирован) или выбор имени для моего будущего сайта (если еще не зарегистрирован). Я набрал в поле ввода для имени сайта то название, которое придумал заранее и с замиранием сердца стал ждать, что мне ответят.

После нажатия на кнопку **Занять** передо мной появилась форма (рис. 11.2), в которой была показана первая часть имени моего будущего сайта в качестве имени пользователя (login), и где необходимо было также ввести имя и фамилию. На этом этапе обычно проверяется, нет ли уже такого пользователя и, соответственно, сайта с таким названием.

Услугами как этого, так и сотен других бесплатных хостеров пользуются миллионы посетителей, поэтому практически все интересные названия и даже имена и фамилии (вроде **pupkin.narod.ru**) давным-давно заняты. Обычно сразу же вам предлагают другие названия, которые, однако, тоже чаще всего заняты. Например, когда я захотел использовать название **remont.narod.ru**, оказалось, что такой домен уже занят и мне подсказали, что можно получить домен **remont1.narod.ru** или **remont2004.narod.ru**. Я придумал название **dom-remont.narod.ru**. К моему великому удивлению это имя оказалось не занятым, и я продолжил регистрацию.

Регистрация состоит из нескольких шагов, во время которых нужно указывать различные сведения о себе, о будущем сайте и т. д.

Во время второго шага регистрации (рис. 11.3) мне понадобилось ввести свой пароль, выбрать контрольный вопрос, ответ на который поможет мне в том случае, если забуду пароль. Также я должен был указать свой электронный почтовый ящик (если он у меня уже имеется), а еще я должен ввести контрольные цифры, иначе мне не позволят зарегистрироваться. Эти контрольные цифры вводятся для того, чтобы доказать, что я живой человек, а не программа-робот, которая автоматически регистрирует себя у бесплатных хостеров с той целью, чтобы потом с этого почтового адреса рассылать рекламные письма сомнительного содержания (так называемый спам).

После успешной регистрации меня попросили сообщить некоторую информацию о себе, о том, как ко мне обращаться, какого я пола и т. д. (рис. 11.4).

Финалом регистрации явилась страница "Мастерская" (рис. 11.5), на которую я попал после ввода своего логина (имени пользователя) и пароля. С этой страницы я могу загружать на свой сайт файлы, читать почту, создавать различные разделы (вроде гостевой книги), а также изменять свои данные и персональные настройки.

И теперь после того, как я загружу на эту страницу свои HTML-страницы, таблицы стилей и картинки, мой сайт появится в любой точке земного шара у того пользователя, который наберет в адресной строке своего браузера **www.dom-remont.narod.ru**.

Надо обратить внимание на одно обстоятельство. Во всех гиперссылках всегда указывается имя файла HTML-страницы, к которой необходимо перейти. На своем компьютере вы также скорее всего указывали имя начальной страницы сайта, чтобы начать его просмотр. Однако в адресе **www.dom-remont.narod.ru** нет никакого упоминания о главной странице. Дело в том, что чаще всего такого рода серверы настроены так, что если не указана конкретная страница, они ищут страницу с именем **index.html** и показывают ее.

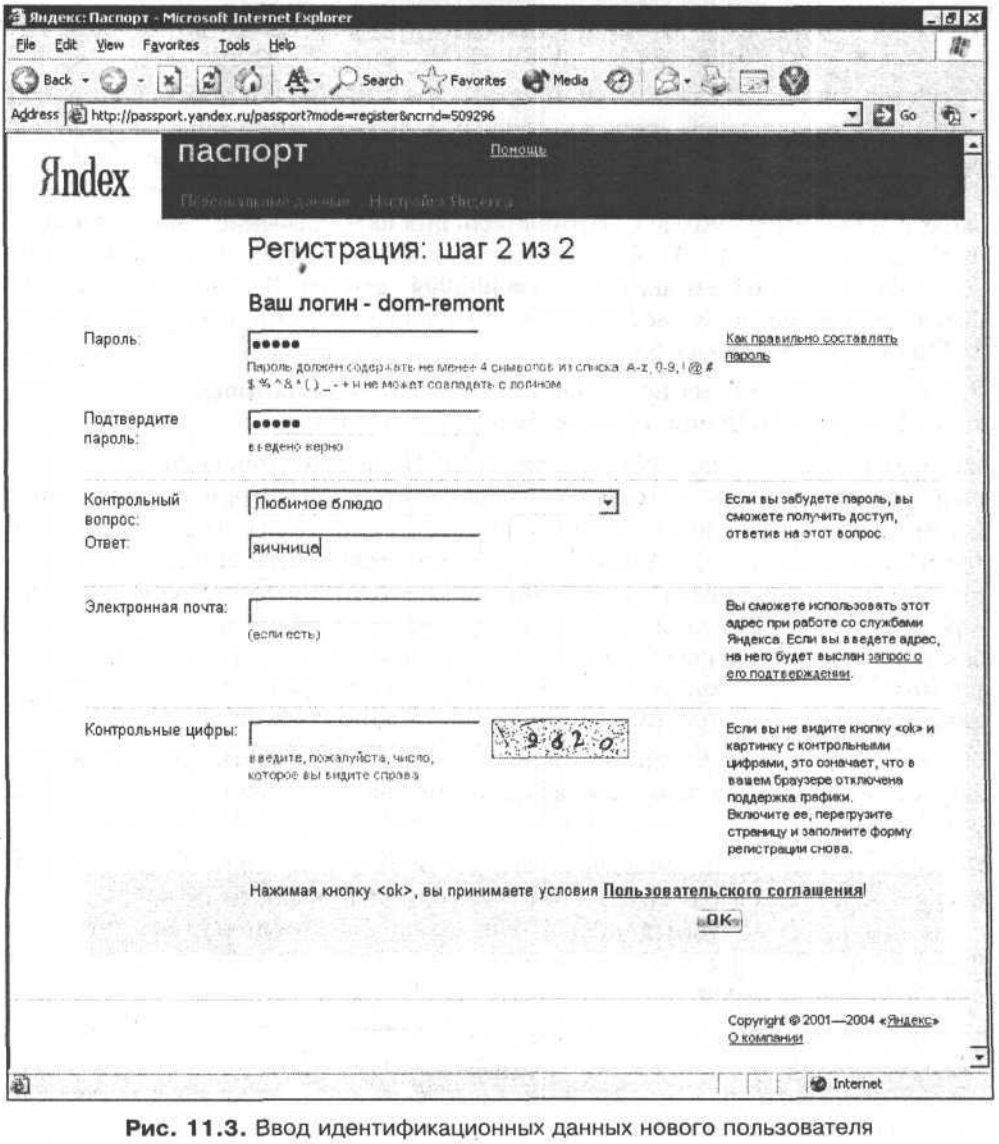


Рис. 11.3. Ввод идентификационных данных нового пользователя

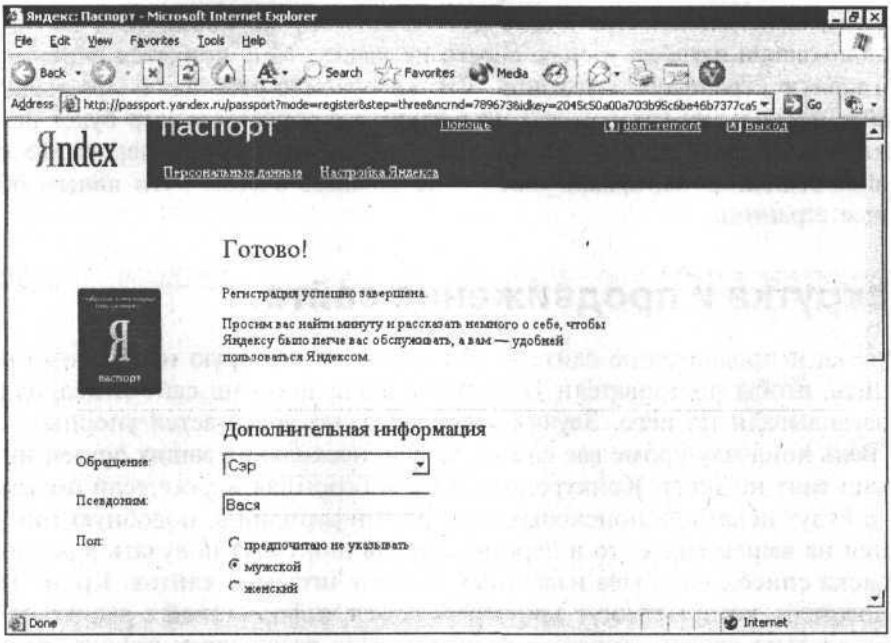


Рис. 11.4. Завершение регистрации

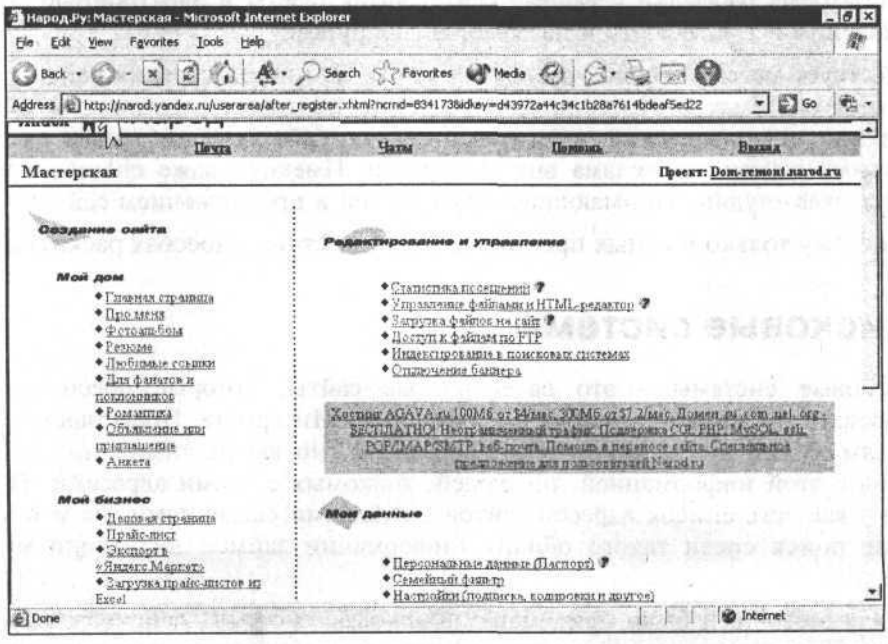


Рис. 11.5. Страница "Мастерская"
(для работы с собственным сайтом и электронной почтой)

Так, если моя главная страница будет называться **start.html**, то я по адресу **www.dom-remont.narod.ru** ничего своего не увижу. Мне придется переименовать главную страницу в **index.html**. Это же правило работает и при наличии на сайте нескольких каталогов. Точно таким же образом сервер будет искать страницу **index.html**, если набрать в браузере или указать в гиперссылке путь **www.dom-remont.narod.ru/hand_made** и не задавать в этом пути явным образом имя страницы.

Раскрутка и продвижение сайта

Раскрутка и продвижение сайта — это та работа, которую необходимо производить, чтобы пользователи Интернета знали про ваш сайт и периодически заглядывали на него. Звучит это просто, но достигается упорным трудом. Ведь поначалу кроме вас самих, да еще нескольких ваших друзей никто про ваш сайт не знает. Конкуренция в Сети огромная и даже если пользователи и будут искать на поисковых серверах информацию, подобную той, что имеется на вашем сайте, то в первую очередь они будут получать в результате поиска список наиболее известных и часто читаемых сайтов. Кроме того, пользователи, которые могут заинтересоваться информацией с вашего сайта, не обязательно должны узнать о нем только по результатам поиска, но также и просматривая интернет-каталоги известных поисковых серверов, получая почтовые рассылки с вашего сайта, читая газеты и электронные доски объявлений и т. д. Все это и называется раскруткой.

Существует масса способов раскрутки сайта. Это и регистрация сайта в поисковых системах в Интернете, и создание почтовых рассылок, и обмен ссылками, и привлечение посетителей на ваш сайт за ваши деньги на специальных сайтах, и реклама вне Интернета. Имеются даже специализированные web-студии, занимающиеся раскруткой и продвижением сайтов.

Я расскажу только о самых простых и общеизвестных способах раскрутки.

Поисковые системы

Поисковые системы — это своеобразные сайты, которые обеспечивают пользователям удобство поиска информации в Интернете. Представьте себе, что вам нужно найти какую-то информацию, но вы не знаете ни адресов сайтов с этой информацией, ни людей, знакомых с этими адресами. Пусть даже у вас есть список адресов сайтов с нужными сведениями, но и в этом случае поиск среди такого обилия информации займет достаточно много времени.

Вот для этого-то и были придуманы поисковые системы, основная суть которых заключается в том, чтобы по запросу пользователя выдавать список сайтов, содержащих запрашиваемую или подобную ей информацию.

Владелец сайта должен зарегистрировать его в поисковой системе, указав некоторые сведения, после чего его сможет найти специальная программа–робот (ее еще называют "паук", ведь речь идет хоть и о компьютерной, но паутине). Паук прочитает всю информацию с сайта, проанализирует ее и запишет в память поисковой системы (или, как еще говорят, проиндексирует). Различные поисковые системы используют различные способы и приемы индексирования. Условно говоря, индекс — это файл, в котором записаны различные слова и указаны сайты, на которых эти слова встречаются, а также имеется информация о том, как часто они встречаются. При индексировании, как правило, пауки приводят все похожие слова к единому виду (например, слова "понятие", "понятию", "понятием" будут записаны в индекс как одно слово "понятие", кроме того, при этом игнорируются предлоги, междометия, союзы и пр.). Информация с ваших страниц попадает в такой индекс после регистрации в поисковой системе.

Для примера я опишу регистрацию в российской поисковой системе Рамблер.

Как правило, на главной странице каждой поисковой системы есть ссылка, которая называется "Добавить сайт", "Добавить ресурс", "Зарегистрировать ресурс" и т. п. (рис. 11.6). В любом случае это ссылка для регистрации вашего сайта в поисковой системе. После того, как ее нажмете, вы попадаете в форму регистрации своего сайта (рис. 11.7).

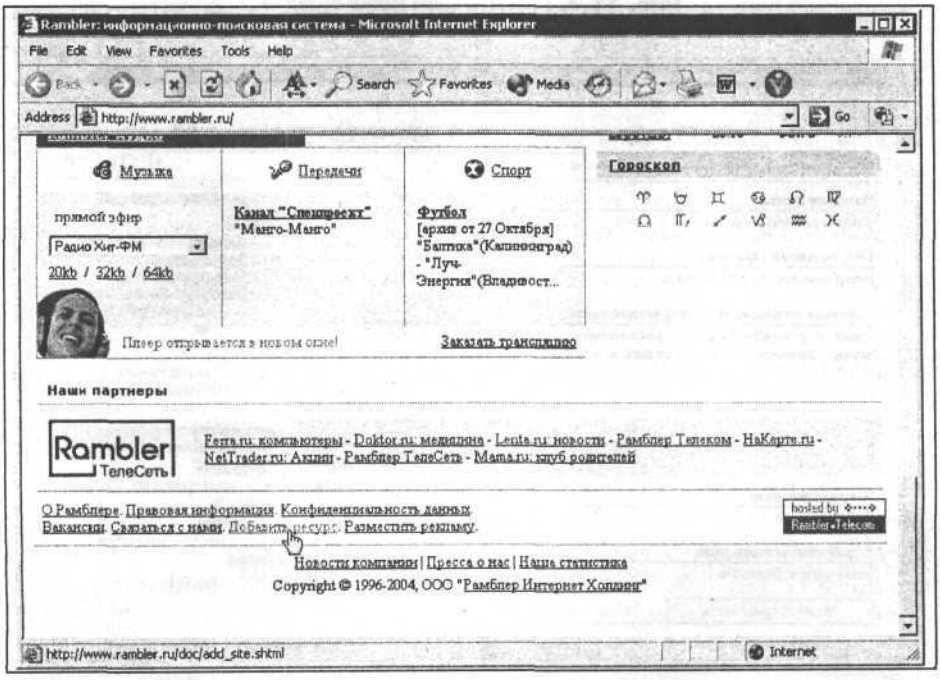


Рис. 11.6. Ссылка "Добавить ресурс" в нижней части главной страницы **rambler.ru** для регистрации нового сайта в поисковой системе Рамблер

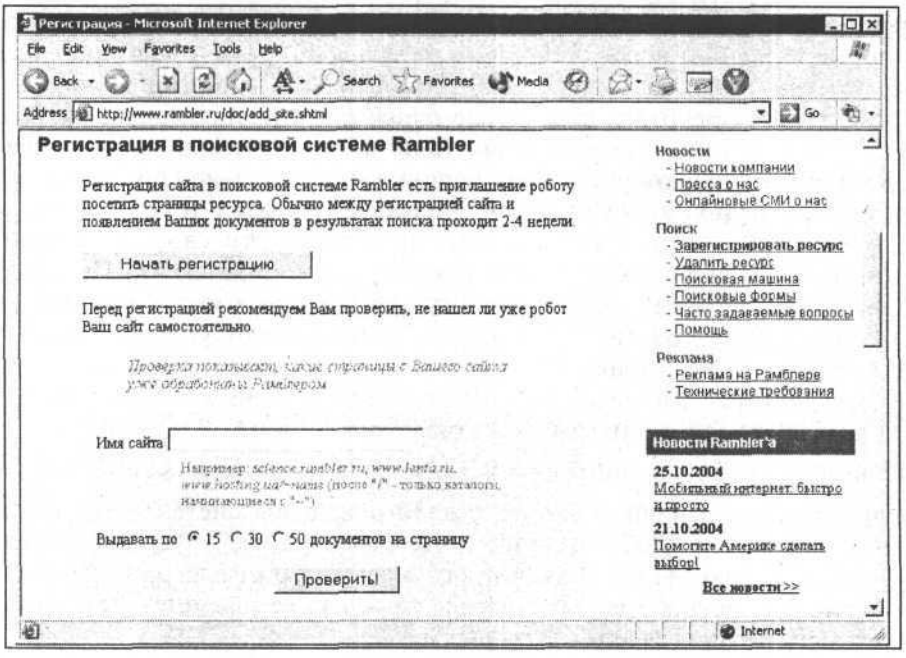


Рис. 11.7. Регистрация нового сайта

Рис. 11.8. Ввод кратких сведений о сайте и о его владельце

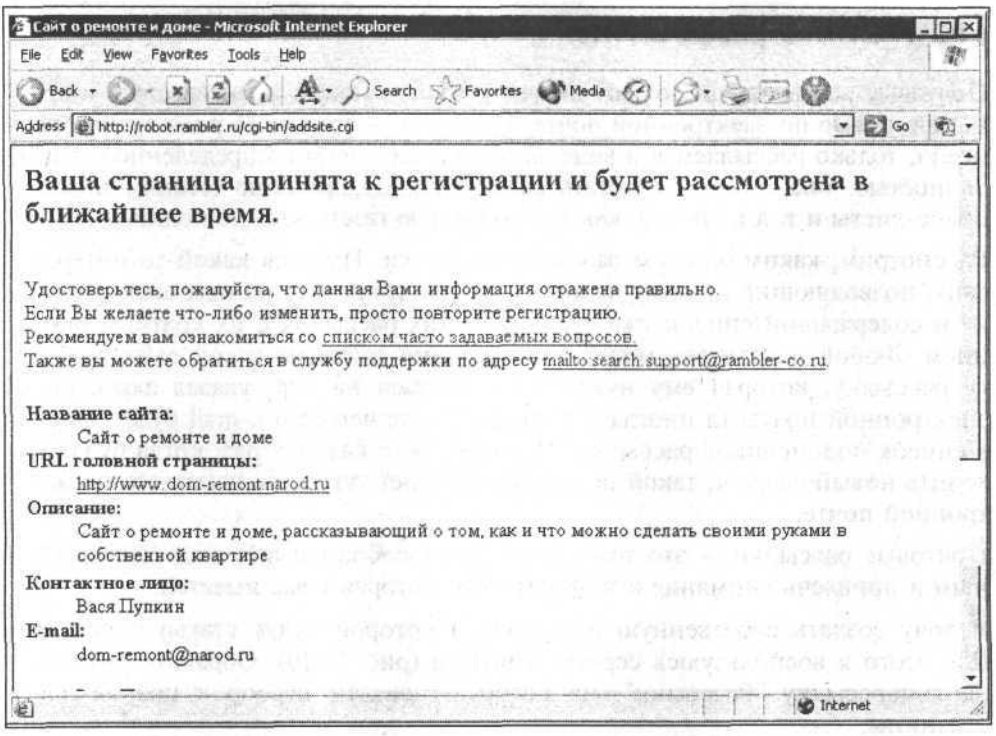


Рис. 11.9. Поздравления с успешной регистрацией

При регистрации на Рамблере, на первой странице мне предложили проверить, не нашел ли робот этой поисковой системы мой сайт. Если учесть, что я только что создал свой сайт, то вероятность этого равна нулю, поэтому я сразу начну регистрацию (рис. 11.8).

После ввода кратких сведений о моем сайте я должен нажать на кнопку **Зарегистрировать**. Если все сделано верно, то я увижу страницу с указанными мной сведениями и несколькими советами и рекомендациями (рис. 11.9).

Через две-три недели после регистрации на Рамблере ваш сайт начнет появляться в результатах поиска. Кроме Рамблера существует еще множество поисковых систем, из которых нужно выбрать несколько наиболее известных и популярных (например, **Google.com**, **Aport.ru**, **Tut.by**) и зарегистрировать сайт на них. Для этого полезно нужную информацию (вроде названия сайта, его описания и адреса электронной почты) записать в каком-либо файле, а потом просто вставлять ее в нужные поля. Ведь практически все поисковые серверы требуют при регистрации одинаковые сведения.

Почтовые рассылки

Почтовые рассылки позволяют людям получать новую и интересную информацию прямо по электронной почте. Рассылка — это своего рода электронная газета, только рассылаемая в виде электронных писем с определенной периодичностью, она может содержать различную информацию (статьи, новости, прайс-листы и т. д.). На нее, как и на обычную газету, можно подписаться.

Рассмотрим, каким образом работают рассылки. Имеется какой-то интернет-сайт, позволяющий любому человеку зарегистрировать на нем свою рассылку и содержащий список уже существующих рассылок с их кратким описанием. Любой посетитель может при желании зайти на такой сайт, выбрать ту рассылку, которая ему нужна, подписаться на нее, указав адрес своей электронной почты (а иногда и пароль), после чего его e-mail будет занесен в список подписчиков рассылки. И после этого каждый раз, когда будет выходить новый выпуск, такой пользователь будет тут же получать его по электронной почте.

Почтовые рассылки — это тоже один из способов сделать ваш сайт известным и привлечь внимание к информации, которая у вас имеется.

Я хочу создать собственную рассылку, в которой будут статьи о ремонте. Для этого я воспользуюсь сервером **mail.ru** (рис. 11.10). Обратите внимание на гиперссылку "Рассылки", на которую наведен курсор в правой части страницы.

После перехода по ссылке "Рассылки" я попадаю на страницу, полностью посвященную рассылкам сервера **mail.ru** (рис. 11.11). Здесь содержится каталог существующих рассылок, гиперссылки для создания новой, отправки существующей рассылки, для подписки на рассылку и т. д. Я выбираю "Создать рассылку".

Далее, подобно тому, как это было при регистрации сайта, необходимо указать некоторую информацию о себе — пароль, адрес сайта (если он есть, ведь для того, чтобы делать рассылки не обязательно даже иметь свой сайт), информацию о своей рассылке (тему, описание, периодичность и т. д.) (рис. 11.12).

После регистрации указанная вами информация будет отправлена администратору рассылок (рис. 11.13), который и решит, можно ли создавать такую рассылку, не нанесет ли она ущерба репутации **mail.ru**, полезна ли она и т. д. Так или иначе, но в большинстве случаев ответ бывает положительным. Спустя несколько дней вам должно прийти письмо по электронной почте о том, что все нормально, рассылку зарегистрировали, ее можно рассылать тем подписчикам, которые уже успели зарегистрироваться.

В дальнейшем можно просто заходить по гиперссылке "Отправить рассылку", вводить идентификатор рассылки, свой пароль, текст рассылки. После нажатия на кнопку **Отправить**, спустя какое-то время все ваши подписчики получат письма с новым выпуском рассылки.

Рис. 11.10. Главная страница **mail.ru**

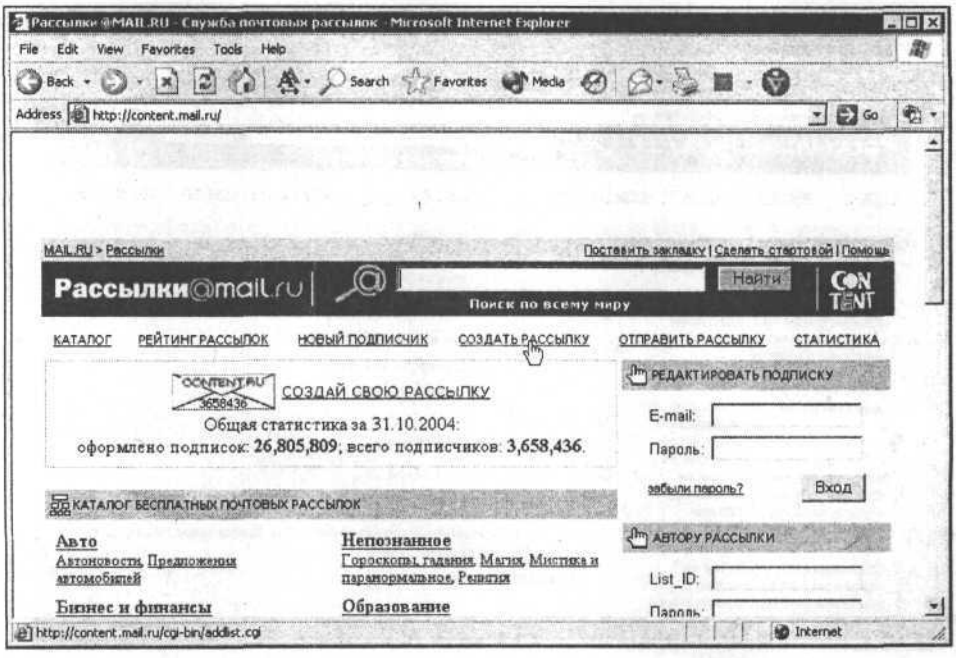


Рис. 11.11. Главная страница работы с рассылками на **mail.ru**

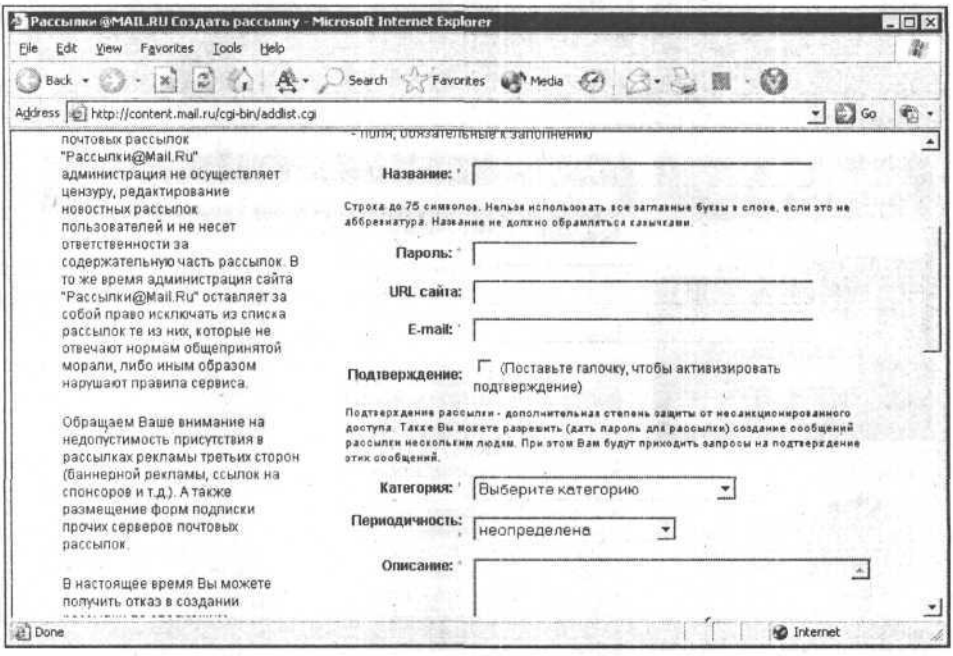


Рис. 11.12. Страница регистрации новой рассылки на **mail.ru**

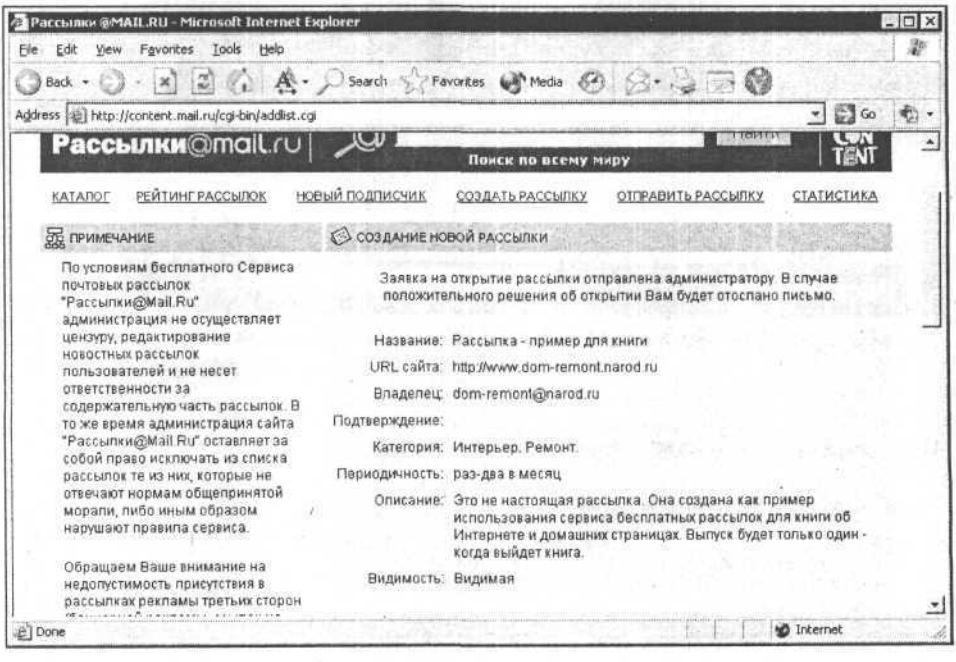


Рис. 11.13. Финальная страница регистрации рассылки

On-line и off-line реклама

Кроме регистрации сайта на поисковых серверах в различных информационных интернет-каталогах и создания собственной рассылки, существуют еще некоторые простые и ненавязчивые способы раскрутки сайта.

Я разделяю их на способы, относящиеся к on-line и off-line рекламе. Под термином on-line понимают способы, связанные с самим Интернетом (on-line — это когда есть соединение с сетью). Off-line — это способы, связанные с обычной, реальной, а не виртуальной жизнью (соединение с сетью отсутствует).

В Интернете существуют различные сервисные возможности (например, так называемые "Доски объявлений", форумы, специальные сайты, на которых можно разместить информацию о своем сайте). На электронной доске объявлений (их в Интернете тысячи) можно поместить объявление о том, что есть, мол, такой-то сайт с такой-то тематикой. Существуют специальные форумы, предназначенные для того, чтобы вы могли поинтересоваться мнением других людей (например, о дизайне или о содержании своего сайта).

Кроме того, имеется способ так называемого обмена ссылками. Кто-нибудь может написать вам письмо с предложением обменятся ссылками. Он разместит вашу ссылку у себя на сайте, а некоторые его посетители могут заинтересоваться и перейти по ней на ваш сайт, и наоборот.

Когда вы регистрируетесь как новый пользователь на различных сайтах (будь то форумы, сообщества единомышленников или что-то еще), то также можно указать адрес своего сайта, и если кто-нибудь заинтересуется вашей темой, то сможет посмотреть регистрационную информацию и оттуда перейти на ваш сайт.

Помимо указанных, имеются еще десятки различных способов раскрутки сайта (наберите в строке запроса любого поискового сервера `Раскрутка сайта` и, может быть, найдете для себя что-нибудь приемлемое).

И, наконец, немного об off-line рекламе (как помните, это реклама вне сети).

Вне сети вы можете рассказывать о своем сайте знакомым, можете печатать статьи в каких-нибудь журналах и газетах, приводить там адрес своего сайта, можете помещать адрес сайта и свой e-mail, например, на визитках и т. д.

Запрещенные приемы

Запрещенные приемы — это такие приемы раскрутки, которые создают искусственное увеличение аудитории путем обманного привлечения пользователей на сайт. Это приемы, в которых используются особенности работы пауков с поисковых серверов. Также нередко для нечестной раскрутки сайта злоумышленники рассылают всем рекламные письма, призывающие посетить сайт.

Иногда искусственно создается большое количество текстов по тематике сайта, который раскручивают. Тексты чаще всего берутся с чужих сайтов, но в них внедряются ссылки на разделы сайта злоумышленника. В результате поисковый сервер обнаруживает на таком сайте очень много ключевых слов по запросам, из-за чего этот сайт и попадает на первые места.

Иногда в дополнение к таким страницам используют автоматическое перенаправление посетителя на какую-то из страниц сайта злоумышленника. Если первый прием обнаружить легко, то второй "ход конем" уже можно упустить.

Часто на страницы сайта с обычным содержимым добавляют опять же тексты с наиболее популярными ключевыми словами и наиболее часто задаваемыми запросами, но только эти тексты делают такого же цвета, как и фон страницы (либо очень мелким). Неопытный пользователь их не видит и недоумевает, почему он попал на какой-то странный сайт с сомнительным содержимым.

Часто используют метод сокрытия истинного содержимого сайта, в результате чего на поисковом сервере такой сайт может быть проиндексирован по информации, которую не содержит.

С каждым годом совершенствуются как запрещенные приемы, так и методы борьбы с ними администраторами известных сайтов, а также работа поисковых роботов. Если по каким-то данным обнаруживается подвох, то сайт злоумышленника исключается из результатов поиска на данном сервере раз и навсегда.

Счетчики посетителей

Счетчик посетителей — это специальный скрипт, который собирает информацию о количестве посетителей, которые заходили на конкретный сайт. Ведь в самом деле, кроме количества подписчиков рассылки, человеку интересно знать, сколько вообще людей попало на его сайт, как они туда попали, что они там читали и т. д. Создавать свой счетчик посетителей, может и интересное занятие, но долгое и требующее значительных усилий и терпения. А люди, которые профессионально занимаются этим, уже имеют огромный опыт и знают, что важно, а что нет. Я, например, для сбора статистики посещения моего сайта пользуюсь услугами компании **Spylog.com**. Там предоставляют информацию о том, кто, когда и каким образом попадал на мой сайт, в какое время суток, с помощью какого браузера, можно узнать также, какая у посетителя установлена операционная система, какое разрешение монитора и т. д.

Для этого надо набрать в адресной строке браузера **http://www.spylog.com**, затем на открывшейся странице выбрать нужный сервис (SpyLOG Tracker) и нажать на ссылку "Зарегистрироваться" (рис. 11.14).

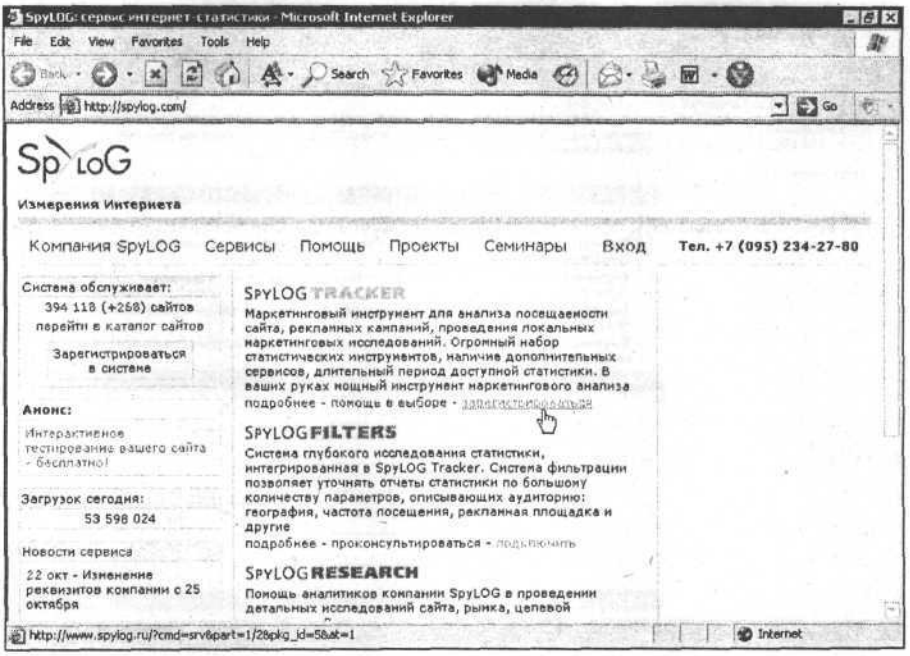


Рис. 11.14. Главная страница регистрации на сервере SpyLOG

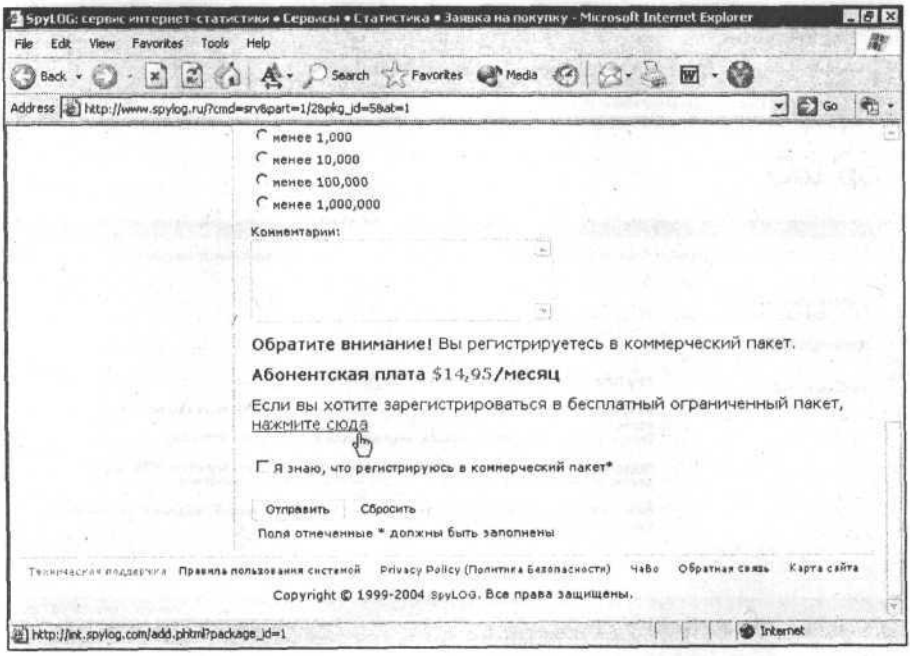


Рис. 11.15. Вход на страницу регистрации для пользователей бесплатной статистики

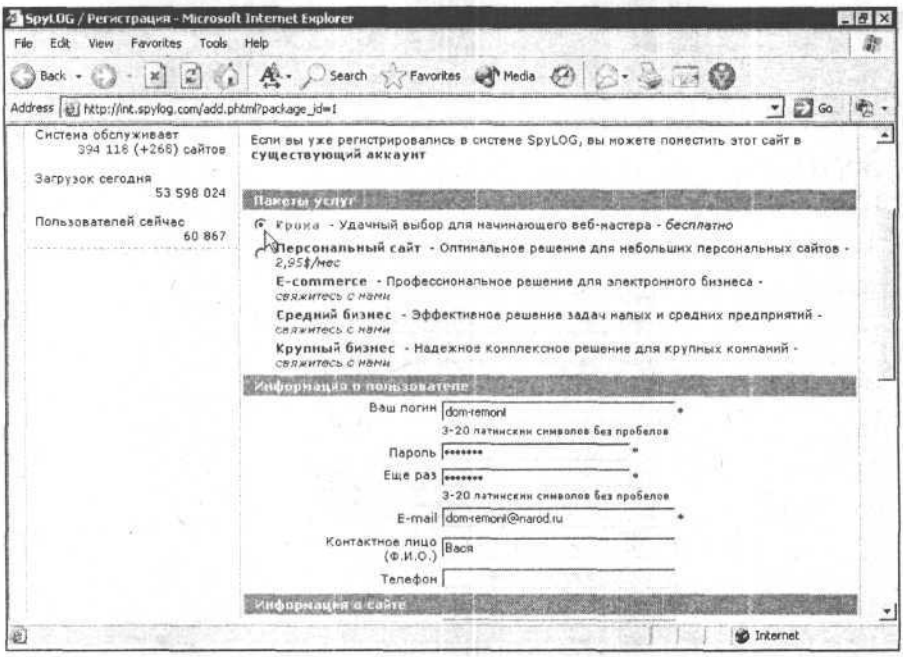


Рис. 11.16. Ввод регистрационных данных

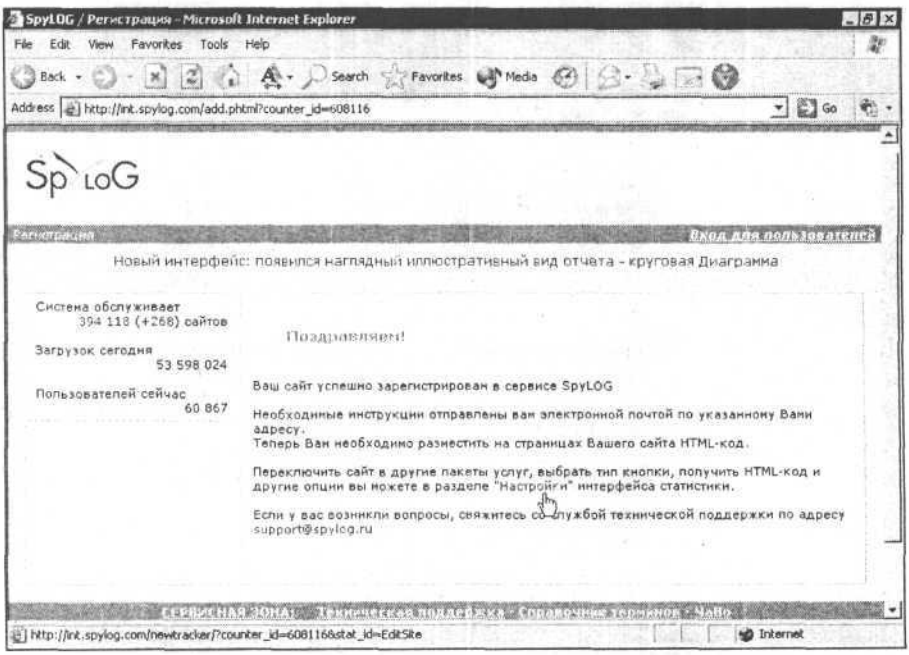


Рис. 11.17. Поздравления

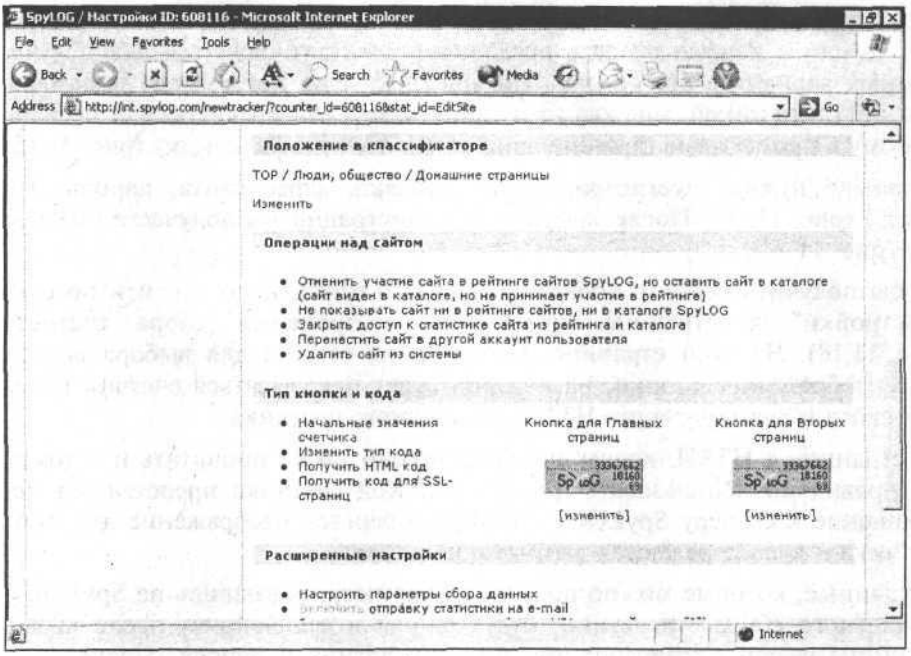


Рис. 11.18. Страница управления настройками сбора статистики и получения кода счетчика посещений

Рис. 11.19. Код счетчика

Поскольку SpyLOG ориентирован в основном как сервис для профессионалов, то они в первую очередь предлагают зарегистрироваться для получения платных вариантов статистики. Но мне пока еще нет смысла пользоваться платной статистикой, мне хватит и той, что SpyLOG предоставляет бесплатно, поэтому я проматываю страницу вниз и нахожу нужную ссылку (рис. 11.15).

А дальше нужно зарегистрироваться (указать адрес сайта, пароль, e-mail и т. д.) (рис. 11.16). После завершения регистрации вы получаете поздравление (рис. 11.17).

Чтобы получить счетчик посетителей своего сайта, надо перейти по ссылке "Настройки" к странице управления настройками сбора статистики (рис. 11.18). На этой странице расположены ссылки для выбора внешнего вида изображения кнопки, на которой будет показываться счетчик посещений сайта и для получения HTML-кода самого счетчика.

На странице с HTML-кодом счетчика можно также прочитать и о том, как его правильно использовать (рис. 11.19). Код счетчика представляет собой обращение к серверу SpyLOG, с которого берется изображение для кнопки счетчика и куда передаются данные о посетителе.

Все данные, которые можно получить, зарегистрировавшись на SpyLOG для бесплатного сбора статистики, будут видны в дальнейшем после входа на страницы вашей статистики при помощи логина и пароля. После первого входа эта страница выглядит так, как представлено на рис. 11.20.

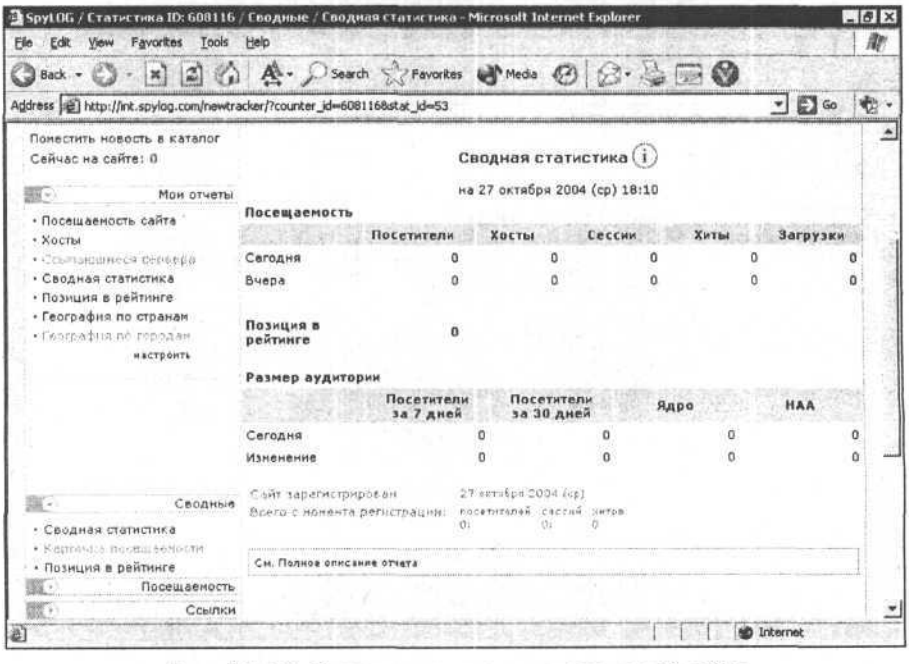


Рис. 11.20. Страница статистики посещений сайта

Обслуживание сайта

Обслуживание сайта является важным процессом в его жизнедеятельности. Оно подразумевает не только постоянное обновление содержимого сайта, но и анализ его работы с целью выявления моментов, которые требуют улучшения или более пристального внимания. Правильное обслуживание позволит вашему сайту быть интересным, полезным и качественным.

Например, внимательное отслеживание статистики позволит не только узнать, насколько широко вам удалось раскрутить сайт, но и понять, что нужно сделать, чтобы еще более усовершенствовать его. Например, если я вижу, что больше половины посетителей в первую очередь интересуется изделиями из дерева или фотогалереями со смешными фотографиями, то постараюсь уделить этим разделам больше внимания в будущем. Разделы сайта, которые пользуются низкой посещаемостью, наоборот, придется переделать или вообще убрать.

В целях профилактики необходимо время от времени проверять работу сайта — не нарушилось ли что-нибудь в структуре страниц и гиперссылок.

Нужно следить за положением сайта в результатах поиска по различным запросам, продолжать его раскрутку, вносить изменения в дизайн и содержимое, если это окажется необходимым (словом, ухаживать за ним, как если бы это был обычный хомяк, а не виртуальный).

Заключение

Ну, вот, друзья, вы уже по эту строну книги. Хочется сказать, что долгое и, надеюсь, увлекательное путешествие закончилось, но нет — на самом деле оно только начинается. Вы многое узнали и можете продолжить изучение искусства создания web-сайтов самостоятельно, без моей помощи. Я лишь приоткрыл вам дверцу в огромный, быстро развивающийся, интересный мир. Надеюсь, что изучение языка HTML, Интернета и новых сетевых технологий вы продолжите и когда-нибудь вспомните Васю Пупкина и его сайт "О ремонте и доме" добрыми словами.

А сайт, о котором я писал, действительно существует, и вы можете найти его по адресу **http://www.dom-remont.narod.ru.** Правда, на нем нет статей по ремонту — я говорил о них лишь для примеров, но зато вы можете посмотреть "вживую" на те HTML-страницы, что появлялись в процессе повествования. Вы найдете в коде страниц описываемый мной JavaScript, увидите реальный счетчик посетителей сайта и, может быть, напишете мне что-нибудь в гостевой книге.

Искренне ваш, Вася Пупкин.

Предметный указатель

Р

С

Т

У

Ф

Х

Ц

Ш

Я