Дунаев В. В.

HTML, скрипты и стили

2-е издание

Санкт-Петербург

«БХВ-Петербург»

2008

УДК 681.3.06
ББК 32.973.26-018.2
Д83

Дунаев В. В.

Д83 HTML, скрипты и стили. 2-е изд. доп. и перераб. — СПб.: БХВ-Петербург, 2008. — 1024 с.: ил.

ISBN 978-5-9775-0111-8

Рассмотрены основные сведения по разработке Web-приложений. Подробно описаны возможности языка разметки гипертекстовых документов (HTML) и каскадных таблиц стилей (CSS). В отличие от первого издания большое внимание уделено программированию клиентских и серверных сценариев на языках JavaScript, VBScript и PHP. Приведено большое количество примеров всевозможных сценариев от простых визуальных эффектов до использования элементов ActiveX и разработки баз данных. Показано решение типовых задач: учет количества посещений страницы, передача данных на сервер из формы, работа с базами данных и др. Описана работа с файловой системой и реестром Windows. Книга сопровождается большим количеством примеров.

Для Web-дизайнеров

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Татьяна Лапина*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 22.11.07.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 82,56.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

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

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

ISBN 978-5-9775-0111-8

Оглавление

Предисловие ко второму изданию

Во 2-м издании данной книги исправлены замеченные опечатки, неточности и ошибки, а также добавлен новый материал, в частности, по PHP. Таким образом, теперь серверным сценариям уделено существенно больше внимания, чем в предыдущем издании.

Во избежание возможных недоразумений считаю необходимым предупредить, что рассматриваемые в книге клиентские сценарии корректно работают в браузере Microsoft Internet Explorer 5+ под Windows. Некоторые из них могут неправильно выполняться или совсем не выполняться в других браузерах (например, Opera и Mozilla Firefox). Это обусловлено в значительной степени особенностями объектной модели документа и браузера, которую поддерживает тот или иной браузер. Все современные браузеры поддерживают модель, рекомендованную W3C, в той или иной степени, и Internet Explorer это делает не хуже, если не лучше, других, но кроме объектов этой стандартной модели он предоставляет разработчику и другие очень полезные объекты, недоступные браузерам-конкурентам. Следует отметить еще и отличия в моделях распространения событий по элементам документа. В конце концов, Internet Explorer поддерживает не "чистый" JavaScript, а его специфическую редакцию JScript. Однако практически любой сценарий на JavaScript, корректно выполняющийся в браузере-конкуренте, правильно сработает и Internet Explorer, но не всякий JScript-сценарий будет верно интерпретироваться другими браузерами. Для межбраузерной совместимости сценариев их следует делать возможно более простыми, а обработку событий организовывать так, чтобы она не зависела от модели их распространения. Разумеется, существуют и другие, более сложные способы обеспечения инвариантности Web-страниц к различным браузерам, но они не рассматриваются в предлагаемой книге.

Я старался раскрыть возможности HTML, CSS и скриптовых языков для разработки Web-приложений, не претендуя на полноту изложения. Это не справочник, а скорее расширенное практическое руководство по самостоятельному изучению разнообразных средств и приемов Web-программирования, доступное даже начинающим.

Введение

Чтобы опубликовать какой-либо документ в Word Wide Web (WWW, Всемирная паутина), достаточно расположить его на Web-сервере. Такие документы называются еще *Web-страницами*, а совокупность из нескольких страниц, взаимосвязанных гиперссылками, общей темой или дизайном — *Web-узлом* или *сайтом*. Другими словами, Web-сайт — это множество информационных ресурсов, доступных посредством службы WWW Интернета. Сайты просматриваются с помощью браузеров (обозревателей), таких как Microsoft Internet Explorer, Netscape Navigator и др. Если вы создаете свой сайт, то все начинается с разработки главной (начальной, домашней) страницы, содержащей ссылки на другие документы. Как главная, так и иерархически подчиненные страницы (документы), в Web представляются файлами. Все в компьютерном мире сохраняется в файлах и отображается теми или другими программами (вьюверами, браузерами, обозревателями). Файлы, представляющие Web-страницы, являются обычными текстовыми файлами, которые кроме простых текстов могут содержать еще и специальные команды, влияющие на отображение содержимого в окне браузера.

Создать документ для публикации в Web можно в обычном текстовом редакторе, таком как Блокнот Windows. Можно также подготовить его в мощном текстовом процессоре типа Microsoft Word, а затем сохранить как Web-страницу, хотя для создания отдельных страниц и целых сайтов служат специальные системы визуальной разработки, такие как Macromedia Dreamweaver, Microsoft FrontPage и Macromedia Flash. Каким бы средством вы ни воспользовались, в результате получите файлы, содержащие, кроме всего прочего, специальные коды разметки или форматирования документа. Эти коды, называемые *тегами*, — инструкции (команды) языка HTML (HyperText Markup Language, язык разметки гипертекста). Они представляют собой некоторые предопределенные ключевые слова, заключенные в уголковые скобки < и >. Теги не отображаются на экране или при печати, а лишь управляют отображением. Это своего рода управляющие символы, предназначенные

для указания программе просмотра, как и где следует показывать содержимое, заключенное между ними.

HTML задумывался как язык разметки документа, содержащего текстовую и графическую информацию вместе с элементами управления, такими как ссылки на другие документы. Тексты, имеющие элементы, которые являются ссылками на другие документы, называются *гипертекстами*. В качестве гиперссылок могут выступать фрагменты текста, графические изображения и другие элементы. Кроме текстовой и графической информации, а также ссылок, в документ можно включать аудио- и видеофрагменты. Чтобы создать такой документ, достаточно в обычном текстовом редакторе (например, в Блокноте Windows) написать программу на языке HTML. Такие программы сохраняются в файлах с расширением htm или html, а выполняются в Web-браузере.

Из истории

К середине 90-х годов прошлого века язык HTML приобрел широкую популярность. Различные компании-разработчики программного обеспечения для работы в Интернете стали предлагать свои варианты тегов HTML и способы их интерпретации браузерами. Поэтому назрела необходимость стандартизации языка. Созданием спецификации HTML занялся Word Wide Web Consortium (W3C, Консорциум всемирной паутины) — организация, в состав которой вошли крупнейшие производители программного обеспечения для Интернета, такие как Microsoft, Sun Microsystems, Netscape и др. В конце 1995 г. вышла в свет спецификация HTML 2.0, оформившая практику использования HTML, которая сложилась к концу 1994 г.

Схема подготовки спецификаций консорциумом такая: сначала выпускается проект спецификации, в результате обсуждения которого появляется ее рабочий вариант. Рабочий вариант предлагается к обсуждению в течение некоторого времени, по окончании которого рабочий вариант может стать рекомендацией — официальным вариантом спецификации HTML.

Вскоре после выхода спецификации HTML 2.0 появился рабочий вариант HTML 3.0 со сроком окончания обсуждения в сентябре 1995 г. Эта спецификация так и не стала официальной рекомендацией.

В мае 1996 г. был выпущен проект HTML 3.2, а уже в сентябре того же года он стал рабочей обсуждаемой версией. В январе 1997 г. спецификация HTML 3.2 была принята в качестве официальной рекомендации. Когда говорят о простом или классическом HTML, имеют в виду именно спецификацию версии 3.2.

Однако вскоре выяснилось, что HTML 3.2 не вполне отвечает потребностям разработчиков и пользователей Web-сайтов. Во-первых, нужны были более мощные и гибкие средства форматирования. Во-вторых, требовалась возможность вставки в HTML-документ объектов сторонних производителей. В-третьих, требовались средства для управления содержимым HTML-документа, загруженного в браузер, а также для обработки действий пользователя (манипуляции мышью, нажатия клавиш). Иначе говоря, HTML-документ должен быть динамическим и интерактивным (т. е. взаимодействовать с пользователем).

В июле 1997 г. консорциум W3C выпустил проект спецификации HTML 4.0, который уже в декабре стал официальной рекомендацией, которую еще называют *динамическим HTML* (DHTML). На момент выхода в свет данной книги последней официальной рекомендацией является HTML 4.01.

В ответ на потребность в мощных средствах форматирования в DHTML появились так называемые CSS (Cascading Style Sheets, каскадные таблицы стилей). Теперь параметры (атрибуты) тегов стало возможным задавать не только с помощью атрибутов, но и с помощью специальных средств — *правил форматирования*. Главная идея применения CSS — отделение описания структуры документа от его визуального представления.

Объекты сторонних производителей, созданных с помощью различных языков программирования, вставляются в HTML-документ посредством специальных тегов. Используя их, можно встроить в HTML-документ звук, видео, Flash-анимацию, векторную графику, таблицы базы данных и многое другое.

Для управления элементами HTML-документов и даже самим браузером, генерации новых документов, организации диалогового взаимодействия с пользователем, выполнения каких-то расчетов и обработки данных в HTML была предусмотрена интеграция со специальными языками программирования. Программы, написанные на этих языках, называют *сценариями* (scripts). Стандартным языком сценариев является JavaScript. Его должны уметь интерпретировать все Web-браузеры. Браузер Microsoft Internet Explorer помимо JavaScript воспринимает еще один язык — VBScript (Visual BasicScript), чего нельзя сказать о других браузерах.

Каким образом сценарии взаимодействует с HTML-документом и браузером? Дело в том, что браузер и загруженный в него HTML-документ представлены внутри браузера посредством иерархического множества объектов — так называемой *объектной модели документа* (Document Object Model, DOM). Для сценария объекты браузера и HTML-документа образуют доступную среду. Изменяя свойства этих объектов с помощью сценария, можно модифицировать содержимое и внешний вид документа без перезагрузки его в браузер.

Таким образом, динамический HTML покоится на трех "китах": классическом HTML, CSS и скриптах, а сами "киты плавают в море" объектной модели.

Как устроена эта книга

Данная книга содержит, я бы сказал, "джентльменский" набор сведений, необходимый для разработки Web-приложений: далеко не все, но все, что нужно в первую и даже во вторую очередь. Книга вполне доступна для начинающих, поскольку не требует никаких предварительных знаний и навыков в области программирования. Вместе с тем, и более опытные читатели, я надеюсь, найдут в ней немало интересного и полезного.

Книга состоит из двух частей. *Часть I (гл. 1—10)* посвящена классическому HTML и CSS, т. е. всему тому, что можно применять без скриптов. В *части II (гл. 11—19)* рассматривается программирование сценариев, как клиентских, так и серверных. Основное внимание здесь уделено языку JavaScript, но также приведены краткие сведения и по VBScript. Язык JavaScript (для Internet Explorer — Jscript) является основным для создания клиентских сценариев и довольно широко используется в серверных сценариях. VBScript понимается только браузером Internet Explorer. Однако этот язык является основным при создании серверных сценариев на Active Server Pages (ASP), поддерживаемых Web-сервером Internet Information Services (IIS) на платформе Windows. Серверные сценарии на JavaScript и VBScript (в рамках технологии ASP), а также язык технологии PHP рассмотрены в книге на примерах типичных задач, с которыми обычно сталкиваются начинающие.

Излагать и изучать DHTML можно по-разному, в зависимости от опыта, целей и личных особенностей восприятия. Можно подойти к нему со стороны обычного HTML, рассматривая сценарии как дополнительные средства обеспечения требуемой функциональности создаваемым документам. А можно, наоборот, исходить из сценариев, рассматривая HTML и CSS просто как средства вывода информации на экран или печать.

Мне больше импонирует второй подход. Однако изложение материала в этой книге построено приблизительно в соответствии с логикой его исторического развития, которая лучше воспринимается, как мне кажется, новичками. При этом уже с первых шагов изучения вы сможете получать осязаемые результаты, а по мере продвижения по главам — заметно наращивать свое могущество. В связи с этим следует иметь в виду, что нередко одна и та же тема рассматривается в различных главах. Например, фреймам с точки зрения HTML посвящена *гл. 6*, а с точки зрения сценариев — *разд. 12.7.2*. В табл. В1 приведен указатель для некоторых наиболее важных тем.

Таблица В1. *Темы, рассматриваемые в книге*

Тема	Где описано
Тексты	*Гл. 2*, *разд. 10.5*, *10.8*, *14.1.5*, *14.1.7*
Гиперссылки	*Гл. 4*, *разд. 4.2*, *4.3*
Графика и визуальные эффекты	*Гл. 3*, *разд. 4.2*, *4.3*, *10.11*, *12.10*, *14.1—14.3*
Таблицы	*Гл. 5*, *разд. 14.8*
Фреймы	*Гл. 6*, *разд. 12.6.1*, *12.7.2*, *12.7.3*
Меню и списки	*Разд. 2.6*, *7.3*, *14.6*
Формы	*Гл. 7*, *14.5*, *17.2.2*, *17.2.3*, *19.1.1*, *19.2.2*, *19.5.4*
Элементы управления ActiveX	*Гл. 15*, *разд. 9.2*, *14.6.3*, *14.6.4*, *14.8*, *17.2.1*
Flash	*Разд. 9.2.1*, *3.10.7*, *14.9*
Передача данных на сервер	*Разд. 7.6*, *14.5*, *16.3.2*, *17.2.3*, *19.1*, *19.2*
Объекты	*Разд. 11.9*, *11.10*, *12.6*, *15.1*, *18.12*
Файловая система	*Гл. 15*, *разд. 17.2.1*, *19.4*
Базы данных	*Разд. 17.2.3*, *19.5*

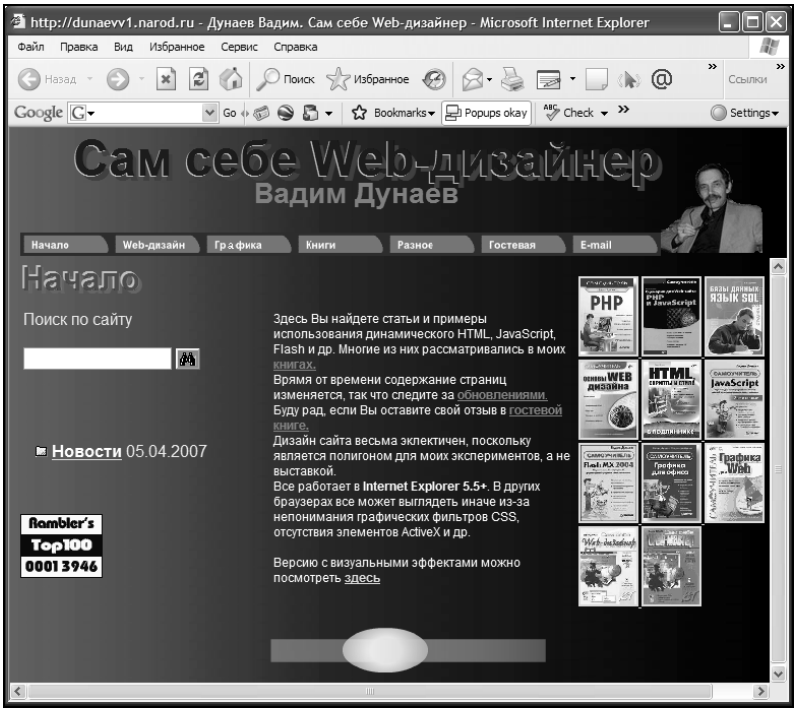

Рис. В1. Сайт автора **http://dunaevv1.narod.ru**

Все затронутые в книге темы иллюстрируются большим количеством примеров. Многие из них воспроизведены на моем сайте "Сам себе Web-дизайнер" по адресу: **http://dunaevv1.narod.ru**. Этот, весьма эклектичный по своему дизайну (рис. В1), сайт содержит статьи и примеры, предназначенные главным образом для новичков. Они корректно воспроизводятся в браузере Internet Explorer, начиная с версии 5.5. Мой сайт иллюстрирует разнообразные возможности средств разработки, а его дизайн не следует рассматривать в качестве примера для подражания. Дизайн — интересная и важная тема, но не ей посвящена данная книга. Я описываю лишь средства, а какие из них применить — дело вкуса разработчика сайта. Буду рад, если вы оставите свой отзыв в моей гостевой книге. Многие темы и решения, изложенные здесь, были предложены или навеяны читательскими отзывами на мои предыдущие книги.

Обзор содержания книги

Если вы уже имеете некоторый опыт обращения с HTML, CSS и скриптами, то последующий текст, возможно, вам поможет настроиться на данную книгу. В противном случае вы ничего не потеряете, если пропустите его.

Итак, мы начинаем с изучения обычного или, как еще говорят, классического HTML *(гл. 1—9)*. Выбранная точка старта обусловлена отнюдь не моими симпатиями, а простым желанием предоставить новичку в этом деле возможность быстрого входа в тему.

Как уже отмечалось, документ, который вы хотите опубликовать в Интернете, следует сохранить в обычном текстовом файле с расширением htm или html. Допустимы и другие расширения (например, asp, shtml), но только htm или html обычно ассоциируются с Web-браузером, который должен их воспроизводить, т. е. показывать на экране компьютера.

HTML-документ, сохраняемый в html-файле, может содержать разнообразную информацию, которую удобно разделить на два типа: содержательную и управляющую. К *содержательной информации* обычно относят то, что должен увидеть пользователь на экране своего компьютера, например, текст и графические изображения. К *управляющей (форматирующей) информации* относится все, что определяет внешний вид и расположение содержательной информации. HTML предоставляет средства для определения как содержательной, так и управляющей информации. Например, если требуется отобразить некоторый текст (содержательная информация), достаточно просто ввести этот текст в документ. Браузер покажет его в виде, соответствующем своим настройкам. Однако вы можете явно указать, как именно следует отобразить этот текст. Для этого служат специальные команды (теги) языка

HTML. Например, если вы введете перед текстом тег `<b>`, а после него `</b>`, то этот текст будет отображен жирным шрифтом. Аналогичным образом можно указать и другие параметры шрифта. Для текста в HTML существует множество тегов, с помощью которых можно задать его форматирование — внешний вид и расположение относительно других элементов документа. В этом плане идея HTML чрезвычайно проста: с помощью тегов следует выделить форматируемый текст и указать, как именно его нужно отобразить.

Кроме текста в документ часто приходится вставлять графические изображения. Для этой цели был придуман специальный тег `<img>` с параметром, указывающим, из какого графического файла следует взять изображение. Например, для вставки в документ изображения из файла mypicture.jpg следует использовать такую запись: `<img src="mypicture.jpg">`. Таким образом, появился тег, который вставляет в документ элемент из внешнего файла. Это нечто иное, чем просто форматирование текста, расположенного в том же файле.

Далее, некоторые слова или целые фразы, а также графические изображения можно наделить функциональностью *гиперссылок*. Для этого достаточно заключить их в специальные теги `<a>` и `</a>`, подобно тому, как блоки текста окаймляются тегами форматирования. Инициализация гиперссылки щелчком на ней кнопкой мыши приводит к загрузке в браузер документа (файла), указанного в параметре этой гиперссылки. Тексты, графика и гиперссылки — набор элементов, наиболее часто используемых при создании HTML-документов.

Кроме графики в HTML-документ можно вставлять и другое внешнее содержимое. Под *внешним содержимым* понимается то, что не размещено непосредственно в html-файле, а находится в других файлах. Так, в HTML-документ можно вставить, кроме графики, звуковое сопровождение, видео и специальные элементы, обеспечивающие, например, интерфейс пользователя (кнопки, поля ввода данных и т. п.), связь с базами данных и др. Эти элементы вставляются в HTML-документ с помощью различных тегов, например, `<input>`, `<button>`, `<object>` и `<embed>`. Таким образом, HTML обеспечивает создание разнообразных мультимедийных документов.

При создании простых HTML-документов смесь содержательной и управляющей информации, представляемой тегами и их параметрами, не вызывает особых хлопот. Однако положение дел заметно усложняется в случаях многостраничных сайтов с часто обновляемым содержимым. Перенос сложившегося стиля сайта в новую страницу, используя лишь теги и параметры форматирования, связан с весьма ощутимыми неудобствами и затратами времени. Поэтому появилась возможность информацию, определяющую внешний вид

элементов, хранить отдельно в виде таблиц стилей *(см. гл. 10)*. Таблицы стилей, содержащие параметры форматирования, могут храниться как непосредственно в HTML-документе, так и в отдельных файлах. Таблицы стилей являются мощным технологическим средством, позволяющим легко разрабатывать и модифицировать дизайн отдельных документов и сайта в целом. Кроме того, они помогают произвольным образом позиционировать элементы документа, а также создавать разнообразные визуальные эффекты (например, постепенное появление или исчезновение элементов, подсветку, полупрозрачность и т. п.). В идеале вы можете использовать всего лишь несколько тегов для вставки в документ внешнего содержимого, элементов пользовательского интерфейса, сценариев и таблиц стилей, а внешний вид, размеры и расположение видимых элементов задавать с помощью параметров стилей, а не тегов и их атрибутов.

Элементы документа, заданные с помощью тегов HTML и таблиц стилей, представляются внутри браузера множеством объектов со своими свойствами и методами. Элементы браузера также имеют объектное представление. Все эти объекты доступны с помощью *сценариев* — программ, написанных не на HTML, а на специальных языках. Это означает, что в сценариях можно читать и, в большинстве случаев, изменять свойства объектов. Клиентские сценарии (т. е. выполняемые браузером пользователя) пишутся на языке JavaScript. Любой браузер, воспринимающий сценарии, понимает, в той или иной степени, этот язык. Браузер Microsoft Internet Explorer интерпретирует редакцию JavaScript, называемую JScript. Эта редакция содержит дополнительные средства, не входящие в официальную спецификацию JavaScript. Кроме того, Internet Explorer — единственный браузер, который может выполнять сценарии, написанные и на языке VBScript. Этот язык очень похож на Visual Basic for Applications (VBA), который применяется в приложениях пакета Microsoft Office. Если вы используете клиентские сценарии в документах, опубликованных в Интернете и для вас важна инвариантность относительно браузеров различных типов, то их следует писать на JavaScript. Для публикаций в локальной или интрасети при работе с браузером Internet Explorer можно использовать оба языка. Синтаксис этих языков прост, а сценарии в большинстве случаев содержат всего несколько строк кода. Поэтому они легко могут быть освоены даже новичками. Основные сведения о JavaScript и VBScript приведены в *гл. 11* и *13* соответственно.

Сценарии могут выполняться и на стороне сервера *(см. гл. 16 — 19)*. Например, гостевые книги, счетчики количества посещений сайта, электронные магазины и многие другие приложения основаны на серверных сценариях. Если вы используете технологию ASP (Active Server Pages, активные серверные страницы), поддерживаемую Web-сервером IIS (Internet Information Services,

информационные службы Интернета), то сценарии можно писать как на JScript, так и VBScript. Аналогичным образом создаются сценарии на языке PHP, который поддерживают почти все Web-серверы на различных платформах (Windows, UNIX/Linux, Mac).

На JavaScript и VBScript можно также писать сценарии, предназначенные не только для выполнения Web-браузером или серверным интерпретатором. Так, в Windows имеется встроенный интерпретатор сценариев, называемый Windows Scripting Host (WSH, сервер сценариев Windows). С помощью этого интерпретатора обычно выполняют сценарии, хранящиеся в текстовых файлах с расширениями js и vbs и написанные на JavaScript и VBScript соответственно. Рутинную работу с файлами, реестром Windows, запуск приложений и многое другое можно выполнять не вручную, а посредством сценариев *(см. гл. 15)*. Аналогично имеется интерпретатор PHP, предназначенный для выполнения сценариев вне связи с Web-сервером.

Клиентские сценарии вставляются в HTML-документ с помощью тега `<script>`. При этом код сценария может быть записан как непосредственно между тегами `<script>` и `</script>`, так и в отдельном текстовом файле. Обычно простейшие сценарии описывают действия, выполняемые браузером при наступлении тех или иных событий, таких как щелчок кнопкой мыши (`onclick`) или наведение ее указателя на некоторый элемент документа (`onmouseover`). Вместе с тем, с помощью клиентских сценариев можно изменить содержимое HTML-документа без его перезагрузки. Например, сменить графическое изображение, заменить кнопку картинкой, а текст — раскрывающимся списком. Это делается с помощью изменения значения атрибута `src` или свойства `innerHTML`. Более того, исходный html-файл может содержать лишь сценарий, формирующий все содержимое документа. Так, в сценарии можно сформировать строку, содержащую последовательность тегов, а затем применить метод `document.write(`*`строка`*`)`, передав ему эту строку в качестве параметра. Принципы создания и примеры клиентских сценариев рассмотрены в *гл. 12* и *14*.

Благодарности

Я благодарен многочисленным читателям, приславшим мне свои замечания и отзывы, которые я попытался учесть при подготовке данного издания.

Я также благодарю свою жену Валентину, проявившую понимание и заботу в нелегкий для меня период работы над этой книгой.

ЧАСТЬ I

HTML И СТИЛИ

Глава 1

Структура HTML-документа

HTML-документ хранится на диске компьютера в виде обычного текстового файла с расширением htm или html. Чтобы просмотреть или отредактировать содержимое этого файла, достаточно открыть его в текстовом редакторе, например, Блокноте Windows (исполняемый файл notepad.exe). С другой стороны, если вы открыли или создали новый текстовый файл и наполнили его содержимым некоторого HTML-документа, то сохраните этот файл под каким-нибудь именем с расширением htm или html. Такой HTML-файл может быть открыт в Web-браузере (например, Microsoft Internet Explorer) двойным щелчком на его имени в окне Проводника Windows. Разумеется, HTML-документ можно создать, открыть для просмотра и редактирования в специальных редакторах (системах визуальной разработки Web-сайтов), таких как Macromedia Dreamweaver или Microsoft FrontPage.

При открытии HTML-файла в Web-браузере последний интерпретирует его содержимое и отображает на дисплее в своей клиентской области все, что требуется и возможно отобразить. Интерпретация HTML-документа заключается в последовательном выполнении браузером тегов (команд, дескрипторов, инструкций), записанных в HTML-документе.

1.1. Основные понятия

HTML-документ (содержимое HTML-файла) имеет довольно простую структуру. В этом документе располагаются теги языка HTML, которые определяют, что и как отобразить в окне браузера, а также некоторую дополнительную информацию, не связанную с визуализацией какого бы то ни было содержимого, но используемую браузером и/или серверными поисковыми системами.

Теги представляют собой некоторые предопределенные ключевые слова — имена тегов, заключенные в угловые скобки < и >. HTML-документ начинается с тега <html> и заканчивается тегом </html>. Здесь html и /html — ключевые слова, обозначающие начало и конец HTML-документа соответственно. Между этими тегами могут находиться другие теги и/или обычный текст. Ниже приведен пример HTML-документа, содержащего только некоторый текст:

```
<html>
  Привет всем!
</html>
```

Если сохранить этот текст в файле с расширением htm или html и затем открыть его в Web-браузере, например, MS Internet Explorer, то в результате получится подобное показанному на рис. 1.1. Это самый примитивный HTML-документ, содержащий только текст без всякого форматирования. Для вставки в HTML-документ графики, звука, видео и других элементов, а также для задания их относительного расположения служат специальные теги и их атрибуты.

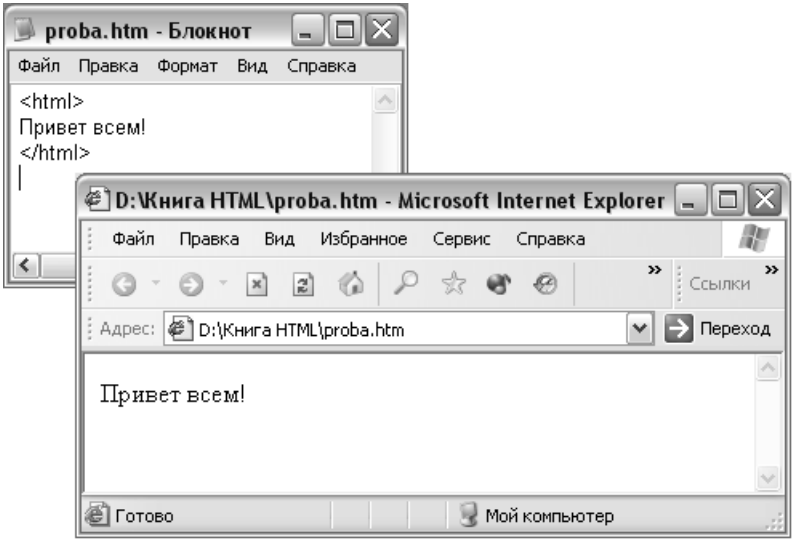

Рис. 1.1. Вид простого HTML-документа в окне текстового редактора и Web-браузера

Примечание

Перед открывающим тегом <html> нередко можно увидеть тег <!DOCTYPE>, с помощью которого объявляются тип и формат содержимого документа. Этот же тег вставляется и в XML-документы. Однако современные Web-браузеры

обрабатывают документы и без этого тега. Ниже приведен пример содержимого тега `<!DOCTYPE>`:

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

Здесь определен тип документа HTML, публичное DTD-описание которого принадлежит организации W3C, соответствует стандарту HTML 4.0 и ориентировано на англоязычные документы.

Если HTML-документ без тега `<!DOCTYPE>` открыть в браузере Internet Explorer 6.0, а затем выполнить команду **Файл | Сохранить как**, выбрав при этом тип файла html, то браузер добавит в сохраняемый файл тег `<!DOCTYPE>`. Более подробные сведения о теге `<!DOCTYPE>` приведены в *Приложении 1*.

Большинство тегов допускают использование *параметров* (атрибутов), задающих их некоторую спецификацию. Эти атрибуты могут задаваться просто именем или же (в большинстве случаев) парой *имя_атрибута=значение*. Значение атрибута называют еще *аргументом*. Имя атрибута является ключевым словом, а допустимое значение определяется спецификацией этого атрибута — описанием того, какие значения возможны и как их задавать. Например, для вставки в HTML-документ изображения из графического файла используется тег `<img>`. Чтобы указать файл графического изображения, следует использовать атрибут `src="`*адрес_графического_файла*`"`. Например, `<img src="picture.jpg">`. Здесь значением атрибута `src` является имя графического файла формата JPEG; в общем случае можно указать URL-адрес графического файла.

Внимание

Значения атрибутов принято задавать в виде строки, заключенной в двойные (") или одинарные (') кавычки. Однако современные браузеры правильно интерпретируют значения многих атрибутов и без кавычек. Вместе с тем во избежание недоразумений рекомендуется использовать кавычки. В данной книге эта рекомендация не везде выполняется.

Тег может иметь один или несколько атрибутов или не иметь их совсем. Некоторые теги имеют атрибуты, которые следует обязательно указывать, а другие атрибуты задаются лишь при необходимости. Например, в теге `<img>` атрибут `src` нужен, чтобы сообщить браузеру, где взять файл графического изображения. Атрибуты `width` и `height`, определяющие ширину и высоту изображения в окне браузера, не обязательны. Если их не указывать, то изображение будет занимать на экране прямоугольную область в соответствии со своими оригинальными размерами. Если в теге требуется указать несколько атрибутов, то их можно записывать в произвольном порядке. Например, следующие записи эквивалентны:

```
<img src="picture.jpg" width=100 height=200>
<img src="picture.jpg" height=200 width=100>
<img height=200 width=100 src="picture.jpg">
```

Ключевые слова, имена тегов и их атрибутов, могут быть записаны в любом регистре. Например, следующие записи эквивалентны:

```
<IMG SRC="picture.jpg">
<img src="picture.jpg">
```

Между открывающей угловой скобкой < и именем тега не должно быть пробелов. В противном случае вся запись будет воспринята и отображена браузером просто как текст. С другой стороны, между другими элементами в записи тега может использоваться любое количество ничего не значащих пробелов.

Теги, которые не могут быть проинтерпретированы браузером (в том числе с неверно записанными именами), просто игнорируются им.

Для обозначения комментария используется пара <! и >. Все, что находится между <! и ближайшей закрывающей угловой скобкой >, не отображается браузером. При интерпретации браузером следующего HTML-кода ничего не будет отображено:

```
<html>
  <! Это комментарий >
  <!img src="picture.jpg">
</html>
```

Тег комментария используется для сохранения в HTML-документе невидимых в браузере заметок разработчика, а также для временного отключения тегов без их удаления из текста документа, как это было сделано с тегом <img> в приведенном выше примере.

Содержимое HTML-документа может быть записано в одну или несколько строк. Имена тегов, атрибутов и значения атрибутов не следует разрывать для переноса на другую строку.[1] Ниже приведено несколько примеров записи одного и того же HTML-документа:

```
<html>
  <img src="picture.jpg" width=100 height=200>
</html>

<html>
  <img src="picture.jpg"
       width=100 height=200>
</html>

<html><img src="picture.jpg" width=100 height=200></html>
```

[1] Эта рекомендация тоже не всегда соблюдается в данной книге. — *Ред.*

При записи HTML-кода рекомендуется руководствоваться соображениями его удобочитаемости, которая чрезвычайно важна при редактировании особенно давно созданных HTML-документов. Поэтому первый из трех приведенных выше вариантов является наиболее предпочтительным.

Большинство тегов являются *контейнерными* или, иначе говоря, *парными*. Это означает, что тегу <*TEГ*> соответствует тег </*TEГ*>. Первый из них называют *открывающим*, а второй — *закрывающим*. Например, тег `<html>` является контейнерным: открывающему тегу `<html>` соответствует закрывающий тег `</html>`. Между открывающим и закрывающим одноименными тегами можно разместить другие теги. Иными словами, контейнерный тег может содержать внутри себя другие теги, в том числе и контейнерные. Контейнерные теги еще называют просто *контейнерами*. В следующем примере контейнерный тег `<html>` содержит тег изображения `<img>`:

```
<html>
  <img src="picture.jpg">
</html>
```

Некоторые теги не являются контейнерными, для них не предусмотрены закрывающие теги. Примером такого тега является `<img>`.

В HTML-документе могут присутствовать контейнерные теги, с помощью которых организуются важнейшие разделы документа:

- `<head>` — раздел заголовка;
- `<body>` — основной раздел (тело документа);
- `<style>` — содержит таблицу стилей;
- `<script>` — содержит сценарий (скрипт).

Перечисленные выше теги не являются обязательными и используются по мере необходимости.

Тег `<head>` располагается сразу за тегом `<html>` и может содержать другие теги, кроме `<body>`. Теги `<head>` и `<body>` указываются в документе не более одного раза и один из них не может содержать другой.

Теги `<style>` и `<script>` могут встречаться в HTML-документе произвольное количество раз, но не могут содержать другие теги. Внутри тега `<style>` записываются правила каскадных таблиц стилей (CSS), а тег `<script>` содержит код сценария (скрипта), написанного на языке JavaScript или VBScript. Довольно часто, но не всегда, их включают в раздел заголовка, определяемый тегом `<head>`.

1.2. Раздел заголовка <head>

Чтобы разместить в HTML-документе служебную информацию для браузера и поисковых систем, а также сценарии и таблицы стилей, которые должны быть загружены в браузер прежде основной содержательной части документа, служит контейнерный тег <head>. Обычно он содержит теги, определяющие:

- <title> — текст, отображаемый в заголовке окна браузера;
- <meta> — данные для использования серверами и поисковыми системами;
- <base> — базовый URL-адрес документа;
- <link> — связи между документами;
- <style> — таблицу стилей;
- <script> — код сценария.

Далее будут рассмотрены только первые четыре тега, а остальные — в других главах.

1.2.1. Тег <title>

Чтобы при загрузке в браузер отобразить в заголовке его окна некоторый текст (например, название документа), служит контейнерный тег <title>. Например:

```
<html>
  <head>
    <title> Мой сайт </title>
  </head>
  Основное содержание страницы
</html>
```

Способ отображения текста зависит от браузера. Настоятельно рекомендуется вставлять тег <title> в каждый HTML-документ. Содержащаяся в нем строка появляется в заголовке окна браузера еще до окончания загрузки документа. Кроме того, она отображается в качестве названия документа в списке закладок.

1.2.2. Тег <meta>

Информация, содержащаяся в тегах <meta>, не отображается браузером, однако имеет большое значение. В частности, она позволяет задать кодовую страницу языка просмотра документа, параметры его кэширования, ключевые слова, по которым ваш документ будут искать поисковые системы, и т. д. Эти теги обычно вставляются в тег <head> уже после разработки