И. Шапошников

bhv
www.bhv.ru
www.bhv.kiev.ua

Web-сайт своими руками

МАСТЕР

ПРАКТИЧЕСКОЕ РУКОВОДСТВО

Основы языка HTML

Возможности Microsoft FrontPage2000

Администрирование и безопасность сайтов

Создание электронного магазина средствами WebShop 2000

Игорь Шапошников

Web-сайт своими руками

Дюссельдорф ◆ Киев ◆ Москва ◆ Санкт-Петербург

УДК 681.3.06

Книга поможет пользователям создать собственную Web-страницу и свой Web-сайт. Рассматриваются основы языка HTML, процедура создания и обработки графических изображений, мультимедийные варианты наполнения сайтов. Кратко освещены вопросы размещения, администрирования и безопасности создаваемых Web-сайтов.

В книге содержится необходимая теоретическая информация и практические советы, которые помогут освоить специализированные прикладные программы Microsoft FrontPage 2000 и WebShop 2000, а также некоторые приложения для работы с графикой.

Для широкого круга пользователей

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зав. редакцией	*Наталья Таркова*
Редактор	*Елена Бабко*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Наталия Першакова*
Дизайн обложки	*Ангелины Лужиной*
Зав. производством	*Николай Тверских*

Шапошников И. В.

Web-сайт своими руками. — СПб.: БХВ — Санкт-Петербург, 2000. — 224 с.: ил.

ISBN 5-8206-0130-0

© И. В. Шапошников, 2000
© Оформление, издательство "БХВ — Санкт-Петербург", 2000

Лицензия ЛР № 065953 от 15.06.98. Подписано в печать 07.07.2000.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 18,06.
Тираж 4000 экз. Заказ
"БХВ — Санкт-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар, № 77.99.1.953.П.950.3.99
от 01.03.1999 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН.
199034, Санкт-Петербург, 9-я линия, 12.

Содержание

Благодарности

Автор никогда не является единоличным создателем книги. Всегда есть те, без чьего участия книга никогда не появилась бы на свет, несмотря на все усилия автора. Мне хотелось бы поблагодарить всех, чья поддержка была так необходима.

Прежде всего, это, конечно, главный редактор издательства "BHV — Санкт Петербург" Екатерина Кондукова и принимающий редактор Елена Бабко. Без вашей неоценимой помощи, видения перспектив и отточенного чувства языка я бы просто не справился.

Также нельзя не отметить всех работников издательства, которые принимали участие в работе над этой книгой. Спасибо вам.

И уж наверняка этой книги бы не было, если бы не постоянная неоценимая поддержка моей жены — Даниленко Ольги. Любимая, все, что я делаю, я делаю для тебя.

Спасибо всем моим друзьям в Сети. Мой почтовый ящик всегда открыт для вас.

Игорь Шапошников
shiva1@interdacom.ru

Введение

Другу и любимой — Даниленко Ольге

Когда человек получает доступ к Интернету, что он видит, выходя на безбрежные информационные просторы? Куда он движется по скоростным цифровым магистралям? В подавляющем большинстве случаев он видит Web-сайты и Web-страницы. И какое же вполне объяснимое желание начинает вскоре охватывать нашего путешественника? Правильно! Ему хочется сделать свою страничку. У него возникает желание создать свой сайт. Пожалуй, я не буду далек от истины, если скажу, что это желание посещало абсолютное большинство пользователей Интернета. Хорошо тем, кто, как говорится, родился с микрочипом в голове. Тем, кто знает все правила работы в Интернете и умеет программировать. А что же делать всем остальным? Неужели для того, чтобы сделать свое личное представительство во всемирной сети, необходимо сидеть и изучать все эти языки программирования, правила работы и протоколы? На самом деле нет, не надо.

Посмотрим внимательно на ситуацию в компьютерном мире. С каждым годом программы становятся все более дружественными простому пользователю. Создатели операционных систем и программных продуктов очень хорошо понимают, что компьютер является рабочим инструментом не только и не столько для программистов, сколько для обычных людей, для которых работа с компьютером должна быть проста. Те, кого обычно называют пользователями, и являются финансовым мотором для всей компьютерной индустрии. В конечном счете, весь этот многомиллиардный (в долларовом выражении, естественно) бизнес ориентирован именно на обычного человека, у которого достаточно забот и без того, чтобы заниматься программированием. Естественно, для правильной и эффективной работы на компьютере необходимо учиться. Но ведь и для того, чтобы научиться водить автомобиль тоже необходимо время. Мы не должны посвящать учебе слишком много времени. У нас и без этого довольно много дел.

Я должен сразу предупредить, что это будет не просто. Самую простую страничку сделать легко, но если мы захотим создать что-то серьезное, придется как следует потрудиться. Так как мы сразу ограничили себя тем, что программированием заниматься не будем, то некоторая часть возможностей и средств для создания содержимого наших будущих страниц и сайтов будет для нас недоступна. Но вы очень удивитесь, когда узнаете, как многого мы можем добиться, правильно используя те программные средства, которые мы уже знаем и которыми пользуемся в повседневной работе.

Необходимо осознавать, что Интернет — это совершенно особая среда существования информации, поэтому, естественно, существует и специальное программное обеспечение, "заточенное" именно под его нужды. Осваивая его, мы попутно рассмотрим механизм его работы, и после этого сможем корректировать полученный результат для наиболее полного и адекватного выражения нашего замысла. Это действительно необходимо, так как полностью автоматическое создание какого-либо продукта средствами программного обеспечения редко приносит желаемый результат. Вы помните, как выглядели ваши документы, когда в старых версиях Microsoft Word вы применяли к ним команду автоматического форматирования? Мне, например, это никогда не нравилось. Документ получался слишком невзрачным, и его оформление абсолютно не соответствовало моему замыслу. Прогресс, конечно, не стоит на месте, и результат работы программ все больше похож на то, чего хотел добиться пользователь, но полностью полагаться на них все-таки не стоит.

Сначала мы научимся создавать отдельные Web-странички, а затем начнем объединять их в сайты. Для этих целей мы будем применять программный комплекс Microsoft FrontPage 2000. Мы рассмотрим особенности правильного построения сайтов, проблемы, связанные с графикой, различные мультимедийные варианты наполнения сайта, попробуем научиться находить баланс между содержательной частью страницы и скоростью ее загрузки. Затем мы научимся делать промышленные сайты, связанные с теми данными, которые используются в повседневной работе. То есть мы рассмотрим набор процедур и решений, которые позволят привязать сайт к корпоративной базе данных и организовать правильную работу с ней в дистанционном режиме. Мы увидим, как организуется работа с клиентами при помощи сайта и как сотрудники организации могут выполнять свою работу, не находясь непосредственно в офисе. Мы узнаем, как организовать средства безопасности нашего сайта, на котором может находиться большое количество ценной информации. Нас ждет также процедура создания сайта для электронной коммерции, в которой мы используем программу Web-Shop 2000 от Boomerang Software версии 5.10. И я еще раз подчеркиваю, мы обойдемся без программирования. Все, что я предложу вам в этой книге, может сделать любой человек, который уже умеет работать на компьютере.

Вам не кажется, что вступление несколько затянулось? На мой взгляд, нам уже пора перейти к тому, что мы собираемся делать — творить Web-сайт своими руками.

Глава 1

От страницы к сайту

Как это все устроено

Что же происходит в тот момент, когда мы набираем адрес сайта в строке Address или Location нашего браузера? Как мы получаем содержимое страниц сайта в своем браузере? Как их готовят разработчики? В каком виде они хранятся? Ответы на все эти вопросы мы попробуем получить в этой главе.

Если вы пользуетесь Интернетом, то наверняка слышали аббревиатуру HTML, которая расшифровывается как HyperText Markup Language. Это язык, который применяется для создания Web-страниц. Как видно из его названия, он не является языком программирования. Это язык разметки документов, то есть определенный набор инструкций, которые вставляются в текст документа и указывают на то, каким образом необходимо отобразить тот или иной объект этого документа. Сами эти инструкции называются тегами. Нет нужды знать их все наизусть, но основные теги мы рассмотрим в первой главе книги. С помощью этих тегов производится форматирование текста, вставка рисунков, ссылок и активных элементов, оформляются мультимедийные части документов.

Самое главное слово в названии языка — Hypertext. То есть язык разработан специально для осуществления ассоциативных связей. К любому понятию и любому слову документа мы можем привязать ссылку на другой документ, который может разъяснять понятие или слово, толковать и расширять его. Сама по себе эта технология не являлась каким-либо значительным прорывом. В программном заявлении консорциума по WWW было сказано приблизительно следующее: "Гипертекст не делает чего-то такого, что нельзя было сделать без него. Он просто позволяет делать это быстрее". А потом из этой "непрорывной" технологии вырос Интернет, который стал, пожалуй, самым значительным явлением века. Его истинная ценность не в каких-либо технологиях или удобствах. Интернет является средством общения для множества людей. И чем больше людей подключается к этой всеобщей сети, тем выше становится ее ценностью. Боб Меткалф, один из

основателей фирмы 3Com, описывал это явление следующим образом: "Зачем был факс первому покупателю факсовой машинки? Кому он мог пересылать сообщения? А сегодня каждый покупатель факса покупает вместе с ним целую факсовую сеть. Следовательно, ценность (но не цена) каждого факса становится больше."

Итак, как мы уже знаем, сама Web-страничка хранится в виде текста, "разбавленного" тегами HTML. Когда мы соединяемся со своим провайдером и набираем адрес страницы или сайта, а точнее их URL (Universal Resource Location), то сервер провайдера переводит буквенный URL в цифровой адрес (IP-address), соединяется с сервером, имеющим этот адрес, и на его локальном диске находит файл, содержащий искомую страничку. После этого содержимое найденного файла пересылается на ваш компьютер как обычный текстовый файл с помощью протокола HTTP (HyperText Transfer Protocol), а браузер, запущенный на локальной машине, получает этот файл, обрабатывает его, находя в тексте теги HTML, формирует изображение странички и отображает результат. Щелчок кнопкой мыши на любой из гиперссылок этой страницы заставит весь процесс повторяться с самого начала.

Давным-давно, во время становления World Wide Web, странички и сайты делались при помощи обычных текстовых редакторов. Так можно делать и сейчас. Но не стоит. Писать HTML-код, потом загружать страницу в браузер, просматривать ее, находить ошибки, снова возвращаться в редактор, исправлять ошибки. И так раз за разом. Ужасно непроизводительно. Поэтому вскоре начали появляться специализированные HTML-редакторы и редакторы для проектирования Web-страниц в режиме WYSIWYG (What You See Is What You Get). Они позволили автоматизировать процесс создания информационных ресурсов Интернета. В первой главе мы рассмотрим один из таких редакторов — Microsoft FrontPage 2000.

Помимо страничек и сайтов, которые передаются при помощи протокола HTTP, есть и другие виды ресурсов, которые также интегрированы в Интернет. То есть, World Wide Web это не весь Интернет, как это часто ошибочно полагают, а лишь его часть. Есть еще службы для копирования файлов, которые используют протокол FTP (File Transfer Protocol) и, конечно, нельзя забывать про электронную почту, которая обычно работает при помощи протоколов POP3 и SMTP. Прелесть всей работы сейчас заключается в том, что нам не надо вникать в тонкости их реализации, так как они безболезненно интегрируются в возможности стандартных браузеров, а значит, и подключение подобных ресурсов к создаваемым страницам и сайтам не будет представлять больших трудностей.

Однако, перед тем как применять эти возможности, следует научиться создавать обычные Web-странички.

Знакомство с FrontPage

Слово FrontPage обычно используется для обозначения основной, заглавной страницы какого-либо сайта. Для нас это слово будет иметь еще один смысл. Программа, которой мы будем пользоваться для создания своих первых страниц, носит именно такое название — Microsoft FrontPage 2000. Полный номер версии — 4.0. FrontPage входит в состав Microsoft Office и по своему стилевому оформлению полностью соответствует его стандартам. Данная программа достаточно проста в работе, но несмотря на это позволяет делать очень и очень многое.

Стандартное окно FrontPage 2000 показано на рис. 1.1.

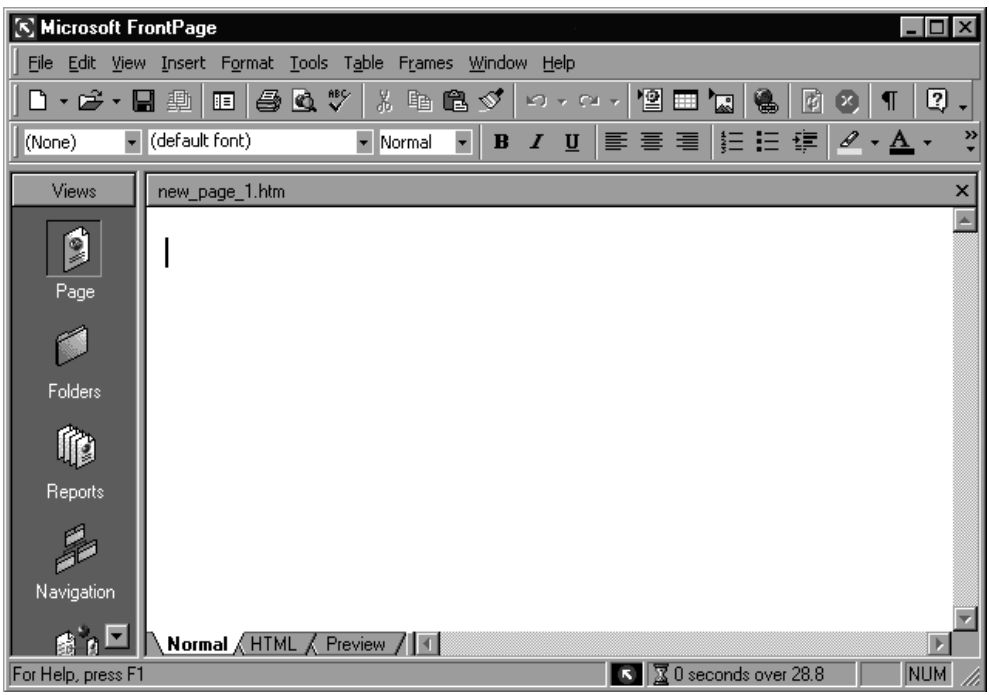

Рис. 1.1. Внешний вид приложения FrontPage 2000

Посмотрим, какими средствами для работы мы располагаем. Как и во всех продуктах Microsoft Office, здесь присутствуют меню и панели инструментов. Особых сложностей их освоение вызвать не должно. Основное рабочее поле разделено подвижной границей (сплиттером) на две неравные части. Левая часть с названием **Views** содержит кнопки, устанавливающие режим работы FrontPage. Правая часть содержит основную информацию, с которой приходится работать, находясь в указанном режиме. По умолчанию установлен режим **Page**, который предназначен для проектирования страницы. В этом режиме правая часть отображает ту страницу, которую создает пользователь. Как

видно на рисунке, правая часть рабочего пространства состоит из трех листов (вкладок), каждый из которых предназначен для различных режимов отображения проектируемой страницы. Лист **Normal** показывает страницу в режиме проектирования, **HTML** предназначен для просмотра HTML-кода страницы, а вкладка **Preview** показывает проектируемую страницу в том виде, в каком она будет отображаться браузером при загрузке ее удаленным пользователем. Может показаться, что страницы **Normal** и **Preview** показывают одно и то же, но на самом деле это не всегда так. Для простых страничек, содержащих только текст и графические изображения они действительно будут одинаковыми, но как только мы попробуем разместить на нашей странице видеофрагменты или какие-либо активные элементы, мы тут же заметим разницу между этими двумя режимами.

В самом низу окна **Microsoft FrontPage** находится строка статуса. На первый взгляд в ней нет ничего особенного. На самом деле строка статуса содержит очень интересную и необходимую для сайтостроителя информацию. В разделе с изображением песочных часов показывается время загрузки текущей страницы при определенной скорости связи. По умолчанию используется скорость 28,8 Кбод, однако всегда есть возможность изменить ее. Для этого достаточно щелкнуть мышью (причем, что интересно, любой кнопкой) и получить список различных скоростей. Помимо стандартных скоростей 14,4, 28,8 и 56,6, там можно найти возможность расчета скорости загрузки исходя из параметров ISDN, а также стандартов T1 и T3 [1].

Как видно, при первом знакомстве FrontPage не отпугивает пользователя своей сложностью. Конечно, при работе с ним будут определенные тонкости, но мы их обязательно рассмотрим. Парадигма же работы в FrontPage ничем не отличается от работы в обычном MS Word. Разместите текст и элементы оформления на странице так, как вы хотите, а для внесения изменений выделите объект и примените к нему необходимое действие.

FrontPage позволяет применять к тексту практически любое шрифтовое и стилевое оформление. Ограничения на формат текста накладывает не программа, а сама технология. К примеру, мы не можем жестко указывать шрифт текста. Связано это с тем, что никогда нельзя точно предсказать, какие шрифты установлены на машине удаленного пользователя. Более того, неизвестно, какую операционную систему он использует и какое разрешение экрана у него установлено. Поэтому вместо точного указания наименования и размера шрифта указывается семейство шрифтов и относительный размер. Более подробно о стилевом и шрифтовом оформлении страницы мы будем говорить в следующем разделе.

Для начала попробуем сделать самую простую страницу. Разместим на рабочем поле проектирования страницы любую строку текста. Например,

[1] Стандарты ISDN, T1 и T3 применяются для выделенных линий с различной пропускной способностью.

"Это моя первая Web-страница!". При этом, если мы посмотрим, что появится на вкладке **HTML**, то обнаружим там следующее:

```
<html>
<head>
<meta http-equiv="Content-Language" content="ru">
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<title>New Page 1</title>
</head>
<body>
<p>Это моя первая Web-страница!</p>
</body>
</html>
```

На первый взгляд, этот текст может показаться достаточно непонятным. Действительно, нашу строку текста еще можно опознать, но вот что там делает вся остальная абракадабра? Нет, это не абракадабра, а те самые теги HTML, которые мы упоминали немного ранее. Давайте разберемся, для чего они предназначены.

Прежде всего, необходимо отметить, что большинство тегов, ответственных за правильное отображение страницы или за надлежащее оформление ее содержания являются парными. Они бывают объявляющими и закрывающими. То есть первый тег объявляет какое-либо оформление или раздел страницы, а закрывающий отменяет, заканчивает его действие. Каждый тег HTML заключен в угловые скобки, а закрывающие теги еще имеют в начале своего имени обратный слеш.

Так, например, каждый HTML-документ должен начинаться с тега `<html>`, а заканчиваться — `</html>`, что мы и видим в нашем примере. Сам документ обычно состоит из трех частей. Первая — заголовок, несущий служебную информацию, которая позволяет браузеру максимально адекватно отображать документ. Эта часть ограничивается тегами `<head>` и `</head>`. Внутри нее помещаются теги `<meta>`, содержащие в своем теле параметры. Например, тег

```
<meta http-equiv="Content-Language" content="ru">
```

содержит параметры `http-equiv` и `content`. Давайте подробнее рассмотрим параметры этого тега. Параметр `http-equiv` предназначен для обозначения переменной протокола HTTP, которой будет назначаться некоторое значение. Обычно HTML-редакторы сами заполняют значения этого параметра. В нашем примере эта переменная носит наименование `Content-Language` и предназначается для указания языка, на котором написано содержимое странички. Параметр `content` предназначается для задания значения этой переменной. Таким образом, первая строка блока заголовка указывает, что наша страничка содержит русскоязычное наполнение.

Параметр `content` может использоваться в паре с параметром URL, и тогда они будут определять дату и время повторной загрузки документа. Так, например, если мы хотим спустя пять секунд после загрузки удаленным пользователем нашей странички принудительно загрузить в его браузер HTML-документ с URL `http://www.myserver.ru/newpage.html`, то мы должны использовать следующую конструкцию:

```
<meta http-equiv="Refresh" content="5;
url = http://www.myserver.ru/newpage.html>
```

Параметр `name` применяется для задания наименования дополнительной информации, помогающей браузеру более адекватно отображать Web-страницу. Так, в теге `<meta name="GENERATOR" content="Microsoft FrontPage 4.0">` он применяется для того, чтобы указать имя параметра, чье значение объявляется в конструкции `content`. В этой строке указывается, что документ был сгенерирован программой FrontPage версии 4.0.

И последний параметр тега `<meta>` — `charset`. Он предназначен для определения кодовой страницы символов, которая использовалась при создании страницы. Так как FrontPage естественным образом ориентирован на операционную среду Microsoft Windows, то и используемая по умолчанию кодовая страница — windows-1251.

Второй раздел HTML-документа предназначен для задания названия странички. То есть, той строки, которая будет отображаться в заголовке окна браузера. Этот раздел ограничен тегами `<title>` и `</title>`. Как видно из нашего HTML-кода для нашей странички использован заголовок `New Page 1`. Не слишком оригинально, на мой взгляд, но мы всегда можем его изменить.

Последний, основной раздел документа, заключен между тегами `<body>` и `</body>` и используется для организации содержимого Web-страницы. Сама строка, как мы видим, обрамлена тегами `<p>` и `</p>`, которые предназначаются для выделения абзацев.

Вот так и состоялось наше первое знакомство с языком HTML.

После создания любой странички, ее необходимо сохранить, а потом просмотреть, чтобы убедиться в адекватном отображении ее браузером. Для сохранения используется команда меню **File/Save** и **File/Save As**, либо соответствующая кнопка на инструментальной панели. Обычно в начале работы FrontPage создает необходимую структуру каталогов в папке **My Webs**, которая в свою очередь находится в каталоге, зарезервированном для сохранения документов. В эту структуру входят каталоги **images**, предназначенный для хранения графических изображений, и **_private**, в котором размещаются данные, выкладываемые на создаваемый сайт. После сохранения страницы, основное рабочее поле делится сплиттером на две части, в одной из которых остается проектируемая страница, а вторая показывает структуру папок, предназначенных для сохранения сайта. Этот режим отображения включается и выключается при помощи команды меню **View/Folder List** или при

помощи кнопки **Folder List** на основной инструментальной панели. Сохранение страницы является необходимым условием для возможности ее просмотра как при помощи встроенного браузера на странице **Preview**, так и в основном браузере системы. В том случае, если в оформлении страницы были использованы какие-либо активные элементы, необходимо подготовить сайт к просмотру. Этот процесс называется "публикацией". Для этих целей используется команда меню **File/Publish Web** или соответствующая кнопка на инструментальной панели. Публикацию можно производить только после того, как были сохранены все страницы, которые входят в создаваемый сайт.

После сохранения странички необходимо увидеть, как она будет выглядеть в окне браузера удаленного пользователя. Для этого применяется команда меню **File/Preview in Browser** или соответствующая кнопка на основной инструментальной панели. Использование этой команды позволяет просмотреть созданную страницу при помощи браузера, являющегося для системы основным, используемым по умолчанию. Возникает законный вопрос, для чего это нужно, если уже есть страница **Preview** в основном рабочем окне. Дело в том, что разные браузеры могут по-разному отображать одну и ту же страницу. Связано это с тем, что во время "браузерных войн" между фирмами Microsoft и Netscape предлагались различные расширения и усовершенствования языка HTML. Естественно, что продвинутые возможности, которые предлагала одна фирма, не поддерживались второй. По наследству эти различия перешли и в последующие версии. Вторая причина заключена в том, что встроенный браузер FrontPage не всегда может адекватно работать с исполняемыми модулями CGI-приложений (CGI – Common Gateway Interface). Именно поэтому при заключительном тестировании созданной страницы настоятельно рекомендуется использовать просмотр при помощи основного браузера.

Ну что ж, мы рассмотрели основные принципы работы с программой Microsoft FrontPage 2000. Пора присмотреться к ней повнимательней, и понять, что мы можем сделать с ее помощью, а главное, как это сделать.

Оформление текста

Что главное в Web-странице? Не дизайн, не графика, и даже не видеовставки или активные элементы. Главное — это ее содержание, которое состоит, прежде всего, из текста. А вот оформление текста позволит наиболее эффективно донести его до потребителя, за чье внимание борется каждый сайт. Конечно, возможности текстового оформления, предоставляемые FrontPage, не так обширны, как, скажем, запас средств Microsoft Word, но нельзя сказать, что они недостаточны. Как мы уже говорили, ограничения на оформление текста налагает сама технология WWW, а из допустимых возможностей FrontPage предоставляет максимальные.

Панель инструментов для форматирования текста практически полностью повторяет такую же панель Microsoft Word. Но отличия все-таки есть. Начнем с установки шрифта. По умолчанию действует набор **default font**, который может быть изменен. Список шрифтов, которые можно применять, конечно же немного меньше, чем такой же список в других приложениях Microsoft Office, но выбор все-таки есть. Давайте для нашей строки установим какой-либо шрифт из группы Arial. При этом мы увидим, что изменился шрифт строки на вкладке **Normal**, а когда мы перейдем на вкладку **HTML**, то увидим следующую конструкцию:

```
<p><font face="Arial">Это моя первая Web-страница!</font></p>
```

После тега `<p>`, открывающего абзац, добавился тег `<font face="Arial">`. Теперь при отображении этого абзаца на компьютере удаленного пользователя система будет извещена, что необходимо использовать шрифт, максимально близкий по своему виду к шрифту Arial. То есть, замысел дизайнера Web-страницы пострадает не слишком сильно, даже если сам шрифт Arial не будет установлен в системе удаленного пользователя.

Помимо самого шрифта, мы можем выбрать и его начертание. То есть, сделать его полужирным, курсивом, подчеркнутым или выбрать любую комбинацию этих признаков. Для этого обычно используются три всем знакомые кнопки, находящиеся сразу после списка выбора размера шрифта. Итак, если мы для нашей строки установим полужирное начертание шрифта нажатием кнопки **Bold**, то при просмотре кода на вкладке **HTML** наша строка будет записана как `<p><b>Это моя первая Web-страница!</b></p>`. То есть, для установки полужирного атрибута текста используются теги `<b>` и `</b>`, которые устанавливаются внутри тегов, обозначающих абзац. Если мы нажмем кнопку **Italic** для получения курсива, то HTML-код будет выглядеть как `<p><i>Это моя первая Web-страница!</i></p>`. А для получения подчеркнутого текста мы должны нажать кнопку **Underline** и при этом строка реализуется с помощью конструкции `<p><u>Это моя первая Web-страница!</u></p>`. Как видим, для курсива используются теги `<i>` и `</i>`, а для подчеркивания — теги `<u>` и `</u>`.

Теперь рассмотрим процедуру выбора размера шрифта. По умолчанию в списке размеров шрифта стоит значение **Normal**. Так как никогда нельзя заранее сказать, какое разрешение монитора будет установлено на машине удаленного пользователя, и каков будет размер окна браузера, то и размеры шрифта будут лишь относительными. К тому же, привычные нам всем пункты, которые применяются в качестве единицы измерения размера шрифта, являются типографскими единицами, то есть ориентированы, прежде всего, на бумагу, а Web-страницы отображаются на мониторах, и поэтому размер шрифта будет зависеть от разрешения видеосистемы пользователя и размера окна браузера. Итак, мы можем использовать семь относительных размеров шрифта. Первый размер приблизительно соответствует восьми типографским пунктам, а седьмой — тридцати шести. Значение по умолчанию **Normal** соответствует третьему размеру, который составляет

приблизительно двенадцать пунктов. Таким образом, мы имеем диапазон размеров, достаточный, на мой взгляд, для оформления Web-страницы.

Две вышеописанные возможности объединяет в себе процедура стилевого оформления текста. В FrontPage 2000 список доступных стилей находится на панели форматирования текста в том же самом месте, где и его близнец в Microsoft Word. Но сама процедура стилевого оформления немного отличается. Word в качестве стиля использует совокупность признаков текста, таких как шрифт, его начертание и размер, межстрочный интервал, отступы абзаца и тому подобные параметры. FrontPage скован рамками технологии WWW, поэтому форматирование текста в нем опирается на теги, которые объявлены в стандартах языка HTML и поддерживаются браузерами. Правда, возможно, не всеми из них. Это, как мы помним, последствия браузерных войн. Итак, каждый образец форматирования текста объявляется конкретным тегом языка HTML. Рассмотрим примеры, а заодно узнаем, как выглядят эти теги для каждого конкретного случая.

Стиль **Normal** предполагает обычный текст, и он является единственным образцом стилевого оформления текста, который не устанавливает тегов перед текстом. Стиль **Formatted** предназначен для размещения предварительно отформатированного текста. В данном контексте это означает, что текст будет отображен моноширинным шрифтом, то есть таким шрифтом, в котором все символы имеют одинаковую ширину. Обычно, в качестве примера подобного шрифта приводится Courier. Если мы применим этот стиль к нашей строке, то на странице **HTML** мы увидим следующую конструкцию:

```
<pre>Это моя первая Web-страница!</pre>
```

Как видно, текст обрамляется тегами `<pre>` и `</pre>`. На самом деле, этот тег изначально предназначался для вставки текста, подготовленного в другом текстовом редакторе. Причем, неизвестно заранее, какая длина строки была установлена в этом текстовом редакторе, поэтому, чтобы не нарушать оформление текста, у тега `<pre>` был введен параметр `width`, который указывает длину строки. Таким образом, тег `<pre width=60>` указывает, что для любого размера окна браузера при просмотре этого текста необходимо установить длину строки в 60 символов. По умолчанию значение этого параметра равно 80.

Стиль **Address** используется, как легко догадаться, для оформления адресов. Он переводит строку, напечатанную обычным шрифтом, в отображение курсивом. Применение этого стиля приводит к следующему HTML-коду:

```
<address>
Это моя первая Web-страница!
</address>
```

Как мы видим, отображение адреса на Web-страницах производится при помощи тегов `<address>` и `</address>`. Все просто и легко.

Следующие 6 стилей с названиями от **Heading1** до **Heading6** предназначены для отображения заголовков различных уровней. Самый главный, естественно,

стиль **Heading1**, который и отображается наиболее массивным шрифтом с размером 6, который соответствует приблизительно 24 пунктам, а стиль **Heading6** отображается шрифтом с размером 1, то есть всего-навсего 8 пунктов. При применении стиля **Heading1** к нашей строке, мы увидим, что она будет обрамлена тегами `<h1>` и `</h1>`. На вкладке HTML это будет выглядеть как `<h1>Это моя первая Web-страница!</h1>`. Если мы используем стиль **Heading2**, то теги будут иметь вид `<h2>` и `</h2>`. Таким образом, мы видим, что номер заголовка записывается в теге после буквы h.

В том случае, если мы намерены изменить параметры стиля, и указать для него конкретный шрифт и размер, вместо его стандартных параметров, которые используются, когда в списках шрифта и размера стоят значения **default font** и **Normal** соответственно, это не так сложно сделать. Например, если мы к нашей строке применим стиль заголовка третьего уровня, а затем принудительно установим шрифт Arial и размер 3, то HTML-код для отображения нашей строки будет выглядет так:

```
<h3><font face="Arial" size="3">Это моя первая Web-страница! </font>
</h3>
```

Как видно, сразу после тега `<h3>` вставляется тег `<font>` с параметрами `face` и `size`, которые непосредственно задают шрифт для отображения строки. И в конце предложения стоит закрывающий тег `</font>`.

Следующие шесть стилей предназначены для оформления текста в виде нумерованных и маркированных списков. Собственно, нумерованный список там только один, а остальные пять стилей являются вариациями маркированного списка.

Для иллюстрации действия этих стилей создадим два абзаца со строками "Первый пункт" и "Второй пункт", а затем применим к этим двум абзацам стиль **Numbered List**. Как мы видим, в результате этого действия каждый абзац получил свой номер. То есть произошла стандартная операция нумерации абзацев, знакомая нам еще по Microsoft Word. Этого же результата можно добиться, нажав кнопку **Numbering** на панели инструментов **Formatting**. Рассмотрим теги, при помощи которых реализуется подобное форматирование текста. Если мы посмотрим на вкладку **HTML**, то увидим следующий текст:

```
<ol>
  <li>Первый пункт</li>
  <li>Второй пункт</li>
</ol>
```

Как мы видим, начало и конец нумерованного списка задаются с помощью тегов `<ol>` и `</ol>`, а обозначение каждого пункта производится тегами `<li>` и `</li>`. Все не так уж и сложно.

Хотелось бы также узнать, каким образом мы можем изменить вид нумерации и начальный номер. Для этого используется пункт меню **Format/Bullets and Numbering**. В том случае, если у нас на странице курсор находится

в строке, принадлежащей нумерованному списку, активизируется диалоговое окно **List Properties** на вкладке **Numbers** (рис. 1.2). По умолчанию выбран вариант с арабскими цифрами. Выбор варианта списка без нумерации приведет к полному снятию стиля **Numbered List** с нашего списка, и, соответственно, незамедлительное его превращение в самый обычный текст со стилем **Normal**. В случае выбора списка с нумерацией в виде римских цифр, отображаемых при помощи заглавных букв латинского алфавита, наш список будет открыт тегом `<ol type="I">` и закрыт тегом `</ol>`. Других изменений нет. Становится видно, что вид нумерации задается параметром `type` тега `<ol>`. Например, для создания списка, в котором нумерация осуществляется при помощи заглавных букв латинского алфавита (A, B, C и т. д.), используется параметр `type="A"`. Если это должны быть обычные строчные буквы, то параметр принимает вид `type="a"`. И, наконец, если нумерация списка осуществляется римскими цифрами, отображаемыми строчными латинскими буквами, параметр указывается как `type="i"`.

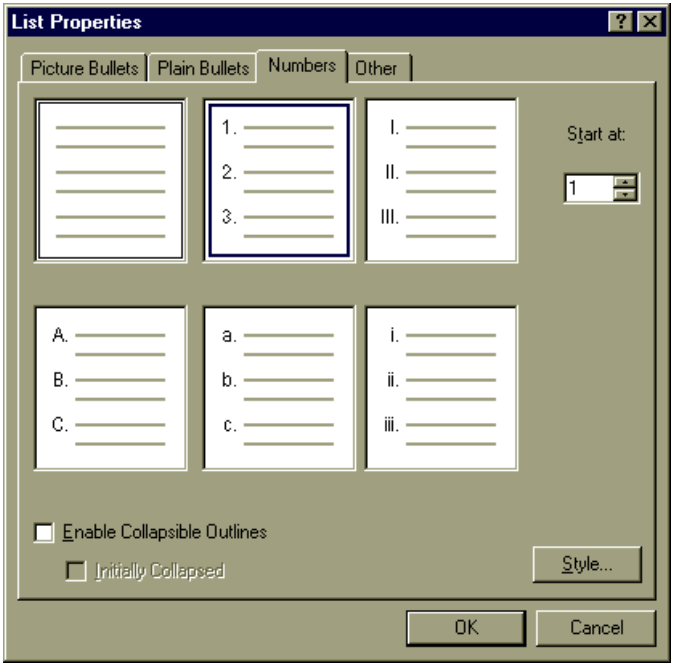

Рис. 1.2. Вкладка **Numbers** диалогового окна **List Properties**

Еще один параметр списка — это его стартовый номер, то есть число, с которого начинается нумерация. Оно устанавливается в поле **Start at** окна **List Properties**. При этом в тег `<ol>` добавляется параметр `start`. То есть, если мы хотим сделать список, нумерованный римскими цифрами из строчных букв, начинающийся с номера 5, то для этого будет использован тег `<ol type="i" start="5">`.

Теперь рассмотрим маркированные списки. Для их создания наиболее часто применяется стиль **Bulleted List** или кнопка **Bullets** на панели инструментов **Formatting**. При применении этого стиля к нашему тексту мы увидим, что для его реализации используется следующая конструкция HTML:

```
<ul>
  <li>Первый пункт</li>
  <li>Второй пункт</li>
</ul>
```

Как и в предыдущем примере, здесь есть теги, объявляющие сам список, то есть `<ul>` и `</ul>`, а пункты списка оформляются при помощи открывающего тега `<li>` и его закрывающего дополнения `</li>`. Так же, как и нумерованный, этот вариант списка может быть оформлен несколькими видами маркеров. Их вид выбирается в диалоговом окне **List Properties** на вкладке **Plain Bullets**, которое активируется при выборе команды меню **Format/Bullets and Numbering**. Но вариантов оформления там меньше, чем в нумерованном списке. Если не считать пустое оформление, остается всего три. Маркеры в виде точек устанавливаются по умолчанию. Мы можем изменить вид маркеров и оформить их в виде окружностей. Тогда тег, объявляющий начало маркированного списка, будет записан как `<ul type="circle">`. А если мы захотим увидеть маркеры в виде квадратов, то HTML-код, реализующий эту прихоть, будет выглядеть как `<ul type="square">`. Впрочем, для отображения маркеров мы можем использовать не только эти три зарезервированные символа, но и практически любое графическое изображение. Для этого все в том же диалоговом окне **List Properties** необходимо выбрать вкладку **Picture Bullets**, которая предназначена для установки внешнего вида маркеров. Проектировщику Web-страницы предоставляется на выбор две альтернативы (рис. 1.3). По умолчанию действует пункт **Use pictures from current theme**. Он указывает на то, что при оформлении страницы будут использоваться изображения маркеров, которые применяются в так называемой *теме* сайта, то есть наборе графических изображений и правил оформления текстов, которые будут употребляться для единообразного оформления всех страниц, входящих в состав сайта. Однако, если у создателя страницы есть свои соображения по поводу формы маркеров, то он должен выбрать **Specify picture**, и указать путь к графическому файлу, который он хочет использовать, при помощи кнопки **Browse**. При этом активизируется стандартное диалоговое окно (рис. 1.4).

Естественно, рекомендуется выбирать изображения из числа тех, которые находятся в пределах структуры папок, предназначенных для хранения файлов создаваемого сайта. Однако проектировщик Web-страницы может выбрать необходимое изображение тремя различными способами. Первый — наиболее привычный для всех нас выбор файла из локальной файловой системы. Это можно сделать либо простым указанием файла в структуре

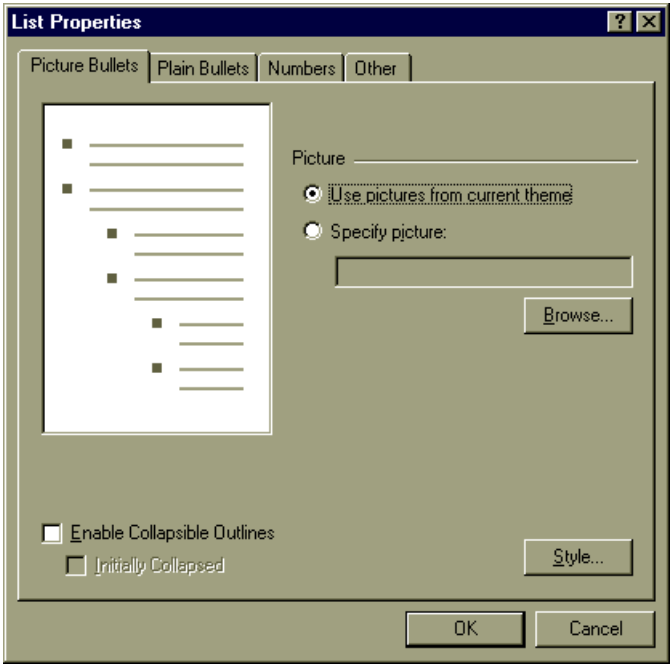

Рис. 1.3. Вкладка **Picture Bullets** диалогового окна **List Properties**

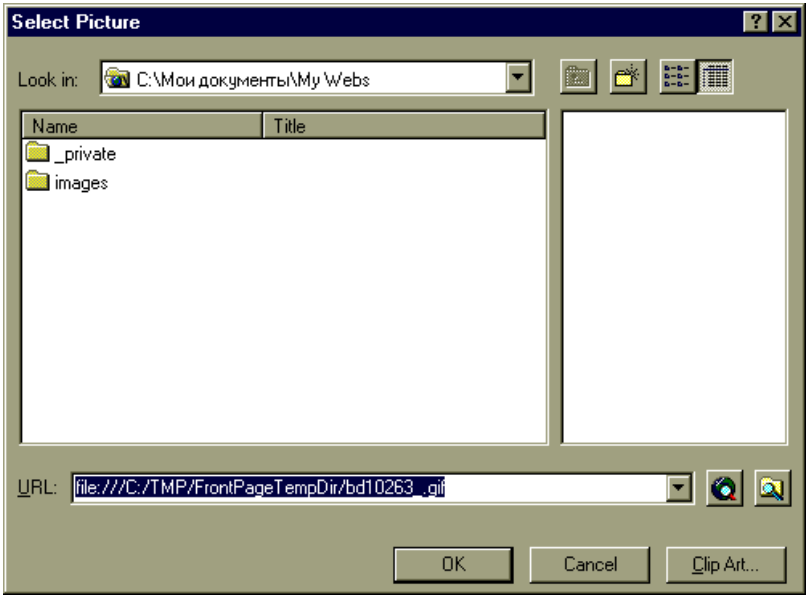

Рис. 1.4. Диалоговое окно **Select Picture**

папок сайта или, если файл находится за ее пределами, нажатием кнопки с изображением папки и лупы, которая позволит просмотреть любую папку

на локальном компьютере. Второй вариант связан с использованием изображения, которое находится на удаленном компьютере. Нажатие кнопки с изображением земного шара и лупы запускает браузер, установленный в системе, и пользователь может выбрать любой графический элемент, который он найдет в Интернете. При этом будет вставлен не сам этот элемент, а ссылка на него. Каждый раз при загрузке нашей страницы браузер будет вынужден обращаться по указанному адресу и скачивать оттуда изображение. Вопрос о целесообразности использования этого варианта целиком и полностью остается в ведении проектировщика. Третий вариант — выбор изображения из коллекции Clip Art, поставляемой вместе с Microsoft Office. Для этого необходимо нажать кнопку **Clip Art**. При этом файл с выбранным вами изображением будет перенесен во временную папку Microsoft FrontPage 2000. В этом случае тег, ответственный за объявление маркированного списка, получит дополнительный параметр и примет вид:

```
<ul imagesrc="file:///C:/TMP/FrontPageTempDir/bd10263_.gif">
```

Как мы видим, в параметре `imagesrc` все равно указывается URL файла с графическим изображением, но вместо протокола HTTP используется протокол file, который предназначен для работы с файловой системой локального компьютера.

Помимо обычного маркированного списка, FrontPage предоставляет также несколько других видов списков. Они не отличаются оформлением от обычного списка. Различие кроется в тегах, реализующих их. Стиль **Directory List** задается при помощи следующего HTML-кода:

```
<dir>
  <li>Первый пункт</li>
  <li>Второй пункт</li>
</dir>
```

Стиль **Menu List** открывается и закрывается при помощи тегов `<menu>` и `</menu>` соответственно. Элементы этого стиля отображаются абсолютно идентично элементам обычного маркированного списка. Наличие нескольких стилей с одинаковым отображением по умолчанию позволяет при помощи каскадных таблиц стилей (CSS) создавать различные варианты отображения элементов ненумерованных списков.

Для списка, который состоит из определений, предназначен стиль **Definition List**. Данным вариантом оформления часто пользуются в академических и обучающих материалах, когда необходимо в списке поместить несколько определений. В случае его применения меняются не только открывающий и закрывающий теги, но и теги, обрамляющие каждый отдельный пункт списка. Таким образом, получается следующий HTML-код:

```
<dl>
  <dd>Первый пункт</dd>
  <dd>Второй пункт</dd>
</dl>
```

При этом, каждый пункт приобретает стиль **Definition**. Его признаки, как мы видим, это теги `<dd>` и `</dd>`.

Задание шрифта, его начертания, размера и стилевого оформления текста может быть произведено при помощи диалогового окна **Font**, которое активизируется при выборе команды меню **Format/Font**.

Для оформления текста также применяются различные выключки и отступы. *Выключкой* или *выравниванием* называется расположение текста относительно горизонтальной базовой линии страницы. Применяется четыре вида выравнивания текста: текст, прижатый к левому краю, отцентрированный текст, текст, прижатый к правому краю, и текст, растянутый на всю ширину страницы. В FrontPage 2000 на панели инструментов **Formatting** доступны кнопки **Align Left**, **Center** и **Align Right**, которые реализуют первые три варианта выравнивания текста. Текст, прижатый к левому краю, является стандартным вариантом расположения, но может быть задан и явно. Выравнивание задается при помощи параметра тега, объявляющего абзац `<p>`. Так, для явного объявления выключки текста по левому краю применяется тег `<p align="left">`. В случае отцентрированного текста используется конструкция `<p align="center">`. И, как нетрудно догадаться, прижатый к правому краю текст объявляется при помощи тега `<p align="right">`. Из приведенных примеров видно, что выключка текста задается применительно к целому абзацу, и действие тегов выравнивания текста прекращается стандартным тегом окончания абзаца `</p>`. Для равномерного растяжения текста по всей ширине страницы специальной кнопки нет. Чтобы его использовать, необходимо выполнить команду **Format/Paragraph**, которая активизирует диалоговое окно **Paragraph** (рис. 1.5). Это окно предназначено для задания параметров абзаца, а если говорить конкретнее, для задания отступов и интервалов. Выравнивание текста указывается в списке **Alignment**. Для того чтобы равномерно растянуть текст абзаца на всю длину строки, необходимо выбрать пункт **Justify**. При этом, тег объявления абзаца примет вид `<p align="justify">`. А выбор элемента **Default** в списке **Alignment**, полностью убирает параметр `<p>`, отвечающий за назначение выравнивания из тела тега `<p>`.

Следующий блок органов управления **Indentation** диалогового окна **Paragraph** предназначен для указания отступов текста от краев страницы. Поле ввода **Before text** позволяет указывать отступ абзаца от левого края страницы, а поле **After text** — от правого. Поле ввода **Indent first line** указывает отступ красной строки, то есть, первой строки каждого абзаца. Если мы укажем отступ перед текстом как три единицы, после него — четыре, а для красной строки — пять единиц, то тег объявления абзаца примет вид `<p style="text-indent: 5; margin-left: 3; margin-right: 4">`. То есть, тег `<p>` получает параметр `style`, значением которого будет одна длинная строка, в которой сначала указывается отступ красной строки, а затем отступы всего абзаца слева и справа. По умолчанию в качестве единиц измерения используются миллиметры.

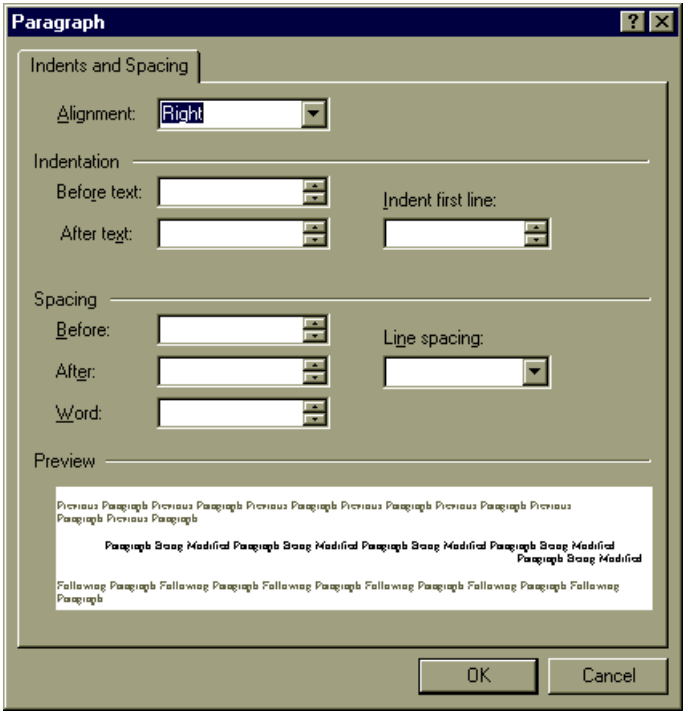

Рис. 1.5. Диалоговое окно **Paragraph**

Отступы текста от края страницы могут быть изменены при помощи кнопок **Decrease Indent** и **Increase Indent**, которые позволяют уменьшать и, соответственно, увеличивать отступ текущей строки. Увеличение отступа текста осуществляется при помощи отдельных тегов. Для уменьшения отступа (который до этого был увеличен) достаточно удалить эти теги. Если для нашей строки отступ не задан явным образом и мы воспользуемся кнопкой **Increase Indent** для его установки, то при просмотре HTML-кода для отображения нашей строки, мы увидим следующую конструкцию:

```
<blockquote>
  <p>Это моя первая Web-страница</p>
</blockquote>
```

При двойном нажатии этой кнопки добавится еще одна пара тегов `<blockquote>` и т. д.

Есть еще один способ позиционирования текста — при помощи клавиши табуляции. Так как для Web-страниц не создано специальных табуляторов, то для отработки табуляции применяется совокупность неразрывных пробелов. При единичном нажатии клавиши табуляции в начале строки, строка немного сдвигается вправо, а HTML-код для нашего абзаца принимает вид `<p>    Это моя первая Web-страница</p>`. Как видно, после тега, объявляющего начало абзаца, вставляется три символьные после-

довательности, которые обозначают неразрывные пробелы. Подобные последовательности применяются для отображения всех нестандартных символов и тех символов, которые зарезервированы для языка HTML, то есть угловых скобок, амперсантов и двойных кавычек.

Последний блок органов управления диалогового окна для установки атрибутов абзаца носит название **Spacing** и предназначен для установки различных интервалов. Поле ввода **Before** позволяет указать интервал перед началом абзаца, а поле **After**, естественно, интервал после него. Необходимо отметить, что если эти параметры применяются к нескольким абзацам, то расстояние между ними будет складываться из отступа **After** одного абзаца и отступа **Before** последующего абзаца. Поле ввода **Word** предназначается для указания величины пробела между отдельными словами, а список **Line spacing** позволяет указывать межстрочный интервал. Для него используется три значения: **Single**, который обозначает обычный отступ, **1.5 lines** — для установки полуторного отступа, и **Double** — для двойного.

Если мы установим значения в этом блоке и применим их к текущему абзацу, нажав кнопку **OK**, то реализация этих условий будет выполнена при помощи тега

```
<p style="word-spacing: 4; line-height: 150%; margin-top: 5; margin-bottom: 5">
```

где `word-spacing` указывает расстояние между словами, `line-height` — межстрочный интервал, а `margin-top` и `margin-bottom` — отступы перед абзацем и после него соответственно.

В этой главе нам осталось рассмотреть только возможности цветового оформления текста. Цвет может быть задан для фона текста и для самого шрифта. Цвет фона задается при помощи кнопки **Highlight Color** с изображением маркера. Применение этой возможности изменяет фон вводимого или заранее выделенного текста, и возникает ощущение, что по нему действительно провели цветным маркером. Если мы для нашей строки зададим желтый фон, то HTML-код для ее реализации будет выглядеть как

```
<p><span style="background-color: #FFFF00">Это моя первая Web-страница!</span></p>
```

Как видно, цвет задается в теге `<span>` и определяется при помощи сочетания шести шестнадцатеричных цифр. Это сочетание задает RGB-код цвета. Первые две цифры показывают насыщенность красного цвета, вторые две — насыщенность зеленого, и, наконец, последние две — насыщенность синего.

Для того чтобы определить цвет самого шрифта, используется кнопка **Font Color**. Если мы назначим для нашей строки светло-зеленый цвет (Lime), то наша строка будет реализована с помощью конструкции

```
<p><font color="#00FF00">Это моя первая Web-страница!</font></p>
```

В этом случае, применяется уже знакомый нам тег `<font>` с параметром `color`, а сам цвет все так же задается шестизначным кодом.

Очень часто отдельные разделы документа отделяются друг от друга при помощи горизонтальных линий. Для вставки горизонтальной линии применяется команда меню **Insert/Horizontal Line**. Линия будет вставлена в то место, где расположен текстовый курсор. Применяемый для этого HTML-тег выглядит как `<hr>`. Вставленная линия является объектом, следовательно, у нее есть свои свойства, которые можно устанавливать и редактировать. Для этого необходимо выделить вставленную горизонтальную линию и выполнить команду **Format/Properties** или команду **Horizontal Line Properties** контекстного меню, вызываемого щелчком правой кнопки мыши. При этом активизируется одноименное диалоговое окно для задания свойств горизонтальной линии. В поле **Width** мы можем указать ширину линии, а затем одним из переключателей указать единицу измерения. Длина линии может указываться как в пикселах, так и в процентах ширины окна просмотра браузера удаленного пользователя. В последнем случае, при изменении пользователем размеров окна просмотра, ширина горизонтальной линии также будет пересчитана. В поле **Height** указывается толщина линии в пикселах. По умолчанию она составляет два пиксела. С помощью группы радиокнопок **Alignment** мы можем указать, как будет выравниваться линия в окне просмотра. Переключатель **Left** прижимает ее к левому краю, **Center** позволяет отцентрировать ее, а радиокнопка **Right** прижимает нашу линию к правому краю окна просмотра. Если установлен переключатель **Solid line (no shading)**, то линия отображается плоской, без трехмерного выделения тенями. Для указания цвета линии используется список цветов **Color**. После установки значений свойств линии, отличных от вариантов по умолчанию, мы увидим, что тег, объявляющий линию, изменился. Он приобрел вид:

```
<hr size="3" width="80%" color="#00FFFF" align="left" noshade>
```

В параметре `size` указывается толщина линии в пикселах, параметр `width` задает длину линии, `color`, как нетрудно догадаться, указывает ее цвет, `align` — выравнивание, а наличие параметра `noshade` убирает трехмерные тени.

Ссылки и графика

Что главное в гипертексте? Главное в нем то, что он — "гипер" и позволяет реализовывать ссылки на другие документы, которые могут помочь при работе с основным текстом. То есть, то, что называется гиперссылками. Это сердцевина и основа, альфа и омега World Wide Web. Гиперссылки разделяются на два типа — внешние и внутренние. Внешние ссылки позволяют удаленному пользователю переходить к другим HTML-документам, а внутренние служат для быстрого передвижения внутри одного документа. Простейшим примером организации внутренних ссылок является оглавление содержимого Web-страницы, в котором каждая строка является ссылкой на начало новой части документа.

Чаще всего ссылки создаются в виде текста, указывающего адрес, куда произойдет переход. Для того чтобы их было можно отличить от основного текста,

браузер выделяет их цветом. Причем цветовое оформление обычной ссылки и ссылки, к которой пользователь уже обращался, как правило, различается.

Бывает, что ссылки оформляются в виде графического изображения. То есть, для активизации ссылки и начала процесса перехода, необходимо щелкнуть мышкой на рисунке. Именно поэтому вопросы создания графики и ссылок мы поместили в одной главе. Сначала мы рассмотрим вопросы создания гиперссылок, а потом перейдем к использованию графических изображений при оформлении Web-страниц.

Итак, приступаем к созданию гиперссылок. Для вставки гиперссылки в текст Web-страницы необходимо выполнить команду меню **Insert/Hyperlink** или нажать кнопку **Hyperlink** с изображением земного шара и звена цепи, которая находится на основной инструментальной панели. При этом активируется диалоговое окно **Create Hyperlink** (рис. 1.6).

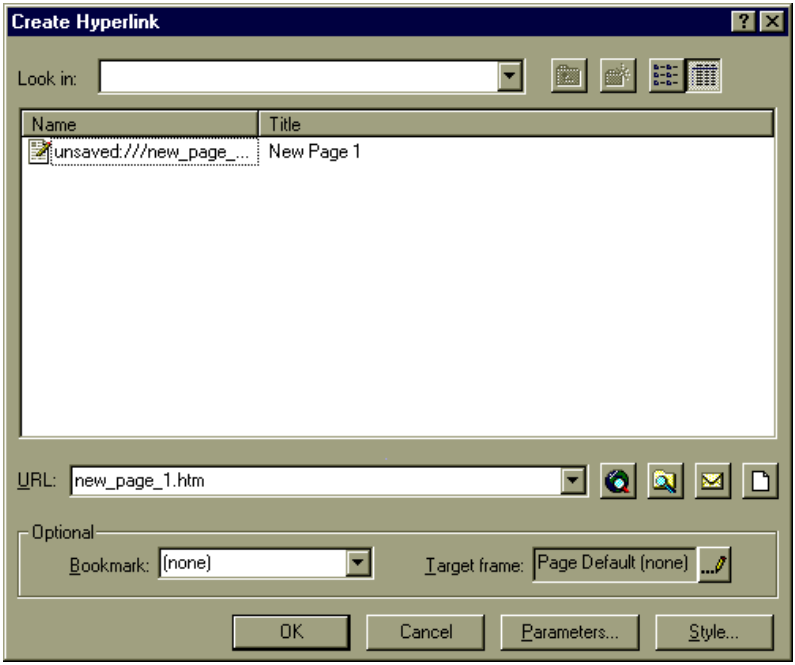

Рис. 1.6. Диалоговое окно **Create Hyperlink**

В том случае, если при создании гиперссылки в макете Web-страницы не было выделенного текста, гиперссылка будет вставлена в виде обычного URL. Если же был выделен текст, то ссылка связывается с этим выделенным текстом и URL не отображается на экране.

Для того чтобы указать URL гиперссылки, то есть адрес точки перехода, его надо написать в поле **URL**. В том случае, если ссылка должна указывать на документ, входящий в состав сайта, его можно выбрать из основного списка

диалогового окна. Для поиска адреса можно воспользоваться дополнительными кнопками, расположенными правее строки ввода URL. Кнопка с изображением земного шара и лупы запускает браузер, установленный в системе по умолчанию. При этом проектировщик страницы может найти необходимый документ обычными средствами Интернета. Обычно при закрытии браузера URL последней просмотренной страницы автоматически подставляется в поле ввода URL. В том случае, если необходимый ресурс находится на локальном компьютере, следует нажать следующую кнопку с изображением папки и лупы, которая активизирует стандартное диалоговое окно для открытия файла. Третья кнопка с изображением конверта предназначена для создания гиперссылки, которая позволяет отсылать электронное письмо. При нажатии на эту кнопку появляется дополнительное диалоговое окно, в единственное поле ввода которого необходимо ввести адрес, по которому письмо будет отправлено. Как мы уже знаем, World Wide Web по большей части основан на протоколе HTTP, а электронная почта реализуется при помощи протоколов POP3 и SMTP. Следовательно, Web-страницы и электронная почта являются разными вещами. Но в том-то и прелесть нынешнего Интернета, что все различные протоколы интегрированы в единое целое. Но как я и обещал, нам не надо вдаваться в тонкости. Мы можем просто творить. Браузеры в чистом виде не приспособлены к передаче электронной почты, поэтому, как только пользователь щелкает мышью на ссылке с адресом электронной почты, запускается та почтовая программа, которая установлена в его системе. Текст письма он набирает самостоятельно, а адрес подставляется тот, который указан в ссылке.

И, наконец, последняя кнопка с изображением чистого листа. Она используется в том случае, когда страницы, на которую устанавливается ссылка, еще нет. Нажатие этой кнопки добавляет к проектируемому сайту чистую страницу, а в текущей странице размещается ссылка на нее.

Выпадающий список **Bookmark** из группы элементов управления **Optional** предназначен для создания внутренних ссылок, которые в FrontPage 2000 реализуются при помощи закладок. Каждая закладка вставляется в необходимом месте текста при помощи команды **Insert/Bookmark**. Так же, как и в Microsoft Word каждая закладка может иметь свое имя. Но нам это не важно. В HTML-документах каждая закладка является маркером, к которому можно привязать гиперссылку. Таким образом, мы видим, что перед тем, как создавать внутренние ссылки, необходимо создать набор закладок в тексте, на которые мы потом получим возможность ссылаться.

И последний рассматриваемый орган управления — поле **Target frame**. Это поле не предназначено для прямого ввода. Чтобы установить в него значение, используется привязанная к этому полю кнопка. Поле **Target frame** задает имя окна, в которое будет загружаться документ. По умолчанию используется параметр **Page Default**, который отрабатывает стандартный вариант отображения последовательности страниц браузером. Параметр **Same Frame** указывает на то, что документ должен быть загружен в то же самое

окно, где расположена ссылка. Параметр **Whole Page** указывает на то, что загружаемый документ займет все окно браузера. Параметр **New Window**, как нетрудно догадаться, принудительно открывает для документа новое окно. И, наконец, параметр **Parent Frame** указывает на то, что документ будет загружен в окно, которое является родительским, по отношению к текущему.

Теперь давайте посмотрим, как механизм ссылок реализуется в HTML. Для этого на пустой странице разместим две строки. Одна будет являться закладкой, а другая будет представлять собой ссылку. Сначала попробуем создать локальную гиперссылку. Для этого, как уже говорилось ранее, мы используем список закладок **Bookmark** диалогового окна **Create Hyperlink**. В том случае, если установка закладки на первую строку прошла гладко, мы увидим ее имя в списке закладок. Достаточно будет щелкнуть мышью на названии закладки и в строке **URL** появится ее адрес. Нажмем кнопку **OK**, и перейдем на страницу **HTML**, чтобы посмотреть HTML-код нашего творения. Он будет выглядеть приблизительно следующим образом:

```
<p><a name="Место для закладки">Место для закладки</a></p>
<p><a href="#Место для закладки">Ссылка на закладку</a></p>
```

Итак, как мы видим, помимо тегов объявления абзаца, в каждой строке появился тег `<a>` с различными параметрами. Именно он применяется для реализации гиперссылок. В первой строке мы объявили закладку с именем "Место для закладки". Именно поэтому в теге `<a>` появился параметр `name`, который создает маркер для внутренних ссылок. После текста закладки выставлен закрывающий тег `</a>`. Во второй строке установлена сама ссылка. В этом случае используется параметр `href`, в качестве значения которого указывается URL необходимого документа. Так как ссылка в данном случае внутренняя, то перед самим URL ставится знак решетки, а сам URL будет просто именем маркера. Причем цветовое оформление гиперссылки будет применено ко всему тексту, который находится между тегом `<a>` и его закрывающей парой `</a>`.

Теперь попробуем создать внешнюю гиперссылку. Для этого выделим вторую строку и нажмем кнопку **Hyperlink** или выберем в контекстном меню, вызываемом правой кнопкой мыши, команду **Hyperlink Properties**. Результатом этих действий будет активизация диалогового окна **Edit Hyperlink**, которое является близнецом уже знакомого нам диалогового окна **Create Hyperlink**. Введем в поле **URL**, скажем, адрес наиболее известного книжного магазина Сети — http://www.amazon.com. А также укажем окно для загрузки основной страницы этого сайта. Выберем в списке **Target Frame** значение **Whole Page**, т. е. эта страница займет все окно просмотра браузера. После нажатия на кнопку **OK** наша гиперссылка будет изменена. Теперь при просмотре HTML-кода мы увидим следующее:

```
<a href="http://www.amazon.com" target="_top">Ссылка на закладку </a>
```

Как мы видим, в параметре `href` теперь указан обычный адрес с указанием протокола. Параметр `target` содержит значение, которое указывает тип окна для загружаемого документа.

Вот так создаются обычные гиперссылки. Перейдем теперь к использованию графики в оформлении Web-страниц.

FrontPage 2000 позволяет использовать три вида графики: стандартные графические файлы, картинки из коллекции ClipArt и видеофрагменты. Да-да, видеофрагмент тоже считается картинкой, и команда его вставки находится в одной группе с другими командами использования графики. Заниматься видеофрагментами мы будем позже, а сейчас сосредоточимся на обычных изображениях.

Вставка графического файла осуществляется при помощи команды меню **Insert/Picture/From File** или соответствующей кнопки на основной инструментальной панели. При этом активизируется диалоговое окно **Picture** (рис. 1.7).

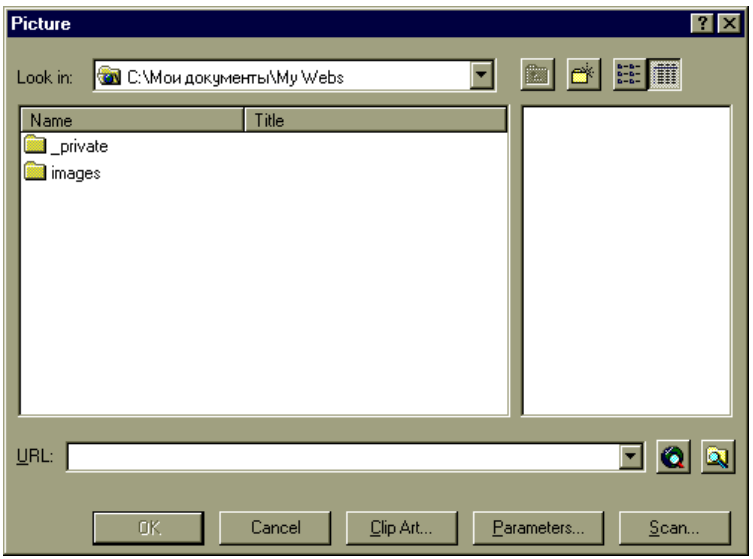

Рис. 1.7. Диалоговое окно **Picture**

Как и при создании гиперссылки, диалоговое окно предназначено для указания источника изображения. Для этого используется не имя файла, а его URL. Конечно, было бы неплохо, если бы файл с изображением находился на той же машине, где установлен Web-сайт, но никаких ограничений относительно его местоположения не существует.

Итак, по умолчанию считается, что файл с графическим изображением находится в структуре каталогов создаваемого сайта, которая показана в основном списке диалогового окна **Picture**. Однако, местонахождение файла будет записано в поле ввода **URL**. Справа от этого поля находятся уже известные нам две кнопки, которые позволяют отыскивать требуемое изображение в необъятных просторах Интернета и в глубинах дискового пространства локальной машины. Помимо этого есть и другие кнопки, которые рас-