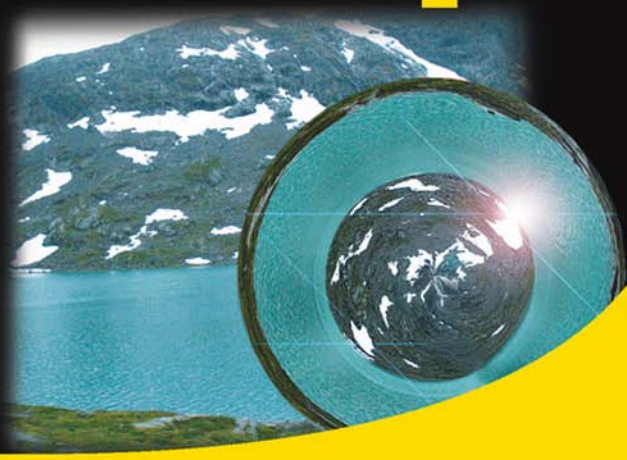
Самоучитель

Вадим Дунаев

Сценарии для Web-сайта

PHP и JavaScript

2-е издание



Основы языков PHP 5 и JavaScript

Разработка клиентских и серверных сценариев

Готовые примеры

Установка и настройка PHP и Web-сервера IIS

bhv

Вадим Дунаев

Сценарии для Web-сайта
PHP и JavaScript

2-е издание

Санкт-Петербург
«БХВ-Петербург»
2008

УДК 681.3.06
ББК 32.973.26-018.2
Д83

Дунаев В. В.

Д83 Сценарии для Web-сайта: PHP и JavaScript. — 2-е изд. перераб. и доп. — СПб.: БХВ-Петербург, 2008. — 576 с.: ил. — (Самоучитель)

ISBN 978-5-9775-0112-5

Книга посвящена использованию языков JavaScript и PHP для разработки Web-приложений. Приведены основные понятия, связанные с разработкой Web-сайта, а также сведения о языке HTML и каскадных таблицах стилей (CSS). Рассмотрены основы программирования на JavaScript и PHP 5. Приведены практические примеры различных клиентских и серверных сценариев. Описаны особенности и даны рекомендации по применению этих языков. В приложениях содержатся сведения по объектам документа и браузера, с которыми работают клиентские сценарии, а также рассказывается о том, как установить и настроить PHP и Web-сервер IIS в системе Windows. Во втором издании добавлены новые примеры.

Для начинающих Web-разработчиков

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 Смирновой*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 29.12.07.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 46,44.
Тираж 2000 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 978-5-9775-0112-5

Оглавление

Введение **1**

Глава 1. Как устроен Web-сайт? **3**

1.1. Основные понятия 3
1.2. Как устроен HTML-документ? 5
1.2.1. Начальные сведения 6
1.2.2. Структура документа 9
1.2.3. Ссылки 22
1.2.4. Вставка элементов из внешних источников 33
1.2.5. Разметка страницы 45
1.2.6. Форматирование текста 62
1.2.7. Формы и элементы пользовательского интерфейса 75
1.3. Каскадные таблицы стилей 91
1.3.1. Встраивание таблиц стилей в HTML-документ 92
1.3.2. Правила форматирования 94
1.3.3. Применение нескольких таблиц стилей 96
1.3.4. Единицы измерения 97
1.3.5. Шрифты 98
1.3.6. Цвет и фон 100
1.3.7. Размеры, поля, отступы и границы 102
1.3.8. Текст 105
1.3.9. Обтекание и видимость 107
1.3.10. Позиционирование 109
1.3.11. Графические фильтры 110
1.4. Что и как делают клиентские сценарии? 122
1.5. Что и как делают серверные сценарии? 126

Глава 2. Основы JavaScript **129**

2.1. Подготовка к программированию 129
2.2. Ввод и вывод данных 132
2.2.1. Метод *alert* 132
2.2.2. Метод *confirm* 133

2.2.3. Метод *prompt* 134
2.2.4. Метод *write* 134
2.3. Типы данных 135
2.4. Переменные и оператор присваивания 142
2.4.1. Имена переменных 142
2.4.2. Создание переменных 143
2.4.3. Область действия переменных 145
2.5. Операторы 145
2.5.1. Комментарии 146
2.5.2. Арифметические операторы 146
2.5.3. Дополнительные операторы присваивания 149
2.5.4. Операторы сравнения 150
2.5.5. Логические операторы 152
2.5.6. Операторы условного перехода 153
2.5.7. Операторы цикла 159
2.5.8. Выражения с операторами 164
2.6. Функции 166
2.6.1. Встроенные функции 167
2.6.2. Пользовательские функции 171
2.6.3. Выражения с функциями 177
2.7. Строки 178
2.7.1. Создание строкового объекта 178
2.7.2. Свойства объекта *String* 179
2.7.3. Методы объекта *String* обработки строк 180
2.7.4. Методы объекта *String* форматирования строк 185
2.7.5. Функции вставки и замены подстрок 187
2.7.6. Функции удаления ведущих и заключительных пробелов 188
2.8. Массивы 190
2.8.1. Создание массива 190
2.8.2. Многомерные массивы 192
2.8.3. Копирование массива 193
2.8.4. Свойства объекта *Array* 194
2.8.5. Методы объекта *Array* 195
2.8.6. Функции обработки числовых массивов 199
2.9. Числа 201
2.9.1. Создание объекта *Number* 204
2.9.2. Свойства объекта *Number* 204
2.9.3. Методы объекта *Number* 205
2.10. Математические вычисления 207
2.10.1. Свойства объекта *Math* 207
2.10.2. Методы объекта *Math* 207

2.11. Дата и время 209
2.11.1. Создание объекта *Date* 209
2.11.2. Методы объекта *Date* 211
2.11.3. Календарь 217
2.12. Объект *Boolean* 223
2.13. Объект *Function* 224
2.13.1. Создание объекта *Function* 224
2.13.2. Свойства объекта *Function* 225
2.13.3. Методы объекта *Function* 226
2.14. Пользовательские объекты 228
2.14.1. Создание объекта 228
2.14.2. Добавление свойств 231
2.14.3. Связанные объекты 231
2.14.4. Пример создания базы данных с помощью объектов 233
2.15. Специальные операторы 238
2.15.1. Побитовые операторы 238
2.15.2. Объектные операторы 239

ГЛАВА 3. КЛИЕНТСКИЕ СЦЕНАРИИ НА JAVASCRIPT 243

3.1. Объекты, управляемые сценариями 243
3.2. Обработка событий 251
3.2.1. Привязка обработчиков к элементам и событиям 252
3.2.2. Свойства события 257
3.2.3. Прохождение событий 263
3.3. Динамическое изменение элементов документа 266
3.3.1. Использование метода *write()* 267
3.3.2. Изменение значений атрибутов элементов 268
3.3.3. Изменение элементов 271
3.4. Работа с каскадными таблицами стилей 274
3.5. Окна и фреймы 280
3.5.1. Создание новых окон 281
3.5.2. Фреймы 285
3.5.3. Плавающие фреймы 293
3.5.4. Всплывающие окна 295
3.6. Загрузка изображений 299
3.7. Управление процессами во времени 302
3.8. Работа с cookie 304
3.8.1. Общие сведения 304
3.8.2. Функции чтения, записи данных и удаления cookie 307

3.9. Перемещение элементов мышью 311
3.9.1. Перемещение графических объектов 311
3.9.2. Перемещение текстовых областей 312
3.9.3. Перемещение плавающих фреймов 315
3.10. Меню 318
3.10.1. Раскрывающийся список 319
3.10.2. Одноуровневое меню 320
3.10.3. Простое двухуровневое меню 321
3.10.4. Сложное двухуровневое меню 323
3.10.5. Древовидный список 331
3.11. Таблицы 335
3.11.1. Доступ к элементам таблицы 335
3.11.2. Добавление и удаление строк таблицы 338
3.12. Простая база данных в текстовом файле 339
3.12.1. Элемент управления табличными данными 339
3.12.2. Перемещение по записям простых баз данных 345
3.12.3. Сортировка данных таблицы 348
3.12.4. Фильтрация данных таблицы 349
3.13. Поиск в тексте 352
3.14. Движение элементов по заданной траектории 354
3.15. Рисование линий 358
3.16. Обработка данных форм 361

Глава 4. Основы PHP 369

4.1. Подготовка к программированию 369
4.2. Вывод данных 372
4.3. Типы данных 374
4.4. Переменные и оператор присваивания 377
4.4.1. Имена переменных 377
4.4.2. Создание переменных 378
4.4.3. Отображение значений переменных 380
4.4.4. Переменные переменные 384
4.4.5. Область действия переменных 385
4.4.6. Проверка существования и типов переменных 386
4.5. Константы 387
4.6. Операторы 389
4.6.1. Комментарии 389
4.6.2. Арифметические операторы 389
4.6.3. Строковый оператор 391
4.6.4. Дополнительные операторы присваивания 391

4.6.5. Операторы сравнения 392
4.6.6. Логические операторы 393
4.6.7. Побитовые операторы 395
4.6.8. Операторы управления 396
4.6.9. Операторы цикла 399
4.7. Строки 399
4.7.1. Двойные и одинарные кавычки 399
4.7.2. Склейка строк 402
4.7.3. Преобразование строк 402
4.7.4. Форматирование строк 407
4.8. Числа 411
4.8.1. Математические функции 411
4.8.2. Математические константы 412
4.8.3. Представление чисел в различных системах счисления 413
4.8.4. Форматирование чисел 415
4.9. Дата и время 417
4.10. Массивы 420
4.10.1. Создание массива 420
4.10.2. Многомерные массивы 423
4.10.3. Отображение массивов 424
4.10.4. Операции над массивами 425
4.11. Глобальные предопределенные переменные 434
4.12. Функции 436
4.12.1. Пользовательские функции 436
4.12.2. Переменные функции 442
4.12.3. Встроенные функции 443
4.13. Объекты 443
4.13.1. Определение класса 444
4.13.2. Применение объектов 448
4.13.3. Ограничение доступа к свойствам и методам 449
4.13.4. Клонирование и удаление объектов 451
4.13.5. Использование методов несозданных объектов 452
4.13.6. Обработка исключений 452
4.13.7. Пример класса формы 453
4.14. Выполнение PHP-кода в текстовых строках 455

Глава 5. Серверные сценарии на PHP 457

5.1. Получение данных от клиента 457
5.1.1. Получение данных из HTML-форм 457
5.1.2. Загрузка файлов на сервер 466

5.2. Переходы и передача данных между Web-страницами 469
5.2.1. Вывод ссылок 470
5.2.2. Использование форм 470
5.2.3. Применение функции *header()* 470
5.2.4. Добавление информации к URL-адресу 472
5.2.5. Применение cookie 473
5.2.6. Применение сеансов PHP 475
5.3. Работа с текстовыми файлами 484
5.3.1. Открытие файла 484
5.3.2. Закрытие и удаление файлов 486
5.3.3. Чтение файла 487
5.3.4. Запись в файл 491
5.3.5. Простой счетчик посещений страницы 492
5.4. Работа с таблицами в текстовых файлах 493
5.4.1. Функции работы с табличными данными 494
5.4.2. Сложный счетчик посещений страницы 498
5.4.3. Баннер 504
5.4.4. Гостевая книга 508
5.4.5. Форум 514
5.5. Работа с базами данных 515
5.5.1. Общие сведения о базах данных 515
5.5.2. Основные средства PHP для взаимодействия с базой данных 517
5.5.3. Гостевая книга 520

ПРИЛОЖЕНИЯ 529

ПРИЛОЖЕНИЕ 1. ОСНОВНЫЕ ОБЪЕКТЫ БРАУЗЕРА И ДОКУМЕНТА 531

П1.1. Объект *window* 531
П1.1.1. Свойства объекта *window* 531
П1.1.2. Методы объекта *window* 533
П1.1.3. События объекта *window* 534
П1.2. Объект *document* 535
П1.2.1. Коллекции объекта *document* 536
П1.2.2. Методы объекта *document* 537
П1.2.3. События объекта *document* 537
П1.3. Объект *location* 538
П1.3.1. Свойства объекта *location* 538
П1.3.2. Методы объекта *location* 539
П1.4. Объект *history* 539
П1.4.1. Свойство объекта *history* 540
П1.4.2. Методы объекта *history* 540

П1.5. Объект *navigator* 540
П1.5.1. Свойства объекта *navigator* 540
П1.5.2. Коллекции объекта *navigator* 541
П1.5.3. Методы объекта *navigator* 541
П1.6. Объект *event* 541
П1.7. Объект *screen* 543
П1.8. Объект *TextRange* 543
П1.8.1. Свойства объекта *TextRange* 543
П1.8.2. Методы объекта *TextRange* 544

ПРИЛОЖЕНИЕ 2. УСТАНОВКА МОДУЛЯ PHP 546

П2.1. Где взять модуль PHP? 546
П2.2. Установка модуля PHP 547
П2.2.1. Размещение модуля PHP на диске компьютера 547
П2.2.2. Настройка модуля PHP 547
П2.2.3. Установка расширений PHP 548
П2.3. Настройка Web-сервера IIS 549

ПРИЛОЖЕНИЕ 3. УСТАНОВКА WEB-СЕРВЕРА IIS В WINDOWS 2000/XP 551

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ 553

Введение

Данная книга посвящена языковым средствам создания Web-узлов (сайтов) и рассчитана на широкий круг читателей, желающих глубже разобраться в "кухне" разработки приложений для Web и интрасетей.

Следуя по пути от простого к сложному, обычно начинают с изучения HTML (HyperText Markup Language, язык разметки гипертекста). Затем для обеспечения дополнительной функциональности Web-страниц в HTML-документы вставляют программы (сценарии, скрипты), написанные на других языках. Сценарии могут выполняться Web-браузером (клиентом) и/или Web-сервером. На языке JavaScript обычно пишутся клиентские сценарии, хотя на нем можно писать и серверные скрипты. На языке PHP создаются серверные сценарии. В настоящее время JavaScript и PHP — наиболее популярные языки для разработки Web-приложений. Кроме того, они могут использоваться и для создания программных проектов, работающих на локальном компьютере или в локальной сети.

Новички обычно, начав с освоения HTML, довольно быстро приходят к использованию на своих страницах хотя бы несложных скриптов, написанных на JavaScript и выполняемых браузером. Однако рано или поздно у них возникает желание создать что-то более сложное, например гостевую книгу, форум, систему учета посещений и посетителей, защиту паролем и т. д. Эти задачи можно решить с помощью серверных сценариев.

Синтаксис JavaScript и PHP довольно прост и вполне доступен даже тем, кто не считает себя программистом или не желает заниматься программированием профессионально. Многие задачи вы можете решить, не имея большого опыта в программировании. Здесь важно не столько знание синтаксиса языка, сколько умение алгоритмически мыслить и создавать модели (обобщенные схемы) объектов и процессов. Так, например, если прочитав *главу 5*, даже новичок поймет, что такие с виду разные проекты, как счетчик посещений страницы, баннер и гостевая книга, устроены аналогично, я буду доволен собой.

В предлагаемой книге вы найдете многое из того, что необходимо знать для разработки более или менее функционально сложных сайтов. Здесь много и справочной информации, и примеров решения конкретных задач. Вместе с тем изложение построено так, чтобы книга оказалась доступной для начинающих.

В *главе 1* рассмотрены основные понятия, связанные с разработкой Web-сайта, а также основные сведения о языке HTML и каскадных таблицах стилей (CSS). Интересующиеся только скриптами могут использовать эту главу в качестве справочного руководства.

Глава 2 посвящена основам языка JavaScript, а *глава 3* — способам программирования и примерам различных клиентских сценариев. Аналогично распределен материал по языку PHP 5: в *главе 4* изложены основы, а в *главе 5* — примеры решения задач.

В *приложениях 1—3* приведены сведения по объектам документа и браузера, с которыми работают клиентские сценарии, а также рассказывается о том, как установить и настроить PHP и Web-браузер IIS в системе Windows.

В силу ограниченности объема в книге рассмотрены далеко не все возможности JavaScript и PHP. Более того, не обсуждается специфика программирования для браузеров, отличных от Microsoft Internet Explorer. Задача книги состоит в том, чтобы ввести читателя в мир JavaScript и PHP и дать ему возможность получить практические результаты хотя бы в системе Windows.

Многие из рассмотренных в книге примеров можно увидеть в действии на моем сайте **http://dunaevv1.narod.ru**.

Глава 1

Как устроен Web-сайт?

В данной главе мы рассмотрим устройство Web-сайта с общей точки зрения, позволяющей увидеть, что и какими средствами делается клиентом и сервером — двумя сторонами, обеспечивающими функционирование сайта.

1.1. Основные понятия

Web-сайт или, другими словами, *Web-узел*, — это совокупность Web-ресурсов, связанных между собой общей темой содержания, дизайном, а также функционально. Ни одна из перечисленных выше составляющих не является обязательной. Сайт — это некий информационный проект, опубликованный в Web. Web-ресурсы, входящие в сайт, — это файлы различного формата и содержания: HTML-документы, простые или отформатированные тексты, графика, аудио, видео, таблицы стилей, скрипты, Flash-ролики и др. Ссылки на эти ресурсы вставляются в HTML-документы, которые составляют основу сайта.

Сайт может состоять из одной или нескольких страниц. Одностраничный сайт создается легко и просто, однако такие сайты в Интернете встречаются редко. Чаще требуется создать хотя бы две-три страницы, и именно в этом случае возникают первые более или менее серьезные вопросы о технологии. В окне браузера можно показать одну или сразу несколько страниц сайта. При необходимости посредством гиперссылок можно организовать переходы между страницами, экспонируемыми в браузере. Сколько страниц в сайте, как они показываются в окне браузера, каким образом осуществлять переходы между ними и как обрабатывать действия пользователя — все это относится к области *разработки функциональности сайта* (development).

Под *дизайном* (design) сайта обычно понимают его внешний вид, а также интерфейс, как средство доступа к функциональным возможностям сайта. Одна

и та же структура сайта, с одной и той же функциональностью, может быть воплощена (представлена внешне) по-разному. При разработке сайта важно помнить, что он создается для посетителей, а потому должен быть содержательным, удобным и, по возможности, красивым. Основной принцип, которым следует руководствоваться при разработке сайта, заключается в том, что информация в нем должна быть представлена в понятном и удобном для использования виде. Если сайт состоит из нескольких страниц (документов), то необходимо продумать хорошую систему навигации, чтобы пользователь не заблудился в информационном пространстве и не тратил зря свои время и деньги на поиски необходимого или, по крайней мере, выхода из вашего лабиринта. Графика и визуальные эффекты должны привлекать внимание и способствовать быстрому и правильному восприятию информации, а не раздражать пестротой и долгой загрузкой в браузер. Считается хорошим тоном, если все страницы сайта выполнены в едином стиле. При этом создается впечатление соотнесенности компонентов сайта с единой концепцией. Именно по внешнему виду отдельных страниц посетитель ощущает, находится ли он еще внутри сайта или уже вышел за его границы.

Дизайн — важная и интересная сфера деятельности. Это то, что стоит впереди или на поверхности ваших многоэтажных разработок. Однако сам по себе, без информационного содержания, он ничего не стоит. Зрелый дизайнер, подобно архитектору, умеет сочетать внешний вид своего сооружения с его функциональностью и окружающей средой. Дизайнер должен немало потрудиться, чтобы проект нашел своего потребителя, принес пользу ему и своему автору, а также не нанес вреда, даже нечаянно, участникам сети. Дизайн представляет содержание в привлекательной, понятной и удобной пользователю форме. Если содержание бедно или вовсе отсутствует, то и представлять нечего. Главное в Интернете — информация в чистом виде. Адаптация этой информации к тем или иным категориям читателей — дело дизайна.

Все информационные ресурсы сайта располагаются на одном или нескольких удаленных компьютерах, входящих в Интернет и играющих роль *серверов*. Компьютер пользователя, обращающийся к ресурсу на сервере, является его клиентом. Web-браузер (например, Microsoft Internet Explorer или Netscape Navigator) является *клиентским программным обеспечением* для приема и отображения информационных ресурсов, а также для передачи данных на сервер.

В простейшем и наиболее распространенном случае браузер обращается к серверу, передавая ему URL-адрес (Uniform Resource Locator, унифицированный адрес ресурса) требуемого файла. Обычно это файл с расширением имени htm или html. Если имя файла в URL-адресе не указано, то подразумевается index.htm или index.html. Сервер ищет этот файл у себя или в сети

и в случае успеха просто пересылает его запросившему клиенту без какой бы то ни было предварительной обработки. Браузер клиента обрабатывает принятый файл и отображает результат в своей области просмотра. Например, если принятый файл содержит HTML-документ, то он обрабатывается интерпретатором языка HTML, а результаты обработки отображаются в окне браузера. Большинство сайтов в Интернете функционируют описанным выше способом. Мы будем называть их статическими, имея в виду лишь то, что сервер только принимает от клиента запрос файла и отправляет ему требуемый файл. Разумеется, это не означает, что содержимое статического сайта не может изменяться. Таким образом, ресурсы *статических сайтов* хранятся на сервере и пересылаются клиентам в ответ на их запрос, а обрабатываются только клиентом.

Кроме статических, существуют и так называемые *динамические сайты*. Они отличаются тем, что все или некоторые запрашиваемые клиентом ресурсы предварительно обрабатываются на сервере, а клиенту передается результат этой обработки. Клиент может передавать на сервер не только URL-адрес, но и данные, вводимые пользователем или генерируемые сценариями. Эти данные затем обрабатываются и/или сохраняются на диске сервера, а результаты обработки могут пересылаться клиенту. Таким образом, ресурсы динамических сайтов обрабатываются не только клиентом, но и сервером.

Многие интернет-провайдеры предоставляют бесплатную возможность размещения статических сайтов на своих серверах. Большинство частных, информационных и рекламных сайтов являются статическими. При этом взаимодействие владельца сайта с его посетителями обычно производится посредством электронной почты.

Размещение и поддержка динамических сайтов, требующих вычислительных ресурсов сервера (выполнения серверных сценариев), обычно осуществляется по специальному договору с провайдером и за определенную плату. Как правило, это сайты поисковых систем (например, Яндекс, Rambler и др.), а также коммерческих и производственных фирм, принимающих от посетителей заказы на товары и услуги не по электронной почте, а непосредственно со страницы сайта. Однако и частные лица, создающие, например, счетчики количества посещений сайта, гостевые книги, форумы и другие приложения, работающие с файловой системой и базами данных, используют серверные сценарии.

1.2. Как устроен HTML-документ?

HTML-документы являются основой любого сайта, поскольку именно они определяют, что и каким образом будет отображаться в окне браузера. В данном разделе мы кратко рассмотрим основные понятия и элементы языка HTML, на котором пишутся коды, формирующие содержимое Web-страниц.

В большинстве случаев HTML-документы создаются с помощью средств визуальной разработки, таких как Macromedia Dreamweaver, Microsoft FrontPage и др. Эти средства избавляют от необходимости писать коды на языке HTML вручную. Однако понимание этих кодов необходимо для применения сценариев — главной цели данной книги.

1.2.1. Начальные сведения

HTML-документы сохраняются на диске компьютера как обычные текстовые файлы с расширением htm или html. Они отличаются от обычных текстов только тем, что содержат специальные метки (команды, инструкции, дескрипторы, теги). Эти метки указывают программе просмотра (обозревателю, браузеру), что и как показывать на экране компьютера. Так, например, в HTML-документе можно указать, что данный текст следует выделить жирным наклонным шрифтом, а в таком-то месте разместить графическое изображение из заданного графического файла. Кроме того, можно создать специальную строку текста или картинку, щелчок левой кнопкой мыши на которой приведет к показу содержимого другого документа. Это так называемая *гиперссылка*. Текст, содержащий гиперссылки, называется *гипертекстом* и формируется с использованием специального языка HTML (HyperText Markup Language, язык разметки гипертекста). Файлы, содержащие собственно информацию, предназначенную для демонстрации посетителю Web-страницы, обычно называются *ресурсами* или *содержимым* (content) *сайта*.

Язык HTML довольно прост для понимания и применения. Все его теги представляют собой некоторые предопределенные ключевые слова (имена тегов), заключенные в угловые скобки < и >. Регистр, в котором записаны теги, не имеет значения. HTML-документ начинается с тега `<html>` и заканчивается тегом `</html>`. Между этими тегами могут располагаться другие теги и/или обычный текст. Большинство тегов являются парными: тегу `<тег>` соответствует тег `</тег>`. Первый из них называется *открывающим*, а второй — *закрывающим*. Такие теги еще называют *контейнерными* (или просто контейнерами), имея в виду, что между открывающим и закрывающим тегами можно разместить как в контейнере другие теги. Ниже приведен пример HTML-документа, который содержит текст, отображаемый в браузере жирным шрифтом, причем слово "друзья" выводится курсивом:

```
<html>
<b>Привет, <i>друзья</i></b>
</html>
```

Здесь в контейнере `<html>` находится контейнер `<b>`, задающий жирный шрифт для всей заключенной в нем строки, а в контейнере `<b>` расположен контейнер `<i>`, задающий шрифт курсивом для содержащейся в нем строки.

Вы можете записать рассмотренный выше код в любом текстовом редакторе, например Блокноте Windows (исполняемый файл notepad.exe), и сохранить на жестком диске в файле с расширением htm или html, а затем открыть его в браузере. Вид данного документа в окне браузера Microsoft Internet Explorer показан на рис. 1.1.

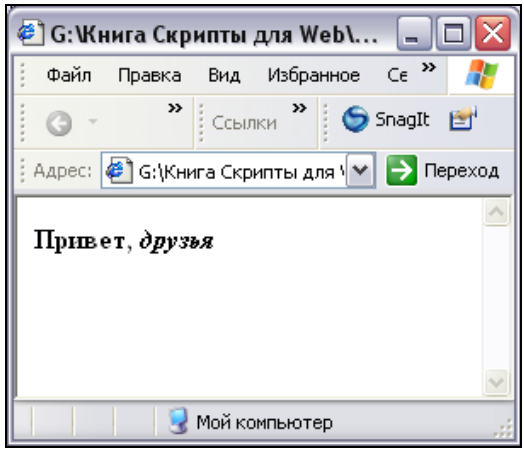


Рис. 1.1. Пример простого HTML-документа

Внимание

Между открывающейся угловой скобкой и ключевым словом (именем) тега не должно быть пробелов. В противном случае тег будет не интерпретироваться, а просто выводиться как обычный текст.

Теги могут иметь параметры, называемые *атрибутами*. Атрибуты, в свою очередь, могут иметь значения (аргументы). Атрибуты со значениями задаются парой вида *имя_атрибута=значение*. Имя атрибута является ключевым словом, а допустимое значение определяется спецификацией этого атрибута — описанием того, какие значения возможны и как их задавать.

Некоторые теги имеют атрибуты, которые обязательно должны быть указаны. Например, для вставки в HTML-документ графического изображения используется тег `<img>` с обязательным атрибутом `src="`*адрес_файла*`"`. В качестве значения атрибута `src` используется имя (при необходимости с указанием пути) или URL-адрес графического файла. Например,

```
<img src="http://www.myserver.ru/images/picture.jpg">
```

Если не указывать необязательные атрибуты, то их значения будут равны принятым по умолчанию. Например, тег `<img>` имеет еще атрибуты `width` и `height`, задающие соответственно ширину и высоту изображения в окне браузера. Если их не указать, то изображение будет занимать на экране прямоугольную область в соответствии со своими оригинальными размерами.

Если в теге необходимо указать несколько предусмотренных для него атрибутов, то они могут следовать за именем тега в произвольном порядке. Разделителем между ними является один или несколько пробелов.

Внимание

Теги, которые браузер не может проинтерпретировать (в том числе с неверно записанными именами), просто игнорируются им.

Часть HTML-кода, заключенная между `<!` и `>`, интерпретируется браузером как *комментарий*, т. е. не выполняется и не выводится на экран. Например, при интерпретации браузером следующего HTML-кода ничего не будет отображено:

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

Содержимое HTML-документа в целом, а также запись отдельного тега может занимать одну или несколько строк. При этом имена тегов, атрибутов и значения атрибутов не следует разрывать для переноса на другую строку. Количество пробелов между элементами тегов и между самими тегами, отступы и количество пустых строк между элементами HTML-документа не влияют на его отображение в окне браузера, а служат лишь для удобства чтения HTML-кода.

Если не принять специальных мер, элементы HTML-документа располагаются рядом друг с другом по горизонтали и вертикали в порядке их записи в коде, а также в зависимости от того, как они вписываются в размеры окна браузера. Это так называемая *базовая линейная модель* интерпретации HTML-кода и отображения ее результатов. Чтобы определить другой порядок размещения элементов в окне браузера, следует использовать специальные теги, атрибуты и/или правила каскадных таблиц стилей (CSS). Например, чтобы некоторый элемент был размещен ниже предыдущего, достаточно перед тегом, определяющим данный элемент, записать тег `<br>`. Для задания нового абзаца при форматировании текста или для размещения элемента ниже предыдущего с пропуском одной строки достаточно использовать тег `<p>`.

Множество всех тегов HTML можно разбить на следующие группы:

- структура документа;
- ссылки;
- вставка содержимого из внешних источников;
- разметка страницы;
- форматирование текста;
- формы и элементы пользовательского интерфейса.

В следующих разделах данной главы мы кратко рассмотрим основные теги этих групп.

1.2.2. Структура документа

Тег *<!DOCTYPE>*

Во многих HTML-документах, созданных в специальных редакторах типа Dreamweaver, перед открывающим тегом `<html>` нередко можно увидеть тег `<!DOCTYPE>`, с помощью которого объявляется тип и формат содержимого документа. Этот же тег вставляется и в XML-документы. Однако современные Web-браузеры обрабатывают документы и без него. 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>`.

Возможны следующие версии DTD (Document Type Definition — определение типа документа):

- Strict DTD — исключает все нерекомендованные консорциумом W3C теги и атрибуты, такие как `<FONT>`, `<CENTER>`, `align` и другие, связанные с форматированием, которые можно заменить соответствующими параметрами CSS.

  Пример:

  ```
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/HTML4.01/strict.dtd">
  ```

- Transitional DTD — включает все элементы версии Strict DTD, а также все нерекомендованные элементы для обеспечения обратной совместимости со всеми существующими браузерами.

  Пример:

  ```
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/HTML4.01/loose.dtd">
  ```

- Frameset DTD — включает такие же элементы, что и Transitional DTD, а также элементы, необходимые для Web-страниц с фреймовой структурой.

  Пример:

  ```
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
  "http://www.w3.org/TR/HTML4.01/frameset.dtd">
  ```

Тег *<head>*

Контейнерный тег `<head>` определяет раздел заголовка HTML-документа и может встречаться в нем не более одного раза. В этом теге размещают другие контейнерные теги:

- `<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.

Тег *<meta>*

Информация, содержащаяся в тегах `<meta>`, не отображается браузером, однако имеет важное значение. В частности, тег позволяет задать кодовую страницу языка просмотра документа, ключевые слова, по которым ваш документ будут искать поисковые системы, и т. д. Вот типичный пример использования тегов `<meta>`:

```
<html>
  <head><title>Пример<title>
    <meta http-equiv="Content-Type" content="text/html; charset=koi8-r">
    <meta name="keywords"
          content="Примеры html скрипты сценарии javascript">
  </head>
```

```
  ... здесь располагаются другие теги
</html>
```

Рассмотрим атрибуты тега `<meta>`.

Группа атрибутов *HTTP-EQUIV* (HTTP-эквиваленты)

Данная группа включает следующие атрибуты.

- `EXPIRES` — дата устаревания документа.

  После истечения указанного срока документ будет каждый раз загружаться заново, а не браться из кэша на вашем локальном диске.

  Формат даты: RFC850.

  Пример:

  ```
  <META HTTP-EQUIV="EXPIRES" CONTENT="Wed, 23 Feb 2005 08:25:53 GMT">
  ```

- `PRAGMA` — управление кэшированием.

  Возможно одно значение `NO-CACHE`, т. е. данный документ не кэшируется браузером. В этом случае запрашиваемая страница будет браться с сервера, а не из кэша (буфера) пользовательского компьютера.

  Пример:

  ```
  <META HTTP-EQUIV="PRAGMA" CONTENT="NO-CACHE">
  ```

- `CONTENT-TYPE` — тип документа и его кодировка.

  Выбор кодовой страницы для правильного отображения символов браузером.

  Примеры:

  ```
  <META HTTP-EQUIV="CONTENT-TYPE" CONTENT="text/html;
        charset=windows-1251">
  <meta http-equiv="Content-Type" content="text/html; charset=koi8-r">
  ```

- `CONTENT-LANGUAGE` — указание языка документа.

  Значение этого параметра может использоваться как поисковыми роботами, так и Web-серверами.

  Формат: *<Язык>-<Диалект>*.

  Примеры:

  ```
  <META HTTP-EQUIV="CONTENT-LANGUAGE" CONTENT="en-GB">
  <META HTTP-EQUIV="CONTENT-LANGUAGE" CONTENT="ru">
  ```

- `REFRESH` — время (в секундах), через которое произойдет автоматическая перезагрузка документа или переход на другой документ с заданным URL.

  Формат: "*время*" или "*время; URL*"

Пример:

```
<META HTTP-EQUIV="REFRESH" CONTENT="30">
```

Здесь указано, что периодическую перезагрузку документа следует производить через 30 секунд.

```
<META HTTP-EQUIV="REFRESH" CONTENT="5; http://www.microsoft.com">
```

Здесь указано, что спустя 5 секунд следует перейти к документу по указанному адресу.

Автоматическую периодическую перезагрузку документа обычно назначают в случаях часто обновляемых данных (котировки акций, результаты спортивных соревнований и т. п.).

- `CACHE-CONTROL` — управление кэшированием.

  Возможные варианты: кэширование в общем (`PUBLIC`) или частном (`PRIVATE`) кэше. Документ вообще не кэшируется (`NO-CACHE`) или кэшируется, но не сохраняется (`NO-STORE`).

  Пример:

```
<META HTTP-EQUIV="CACHE-CONTROL" CONTENT="NO-STORE">
```

Группа атрибутов *NAME* (имя)

К этой группе относятся следующие атрибуты.

- `DESCRIPTION` — описание документа. Один из наиболее важных параметров. Информация, содержащаяся в нем, влияет на результаты поиска, осуществляемого поисковыми системами. В общем случае вид результатов поиска, как правило, выглядит так:
  - URL документа;
  - название документа (содержимое тега `<title>`);
  - описание документа, т. е. `DESCRIPTION` или начальный фрагмент HTML-документа, если `DESCRIPTION` отсутствует (нередко — и то и другое). В первом случае пользователь получает достаточно краткое и информативное описание документа, а во втором случае это может быть бессмысленный набор слов или несколько первых фраз из вашего документа;
  - рейтинг (коэффициент соответствия документа запросу пользователя).

  Пример:

```
<META NAME="DESCRIPTION"
      CONTENT="Описание данного документа, до 100 символов">
```

- KEYWORDS — ключевые слова. Набор слов и фраз, наиболее полно характеризующих данный документ, которые являются основным критерием поиска вашего документа поисковыми системами. В конечном счете, эти слова учитываются при выдаче результатов поиска и способствуют повышению рейтинга вашего сайта.

  Пример:

```
<META NAME="KEYWORDS"
      CONTENT="Примеры html скрипты сценарии javascript">
```

- DOCUMENT-STATE — статус документа. Данный параметр управляет частотой индексации вашего документа поисковыми серверами и может принимать два значения.

  - STATIC (документ статичен, т. е. не меняется, и, следовательно, индексировать его нужно только один раз).

    Пример:

```
<META NAME="DOCUMENT-STATE" CONTENT="STATIC">
```

  - DYNAMIC (для часто изменяющихся документов, которые нужно реиндексировать).

    Пример:

```
<META NAME="DOCUMENT-STATE" CONTENT="DYNAMIC">
```

- ROBOTS — управление процессом индексации. Варианты:

  - INDEX — возможность индексирования данного документа (иначе — NOINDEX);
  - FOLLOW — возможность индексирования всех документов, на которые есть ссылки в данном HTML-файле (иначе — NOFOLLOW);
  - ALL — одновременное выполнение условий INDEX и FOLLOW;
  - NONE — одновременное выполнение условий NOINDEX и NOFOLLOW.

  Пример:

```
<META NAME="ROBOTS" CONTENT="INDEX, NOFOLLOW">
```

- RESOURCE-TYPE — тип ресурса. Для обычных HTML-документов значение этого параметра устанавливается равным "DOCUMENT".

  Пример:

```
<META NAME="RESOURCE-TYPE" CONTENT="DOCUMENT">
```

- UPDATED — дата обновления страницы.

  Пример:

  ```
  <META NAME="UPDATED" CONTENT="25.11.04">
  ```

- URL — базовый URL-адрес; определяет, какой документ следует индексировать (чтобы не обрабатывать "зеркала").

  Пример:

  ```
  <META NAME="URL" LANG="ru" CONTENT="http://www.admiral.ru/hp/dunaev">
  ```

- AUTHOR — информация об авторе данного документа.

  Пример:

  ```
  <META NAME="AUTHOR" CONTENT="Вадим Дунаев">
  ```

- COPYRIGHT — информация об авторских правах.
- GENERATOR — название программы, создавшей HTML-код.

Допустимо, но не обязательно, добавлять в метатеги атрибут LANG, указывающий язык данных.

Пример:

```
<META NAME="KEYWORDS" LANG="ru" CONTENT="музыка видео графика">
```

Примечание

Если HTML-документ без тегов <meta> 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> и несколько тегов <meta>.

Тег *<base>*

В HTML-документах обычно используются *относительные ссылки* на ресурсы (другие HTML-документы, графику и т. п.). Например, в ссылке нередко приводится только имя файла. В этом случае предполагается, что его местоположение совпадает с расположением документа, из которого этот файл вызывается. Однако можно явно указать базовый URL-адрес, который будет использоваться во всех случаях применения относительных ссылок. Это делается с помощью тега <base> с атрибутом href, значением которого является базовый URL-адрес:

```
<base href="URL-адрес">
```

Пример:

```
<html>
  <head>
    <base href="//www.admiral.ru/hp/dunaev/images">
  </head>
  <img src="picture.jpg">
</html>
```

В этом примере графическое изображение имеет относительную ссылку (просто имя файла). Если базовый URL-адрес не был бы указан, то браузер искал бы файл picture.jpg в той же папке, где расположен HTML-документ со ссылкой на него. Поскольку базовый URL-адрес задан, то браузер будет искать графический файл в папке /hp/dunaev/images на Web-узле **www.admiral.ru**.

При указании в теге `<base>` базового URL-адреса относительные ссылки продолжают работать, даже если вы переместите HTML-документ в другую папку. Это обеспечивает определенные удобства при работе с локальной версией HTML-документа в условиях, когда компьютер подключен к сети.

Тег *<link>*

Тег `<link>` предназначен для обозначения связей между документами в больших и сложных по своей структуре узлах. Атрибут `href` содержит ссылку на другой документ, связанный с текущим. Адрес может быть абсолютным или относительным. Атрибут `rel` указывает отношение между документами:

- `home` — расположение домашней страницы Web-узла;
- `next` — расположение следующего документа серии, связанного с текущим;
- `previous` — расположение предыдущего документа серии, связанного с текущим;
- `up` — расположение следующего документа вверх по иерархии;
- `copyright` — расположение документа с информацией об авторских правах для данного Web-узла;
- `stylesheet` — расположение файла с таблицей стилей.

Используются также и другие значения для атрибута `rel`: `glossary`, `help`, `toc`, `index`, `contents`.

Пример:

```
<link rel="stylesheet" href="http://mysite.ru/styles/mystyle.css">
```

В одном и том же HTML-документе тег `<link>` может присутствовать несколько раз.

Тег *<style>*

В контейнерном теге `<style>` размещаются правила каскадных таблиц стилей, определяющих внешний вид и позиционирование элементов HTML-документа. Более подробно каскадные таблицы стилей будут рассмотрены в *разд. 1.3*. В одном и том же HTML-документе тег `<style>` может встречаться несколько раз. При этом они могут размещаться и вне тега `<head>`.

Тег *<script>*

В контейнерном теге `<script>` размещается код сценария (скрипт), выполняемый браузером. В одном и том же HTML-документе тег `<script>` может встречаться несколько раз. При этом они могут размещаться и вне тега `<head>`. Внутри тега `<script>` не могут находиться теги HTML и правила каскадных таблиц стилей.

Код сценария может быть написан на языке JavaScript или VBScript. Последний понимает только браузер Internet Explorer. Языком по умолчанию считается JavaScript. Чтобы явно указать, на каком языке написан сценарий, применяется атрибут `language`, который может принимать следующие значения:

- `"javascript"`, `"JScript"`;
- `"VBScript"`, `"VBS"`.

Редакция языка JavaScript для Internet Explorer называется JScript. Вместе с тем в Internet Explorer можно использовать и `language="javascript"`. В браузере Netscape Navigator значимым является только `language="javascript"`, ссылка `language="JScript"` будет им проигнорирована. Поэтому при разработке сценариев, рассчитанных для различных браузеров, рекомендуется использовать ссылку `language="javascript"`. В примерах сценариев на языке JavaScript, приводимых в настоящей книге, мы не будем указывать атрибут `language` из соображений экономии места.

Современные браузеры распознают еще и версию языка. Например, Internet Explorer 4.0 и Netscape Navigator 4.0 знают версию языка JavaScript 1.2. Во время написания этой книги была доступна версия JavaScript 1.5, которую распознают Internet Explorer 6.0 и Netscape Navigator 6.0. Если версия языка JavaScript в значении атрибута `language` указана, а браузер не знает ее, то сценарий может быть просто проигнорирован. Если же версия языка не указана, то браузер, которому она не знакома, может неправильно выполнять некоторые выражения сценария или даже выдавать сообщения об ошибках.

Сценарий может размещаться и в отдельном файле, обычно имеющем расширение js. В этом случае в теге `<script>` следует указать атрибут `src` с именем или URL-адресом файла. Например,

```
<script src="http://www.myserver.ru/myscript.js"></script>
```

При этом между тегами `<script>` и `</script>` ничего не записывается.

Старые браузеры, появившиеся раньше JavaScript, игнорируют теги `<SCRIPT>` и `</SCRIPT>`, а все, что находится между ними, интерпретируют как текстовое содержимое HTML-документа. Результат может быть самым неожиданным. Чтобы уменьшить вероятность отображения кода сценария в окне старого браузера, следует заключить его в дескрипторы комментария `<!--` и `-->`. Новые браузеры, поддерживающие сценарии, будут игнорировать эти символы, выполняя код сценария, а старые браузеры (не понимающие сценарии), наоборот, станут игнорировать код сценария. Вот как используются символы комментария предохранения сценария от старых браузеров:

```
<SCRIPT LANGUAGE="JavaScript" >
  <!--
    // код сценария
  //-->
</SCRIPT>
```

Обратите внимание на заключительные символы тега комментария, перед которыми стоят две косые черты. Без них JavaScript будет пытаться интерпретировать символы `-->` как заключительные символы комментария HTML, а с ними он просто их проигнорирует.

Внимание

В примерах, приводимых в настоящей книге, мы не будем указывать символы комментария для адаптации к старым браузерам из соображений экономии места. Однако помните, что современная культура оформления сценариев обязывает нас делать это в своих практических приложениях.

Тег *<body>*

Основной раздел HTML-документа, содержащий текст, ссылки, графические, звуковые и другие объекты, размещается вне раздела заголовка, задаваемого тегом `<head>`. Часто основной раздел заключают в контейнерный тег `<body>`, который может встречаться в HTML-документе не более одного раза и не может содержать тег заголовка `<head>`.

Например:

```
<html>
  <head>
```

```
    ... здесь располагаются теги раздела заголовка документа
  </head>
  <body>
    ... здесь располагаются теги основного раздела документа
  </body>
</html>
```

Вообще говоря, использование тега `<body>` не обязательно. Однако он обладает атрибутами, с помощью которых можно задать параметры сразу для многих элементов, содержащихся в `<body>`:

- `alink` — цвет активной ссылки;
- `link` — цвет еще не использованной (непросмотренной) ссылки;
- `vlink` — цвет уже использованной (просмотренной) ссылки;
- `text` — цвет текста;
- `bgcolor` — цвет фона документа;
- `background` — URL-адрес графического изображения, используемого в качестве фона документа;
- `leftmargin`, `rightmargin`, `topmargin`, `bottommargin` — ширина в пикселах левого, правого, верхнего и нижнего полей (отступов) документа соответственно;
- `bgproperties` — если установлено значение `fixed` (`bgproperies=fixed`), то фоновое изображение (атрибут `background`) не будет прокручиваться;
- `scroll` — значение `yes` устанавливает, а значение `no` удаляет полосы прокрутки окна браузера.

Ни один из перечисленных выше атрибутов не является обязательным. По умолчанию браузер заполняет фон сплошным цветом в зависимости от настройки пользователя. Чтобы настройки пользователя не могли испортить восприятие вашего документа, следует самому зафиксировать все цвета (фона, текста и ссылок). Если вы не зададите цвета явным образом, то настройки пользовательского браузера скорее всего позволят прочитать текст и ссылки, однако при этом дизайн вашего документа окажется не контролируемым вами.

Значение цвета задается либо именем цвета из предопределенного списка, либо шестнадцатеричным числом в формате `#XXXXXX`, где `X` — шестнадцатеричная цифра (0, 1, 2, ..., 9, a, b, c, d, e, f). При этом используется модель RGB представления цвета: первые две цифры задают яркость красной (Red), вторые две цифры — зеленой (Green), а последние две цифры — синей (Blue) составляющей цвета. Каждая составляющая цвета может иметь одно из 256 значений (оттенков). Например, черному цвету соответствует значение `#000000`

(все составляющие имеют минимальную яркость); белому цвету соответствует значение `#ffffff` (все составляющие имеют максимальную яркость); если значения яркости всех трех составляющих цвета одинаковы, то получается серый цвет (возможны 256 оттенков серого от черного до белого). Ниже приведены два варианта задания красного цвета для текста и желтого для фона:

```
<body text=red bgcolor=yellow>
<body text=#ff0000 bgcolor=#ffff00>
```

Список предопределенных имен цветов содержит более сотни элементов и поиск в нем названий цветов, отличных от основных, довольно затруднителен. Однако цвета из этого списка соответствуют так называемой *Web-палитре*. Использование цветов из этой палитры гарантирует их одинаковое воспроизведение всеми браузерами.

Теги * * и **

Для выделения некоторого фрагмента документа, чтобы иметь возможность задать его общие параметры, служат контейнерные теги `<div>` и `<span>`. Контейнер `<div>` может содержать различные элементы (текст, графические изображения, ссылки, объекты и т. п.), а `<span>` обычно применяется к фрагментам текста.

Общие параметры для элементов, включенных в данные контейнеры, обычно назначаются с помощью таблиц стилей (тег `<style>` или атрибут `style`). В частности, можно задать произвольное относительное положение фрагментов документа, в том числе и наложение их друг на друга, имитируя тем самым создание слоев. Дополнительно к этому тег `<div>` имеет атрибут `align` с возможными значениями `center`, `left` и `right`, позволяющий выровнять весь фрагмент документа по центру, левому и правому краю окна браузера соответственно. В следующем примере фрагмент документа, содержащий два изображения, выравнивается по центру окна браузера:

```
<html>
  <head><title>Пример</title></head>
  <body>
    Пример использования контейнера:
    <div align=center>
      <img src="picture1.jpg">
      <img src="picture2.jpg">
    </div>
  </body>
</html>
```

Теги, влияющие на расположение элементов

Тексты и теги видимых элементов записываются в документе друг за другом и в этой же последовательности выполняются браузером. В результате они отображаются в окне браузера. Многие элементы по умолчанию (если не предусмотреть специальных мер) располагаются рядом друг с другом по горизонтали и/или вертикали в зависимости от того, как они вписываются в размеры окна браузера. В крайнем случае, при невозможности отобразить все элементы целиком в окне браузера автоматически появляются полосы прокрутки. Впрочем, отображение этих полос можно и запретить. Для контроля порядка расположения элементов в окне браузера служат специальные средства: теги, атрибуты и таблицы стилей. В данном разделе будут рассмотрены следующие теги, влияющие на расположение любых элементов:

- `<p>` — абзац, т. е. переход на следующую строку и пропуск одной строки;
- `<br>` — переход на следующую строку;
- `<nobr>` — невозможность переноса слов и режима обтекания текстом;
- `<wbr>` — "мягкий" перенос в блоке текста, заключенного в контейнерный тег `<nobr>`;
- `<center>` — выравнивание заключенных в него элементов по центру;
- `<hr>` — размещение горизонтальной разделительной полосы.

Тег *<p>*

Тег `<p>` используется для выделения блоков текста (абзацев). Этот тег приводит к вставке одной пустой строки и размещению находящегося за ним видимого элемента в следующей строке (если, разумеется, параметры стиля этого элемента не задают какое-либо другое расположение). Например, чтобы два элемента находились друг под другом и между ними была пустая строка, достаточно между тегами этих элементов вставить тег `<p>`:

```
<html>
  <img src="picture1.jpg">
  <p>
  <img src="picture2.jpg">
</html>
```

Тег `<p>` является контейнерным. Однако закрывающий тег `</p>` редко используется, поскольку браузер считает, что начало следующего абзаца означает конец предыдущего:

```
<html>
  Ниже показаны две мои фотографии:
```

```
  <p>
  <img src="picture1.jpg">
  <p>
  <img src="picture2.jpg">
</html>
```

Очевидно, чтобы вставить несколько пустых строк, достаточно записать тег `<p>` несколько раз подряд.

Тег `<p>` имеет необязательный атрибут `align` с возможными значениями `center`, `left`, `right` и `jastify`, позволяющими выровнять весь фрагмент документа по центру, левому краю, правому краю и одновременно по двум сторонам окна браузера соответственно. Используя атрибут `style`, можно определить специфические параметры расположения. Необязательный атрибут `title` задает текстовую строку, которая появляется при наведении указателя мыши на содержимое тега `<p>` (всплывающая подсказка).

Теги *
*, *<nobr>* и *<wbr>*

Тег `<br>` служит для перехода на другую строку. Он не является контейнерным и, в отличие от тега `<p>`, не вставляет дополнительной пустой строки. Тег `<br>` имеет необязательный атрибут `clear`, принимающий значения `left`, `right` и `all`. Обычно тег `<br>` используется без указания этого атрибута. Однако иногда требуется прекратить выравнивание элемента документа (например, изображения) относительно текста. В этом случае атрибут `clear` позволяет указать, какое именно выравнивание следует прекратить.

Контейнерный тег `<nobr>` предотвращает перенос заключенной в него строки текста. Если при этом строка выходит за границу окна браузера, то появляется горизонтальная полоса прокрутки.

В тексте, заключенном в контейнер `<nobr>`, можно указать место возможного перевода строки ("мягкий" перенос слов), который будет выполнен браузером лишь при необходимости. Это делается с помощью тега `<wbr>`.

Тег *<hr>*

При оформлении документа, чтобы отделить один фрагмент от другого, нередко используют разделительные полосы, задаваемые тегом `<hr>`. Разделительная полоса всегда выводится в новую строку. Элемент, следующий за тегом `<hr>`, выводится ниже разделительной полосы.

Тег `<hr>` имеет следующие необязательные атрибуты:

- `size` — толщина полосы в пикселах;
- `width` — длина полосы в пикселах;

- `align` — способ выравнивания (принимает значения `center`, `left` и `right` для выравнивания по центру, левому и правому краю окна браузера соответственно);
- `noshade` — атрибут, не имеющий значений; если указан, то создается сплошная черная полоса без тени.

Специальные атрибуты

Большинство тегов имеют, кроме прочих, необязательные специальные атрибуты:

- `class` — для связи с таблицей стилей, определенной контейнерным тегом `<style>`;
- `style` — для определения таблицы стилей только для данного элемента (не путайте тег `<style>` и атрибут `style`);
- `id` — идентификатор для обеспечения доступа к элементу из сценария и/или таблиц стилей (подобно имени переменной);
- *атрибут-событие* — для обеспечения связи со сценарием, который выполняется при наступлении соответствующего события (например, `onclick`, `onload`, `onchange` и др.);
- `title` — для определения строки текста, появляющейся при наведении указателя мыши на элемент, заданный тегом, в котором этот атрибут установлен (т. е. для задания всплывающей подсказки).

При описании тегов HTML перечисленные выше атрибуты не всегда упоминаются. Значения и применение этих атрибутов будут подробно рассмотрены в соответствующих разделах (в нужном месте). Так, атрибуты `class` и `style` будут описаны при рассмотрении каскадных таблиц стилей (CSS); атрибуты `id` и атрибуты-события — при рассмотрении сценариев; атрибут `title` не нуждается в особых комментариях. На данном этапе достаточно только иметь в виду, что кроме описываемых атрибутов, теги могут иметь еще и перечисленные выше специальные атрибуты, обеспечивающие взаимосвязь HTML, CSS и сценариев.

1.2.3. Ссылки

Ссылки (или *гиперссылки*) позволяют щелчком кнопкой мыши на выделенном фрагменте текста или изображении перейти к другому файлу или разделу текущего документа. Ссылки применяются в большинстве существующих Web-страниц и являются их главным инструментом. Если документ содержит ссылки на другие документы, то он называется *гипертекстовым*.

Ссылки могут быть текстовыми и графическими в зависимости от того, что является указателем ссылки (видимым элементом документа, щелчок на котором левой кнопкой мыши инициализирует переход к другому документу). Текстовые ссылки представляют собой выделенное слово или целую фразу. Выделение ссылки производится цветом или подчеркиванием, в зависимости от настройки браузера. В графических ссылках роль чувствительного к щелчку элемента играет изображение. Можно также создать комбинированные ссылки, указателем в которых является и изображение, и текст. Ссылки на фрагменты текущего документа (внутренние ссылки) организуются особым образом. В любом случае ссылки задаются контейнерным тегом `<a>`.

Для создания ссылки на другой документ используется следующий синтаксис:

```
<a href="адрес_ссылки">указатель_ссылки</a>
```

Здесь атрибут `href` (от англ. *hipper reference* — гиперссылка) принимает в качестве значения URL-адрес или имя файла (возможно, с указанием пути), к которому следует перейти при инициализации ссылки; *`указатель_ссылки`* — текст или тег `<img>`, вставляющий графическое изображение.

По умолчанию вызываемый при активизации ссылки документ открывается в текущем окне браузера, заменяя документ, содержащий эту ссылку. Это происходит, если браузер способен открыть документ в своем окне (некоторые документы открываются в окне соответствующих приложений). Однако вы можете определить, в каком окне браузера или фрейме следует открыть вызываемый документ. Это делается с помощью атрибута `target`, который принимает следующие значения:

- *`имя_окна`* — имя окна или фрейма;
- `_parent` — указатель на родительский фрейм;
- `_blank` — указатель на новое окно;
- `_top` — указатель на фрейм верхнего уровня;
- `_self` — указатель на тот же самый (текущий) фрейм.

Если требуется, чтобы при наведении указателя мыши на указатель ссылки появлялась всплывающая подсказка (поясняющий текст), то в теге `<a>` следует использовать атрибут `title`, принимающий в качестве значения текст всплывающей подсказки.

Текстовые ссылки

Структура текстовой ссылки имеет следующий вид:

```
<a href="адес_ссылки">текст_ссылки</a>
```

Например, следующий тег описывает ссылку на HTML-файл мой_документ.htm, при этом ссылка на экране будет представлена текстом **Щелкните здесь**:

```
<a href="мой_документ.htm">Щелкните здесь</a>
```

Отметим, что браузер не выводит на экран имя файла, к которому требуется перейти по ссылке, а лишь показывает текст, заключенный в теге между угловыми скобками > и <. Если же вы хотите, чтобы внешне ссылка выглядела как имя файла, на который она ссылается, то просто напишите его имя вместо текста, например:

```
<a href="мой_документ.htm">мой_документ.htm</a>
```

Можно ссылаться не только на другие файлы, но и на собственный файл.

Текст, заключенный в контейнерный тег ссылки <a>, может быть отформатирован как с помощью рассмотренных ранее тегов форматирования, так и с помощью таблиц стилей, которые мы обсудим позже. Например:

```
<a href="мой_документ.htm"><i>Щелкните <b>здесь</b></i></a>
```

Примечание

В приведенных выше примерах ссылка задавалась как имя файла. В этом случае предполагается, что файл находится в той же папке, что и документ, содержащий эту ссылку. Если этого недостаточно, то вы можете указать базовый адрес с помощью тега <base>, использовать относительные и абсолютные пути, а также URL-адрес файла.

Графические и комбинированные ссылки

Структура графической ссылки имеет вид:

```
<a href="адрес_ссылки"><img src="адрес_графического_файла"></a>
```

Например, следующий тег описывает ссылку на HTML-файл мой_документ.htm, при этом ссылка на экране будет представлена изображением из файла picture.gif:

```
<a href="мой_документ.htm"><img src="picture.gif"></a>
```

К графической ссылке можно добавить поясняющий текст:

```
<a href="мой_документ.htm"><img src="picture.gif">Щелкните здесь</a>
```

В этом примере переход к документу мой_документ.htm произойдет при щелчке левой кнопкой мыши как на графическом изображении, так и на поясняющем тексте, поскольку и то и другое находится в контейнерном теге <a>.

Браузер по умолчанию выделяет графические ссылки рамкой вокруг ее графического изображения. Иногда это нежелательно, и чтобы исключить отображение рамки, достаточно использовать в теге `<img>` атрибут `border=0`, как в следующем примере:

```
<a href="мой_документ.htm"><img src="picture.gif" border=0>Щелкните здесь</a>
```

Графические карты ссылок

В рассмотренных выше графических ссылках одному изображению соответствовал один адрес ссылки. Однако имеется и другая возможность. Она заключается в том, чтобы одному изображению сопоставить несколько ссылок, привязав каждую из них к некоторой области изображения. Такие области называют *горячими* или *активными*, а сам технологический прием — *графической картой ссылок* или *сегментированной графикой*. Горячие области графической карты могут быть различной формы: прямоугольной, многоугольной или в виде окружности. Это очень удобный прием оформления группы ссылок, однако при выборе рисунка, служащего основой карты ссылок, следует стремиться к тому, чтобы границы горячих областей были хорошо очерченными и не пересекались. Кроме того, необходимо позаботиться о том, чтобы пользователь понял, что имеет дело с картой ссылок, а не просто с графическим фоном. Для этого можно использовать поясняющие тексты. При наведении на горячую область указатель мыши изменяет свою форму как и при использовании обычных ссылок. При щелчке на горячей области ее границы становятся видимыми.

Возможны два варианта реализации графических карт ссылок: клиентский и серверный. В клиентском варианте вся информация о конфигурации графической карты ссылок размещается в HTML-документе, в котором она применяется. В серверном варианте предполагается использование сценария, который располагается на сервере и выполняет обработку запросов браузера клиента при работе с графической картой ссылок. Кроме того, на сервере должен находиться конфигурационный файл, в котором описываются горячие области карты. Хотя серверный вариант появился раньше, теперь чаще используется клиентский вариант.

Клиентский вариант графической карты ссылок

Графическая карта ссылок в клиентском варианте задается с помощью нескольких тегов. Первым является контейнерный тег `<map>` (карта) с атрибутом `name` для указания имени карты. Имя карты выбирается как имя переменной. Далее, между тегами `<map>` и `</map>` следуют теги `<area>` (область) для зада-

ния горячих областей. Тег <area> имеет ряд атрибутов, описывающих собственно ссылку, а также форму и положение горячей области:

- href — строка, определяющая адрес ссылки;
- shape — определяет форму области; принимает значения:
  - "rect" или "rectangle" (прямоугольник);
  - "poly" или "polygone" (многоугольник);
  - "circle" (круг);
- coords — координаты области, которые задаются в пикселах перечнем чисел, разделенных запятыми; весь перечень заключается в кавычки (для прямоугольника задаются четыре числа — координаты верхнего левого и правого нижнего угла; для многоугольника указываются координаты каждого угла; для круга задаются три числа — координаты центра и радиус); координаты углов задаются в следующем порядке: сначала горизонтальная, а затем вертикальная координата; начало отсчета координат — верхний левый угол изображения;
- title — текстовая строка всплывающей подсказки, появляющейся при наведении указателя мыши на горячую область;
- target — указывает окно браузера или фрейм, в котором следует открыть документ, заданный в адресной части ссылки; имеет такие же значения, как и для тега <a>;
- nohref — атрибут, не имеющий значений; используется для указания, что данная область не имеет адреса перехода. Атрибуты href и nohref взаимно исключают друг друга. Атрибут nohref обычно применяется для того, чтобы исключить переход по ссылке для части уже определенной горячей области.

После закрывающего тега </map> следует указать тег, вставляющий изображение и реализующий привязку карты к нему — это уже известный тег <img>, в котором помимо прочих возможных атрибутов используется атрибут связи с картой:

usemap="#*имя_карты*"

В качестве имени карты приводится значение атрибута name тега <map>.

Допустим, в основе графической карты ссылок находится некоторая топографическая карта (файл Карта.jpg). На ней определены прямоугольная и круглая горячие области, соответствующие двум участкам. При щелчке на горячей области в окне браузера будет выведен документ Район1.htm или

Район2.htm, содержащий, например, описания соответствующих участков местности. Вот пример HTML-кода, реализующего эту задачу:

```
<html>
  <head><title>Графическая карта ссылок</title></head>
  <map name="mymap">
    <area href="Район1.htm" shape="rect"   coords="350,50,450,150">
    <area href="Район2.htm" shape="circle" coords="250,250,50">
  </map>
  <img src="Карта.jpg" usemap="#mymap" width=566 height=554>
</html>
```

Графические карты ссылок обычно используются для создания красочных меню. Средства для разработки таких карт предоставляют Macromedia Dreamweaver, Microsoft FrontPage, Adobe Photoshop и некоторые другие пакеты и утилиты.

Серверный вариант графической карты ссылок

В серверном варианте реализации графической карты ссылок при щелчке на горячей области координаты места щелчка передаются на сервер и обрабатываются серверным сценарием. Эта обработка заключается, по существу, в обращении к таблице ссылок и поиску в ней URL-адреса документа, сопоставленного координатам щелчка. Если такой адрес будет найден, то он передается браузеру; в противном случае возвращается сообщение о том, что место щелчка не находится в горячей области. Таблица соответствий координат всех горячих областей карты и URL-адресов документов хранится на сервере в виде так называемого конфигурационного файла с расширением map.

Для применения серверного варианта графической карты ссылок в HTML-документе, загружаемом в браузер клиента, используется тег `<a>`, атрибут `href` которого содержит адрес конфигурационного файла карты ссылок. Внутри тега `<a>` располагается тег `<img>`, указывающий на изображение, которое служит основой карты ссылок, и содержащий атрибут `ismap`:

```
<a href="адрес_конфигурационного_файла"><img
src="адрес_изображения_карты" ismap></a>
```

Пример:

```
<a href=mymap.map"><img src="mymap.jpg" ismap></a>
```

Примечание

Хорошим правилом является сохранение конфигурационного файла и основного графического изображения карты ссылок в одной папке и под единым именем,

но с разными расширениями. Конфигурационный файл имеет расширение map, а изображение карты ссылок сохраняется в любом файле растрового формата, чаще всего GIF, JPEG или PNG.

Конфигурационный map-файл является обычным текстовым файлом, содержащим описание горячих областей графической карты ссылок. Однако форматы представления этих данных могут быть различными. Поэтому перед реализацией серверного варианта карты ссылок следует выяснить у администратора сервера, какой из форматов конфигурационного файла поддерживается. Сейчас существуют форматы конфигурационных файлов CERN и NCSA (аббревиатуры наименований европейского и американского национального научных центров соответственно).

Формат CERN:

тип_области координаты URL-адрес

Тип области:

- `rect` или `rectangle` — прямоугольник;
- `poly` или `polygone` — многоугольник;
- `circle` — круг;
- `default` — по умолчанию (область изображения, не являющаяся горячей).

Координаты *x* и *y* углов прямоугольника и многоугольника или центра круга разделяются запятой и заключаются в круглые скобки.

Примеры:

```
rect (50,100) (150,200) http://www.myweb.ru
poly (20,10) (30,5) (100,25) (50,30) http://www.rambler.ru
circle (100,150) 80 http://www.admiral.ru/hp/dunaev
```

Формат NCSA:

тип_области URL-адрес координаты

Кроме типов областей, принятых в формате CERN, допускается тип `point` (точечная область). Предполагалось, что в случае нескольких точечных горячих областей по щелчку левой кнопкой мыши должна активизироваться та из них, которая ближе к месту щелчка. Однако при наличии областей типа `default` и `point`, последние могут активизироваться только при точном наведении на них указателя мыши, что практически весьма трудно сделать.

Координаты в формате NCSA разделяются запятыми, но не заключаются в скобки. При этом круглая область задается несколько иначе, чем в формате CERN: координатами центра и любой точки, лежащей на окружности.

Примеры:

```
rect http://www.myweb.ru 50,100,150,200
poly http://www.rambler.ru 20,10,30,5,100,25,50,30
circle http://www.admiral.ru/hp/dunaev 100,150,100,70
```

Внутренние ссылки

Ссылки на разделы одного и того же документа, в котором эти ссылки находятся, называются *внутренними* или *закладками*. Например, на своей странице вы размещаете статью, объемом несколько десятков страниц, как единый HTML-документ. Скорее всего, вам захочется сделать ссылки на предыдущие или последующие разделы этого документа. Предметный указатель к книге также можно оформить с помощью внутренних ссылок.

Для организации внутренней ссылки необходимы *якорь* (anchor) и собственно ссылка. Якорь определяет место в документе, к которому происходит переход по ссылке. Ссылка использует имя якоря вместо имени (адреса) файла.

Формат якоря:

```
<a name="имя_якоря">текст</a>
```

Имя якоря может быть произвольным и задается просто как имя переменной. Текст внутри тега `<a>` для создания якоря не является обязательным и чаще всего не указывается.

Формат ссылки:

```
<a href="#имя_якоря">указатель_ссылки</a>
```

Указатель ссылки может быть как текстовым, так и графическим.

В листинге 1.1 приведен пример, демонстрирующий использование внутренних ссылок.

Листинг 1.1. Использование внутренних ссылок

```
<html>
  <head><title>Внутренние ссылки</title></head>
  <body>
    <h2>Содержание</h2>
    <b>
    <a href="#Глава1">Глава 1. Основы HTML</a>
    <br>
    <a href="#Глава2">Глава 2. Примеры</A>
    </b><p>
```

```
    <a name="Глава1"></a><h2>Глава 1.Основы HTML</h2>
    В этой главе мы собираемся рассмотреть основные элементы языка HTML.
    На языке HTML создаются документы, которые могут быть опубликованы
    в Интернете. После изучения основных конструкций языка мы подробно
    рассмотрим примеры программ (<a href="#Глава2">см. главу 2</a>).
   <p>
   <a name="Глава2"></a><h2>Глава 2. Примеры</h2>
   Здесь мы рассмотрим примеры программ с использованием элементов языка,
   которым посвящена
   <a href="#Глава1">Глава 1.</a>
   Прежде всего, рассмотрим применение текстовых ссылок.
  </body>
</html>
```

Адреса ссылок

Адрес на ресурсы Интернета (значение атрибута `href` тега `<a>`) в общем случае задается в виде URL (Uniform Resourse Locator, унифицированный указатель ресурса). URL-адрес имеет следующую структуру:

протокол://*адрес_сервера*/*адрес_файла*

Адрес файла может быть составным, т. е. кроме имени файла содержать путь к нему; элементы пути разделяются прямым слэшем (наклонной чертой) `/`. Имя файла может быть и не указано. Префикс URL-адреса представляет собой название протокола связи, соответствующий службе Интернета, которая его поддерживает; за названием протокола следует двоеточие и два прямых слэша. В случае URL для электронной почты двойной слэш за двоеточием не указывается. В табл. 1.1 приведены префиксы URL-адресов для основных служб Интернета.

Таблица 1.1. *Наиболее популярные службы Интернета*

Служба в Интернете	Префикс URL-адреса
WWW	**http://**
FTP	**ftp://**
Mail	**mailto:**
Gopher	**gopher://**
UseNet	**news://**
Telnet	**telnet://**

Если вы указываете адрес, начинающийся с **http://**, тем самым вы обращаетесь к ресурсам, доступ к которым осуществляется по протоколу HTTP (HyperText Transfer Protocol, протокол передачи гипертекста). Этот протокол является основным в Интернете при передаче информации, находящейся в HTML-документах. Служба, обеспечивающая обмен HTML-документами, называется *Word Wide Web*.

Префикс адреса **ftp://** означает, что следует использовать протокол передачи файлов (File Transfer Protocol, FTP). Он может применяться при перемещении любых файлов с одного компьютера на другой. В частности, при перекачке файлов вашей страницы на сервер используется именно этот протокол. Протокол FTP обеспечивает высокую надежность передачи файлов. К примеру, если потерю до 10% обычной текстовой информации еще можно пережить, то при передаче программ или сжатых архиватором файлов потери вообще не допустимы — неточно переданная программа просто не будет работать, а архивный файл не распакуется.

Если перед адресом ссылки указывается **mailto:**, это означает, что следует использовать протокол передачи сообщений по электронной почте.

Gopher — служба (значит, и протокол), предназначенная в первую очередь для работы неграфических браузеров. Она предоставляет систему доступа к информации, основанную на меню.

UseNet — служба обеспечения телеконференций, это система типа доски объявлений, на которую вы можете поместить свое сообщение и на которой можно прочитать то, что там разместили другие участники телеконференции.

Ниже приведены примеры ссылок с применением URL-адресов:

```
<a href="http://www.yandex.ru">Яндекс</a>
<a href="http://www.admiral.ru/hp/dunaev">Сам себе Web-дизайнер</a>
<a href="ftp://ftp.admiral.ru/PUB/programs.zip">Бесплатная программа</a>
<a href="mailto:name@anyserver.ru">Отправить почту</a>
```

Пути поиска могут быть абсолютными и относительными. Абсолютный путь описывает местоположение файла, начиная с самого высокого уровня, и включает имена всех папок, ведущих к файлу. Ошибка в записи абсолютного пути (адреса) файла приводит к тому, что файл не будет найден. Относительный путь (адрес) описывает местоположение файла относительно места расположения текущего документа, в котором была инициализирована ссылка с данным адресом. Так, если вы указываете просто имя файла myfile.htm, это означает, что задается относительный адрес. В данном случае браузер будет искать его в той же папке, где находится текущий документ.

Если перед именем файла поставить `../` (например, `../myfile.htm`), то браузер будет искать файл в папке, находящейся на один уровень выше, чем тот, в котором находится текущий документ. Аналогично, если перед именем файла поставить `../../` (например, `../../myfile.htm`), то браузер будет искать файл в папке на два уровня выше, чем текущий.

При создании ссылок вы можете указывать не только на конкретные документы и программы (т. е. конкретные файлы, используя путь к ним), но и на папки (каталоги). Другими словами, адрес — это описание места расположения ресурса (единицы хранения информации). Он может быть точным или "приблизительным" (неполным). Вы можете ссылаться на папку, HTML-документ, документ, созданный каким-либо приложением (например, MS Word, MS Excel), или просто на текстовый файл с расширением txt. Наконец, можно сослаться на файл программы с расширением exe. Однако в последнем случае (по крайней мере, в Internet Explorer) сработает защита вашего компьютера. Появится окно с предупреждением и предложением возможных вариантов: запустить файл программы или сохранить его на диске — способ защиты от потенциальных вирусов. В этом случае вы сами решаете, что делать. Можно сразу запустить файл программы, а можно сохранить его на локальном диске, проверить с помощью антивирусной программы и только потом запустить, вылечить или уничтожить. Если при связи по протоколу HTTP имя файла в URL-адресе не указано, то по умолчанию предполагается файл index.htm или index.html.

Что происходит при инициализации ссылки на некоторый ресурс Интернета? Если запрашивается HTML-документ, то он открывается в окне браузера. Некоторые типы файлов (например, графические файлы форматов GIF, JPEG и др.) браузер также может открыть в своем окне. В ряде случаев для этого требуются специальные подключаемые модули (plug-in). Если браузеру не хватает собственных возможностей, то он пытается открыть файл с помощью ассоциированного с ним приложения, установленного на компьютере пользователя (например, MS Word, MS Excel, Apple QuickTime Player и т. п.).

Ряд файлов (например, с расширениями asp, php, pl и др.) открываются не браузером, а серверной программой, которая их обрабатывает, а браузеру клиента передает лишь результат (обычно в виде html-файла). При инициализации ссылки с адресом электронной почты (`mailto:`*`адрес`*) обычно открывается окно программы почтового клиента (например, Microsoft Outlook Express), в которое указанный адрес получателя вводится автоматически.

При обращении к серверу иногда требуется передать ему некоторые параметры. В этом случае значения параметров следуют за адресом через символ `?`.

Ниже приведен пример графической ссылки на счетчик количества посещений сайта, расположенный на сервере поисковой системы Rambler:

```
<a href="http://top100.rambler.ru/top100/">
<img src=http://counter.rambler.ru/top100.cnt?123456
     alt="Rambler's Top100"
     width=81 height=63 border=0>
</a>
```

Здесь к URL-адресу графического изображения счетчика добавлен идентификационный номер 123456, по которому в базе данных на сервере ищется количество запросов изображения счетчика. Щелчок на изображении счетчика приводит к переходу на страницу Top100 сайта Rambler.

1.2.4. Вставка элементов из внешних источников

Если текст в HTML-документ может быть введен непосредственно, то для вставки на Web-страницу графики, звука, видео и т. п. применяются специальные теги и атрибуты с адресами соответствующих файлов. При загрузке в браузер HTML-документа с такими тегами в него будут загружены и требуемые файлы, если, разумеется, они будут найдены. В данном разделе мы рассмотрим вставку основных элементов, таких как графические изображения, Flash-документы (мультфильмы), звук, видео, и других HTML-документов.

Вставка изображений

Графические изображения из файлов можно вставлять в документ как элемент, занимающий определенную прямоугольную область, и как фон всей страницы.

Вставка отдельных графических изображений

Вставка на страницу изображения из файла графического формата производится с помощью тега `<img>` (от англ. *image* — изображение) с указанием адреса файла в качестве значения обязательного атрибута `src`:

```
<img src="адрес_графического_файла">
```

Адрес графического файла — это либо URL-адрес, либо имя файла, возможно, с указанием пути. Путь может быть абсолютным или относительным. Если в качестве адреса файла указано просто имя файла, то браузер будет искать его в той же папке, в которой расположен вызывающий его HTML-документ.

Примеры:

```
<img src="http://www.myweb.ru/images/picture.jpg">
<img src="C:/myimages/picture.jpg">
<img src="picture.jpg">
```

В теге `<img>` можно использовать атрибут `lowsrc`, который принимает в качестве значения адрес графического файла. Вы можете создать два графических файла: один (например, пусть это файл picture.jpg) содержит изображение, полученное с высоким разрешением, а другой (например, picturelow.gif) — рисунок, полученный с низким разрешением и занимающий меньший объем. Тогда тег

```
<img src="picture.jpg" lowsrc="picturelow.gif">
```

предпишет браузеру сначала загрузить файл picturelow.gif, а затем по мере загрузки страницы заменить его файлом picture.jpg. Этот метод позволяет ускорить формирование страницы в окне браузера.

Для указания размеров изображения на экране предназначены атрибуты `width` (ширина) и `height` (высота), измеряемые в пикселах или процентах. Если указать эти атрибуты, то браузер сначала выделит место под графику, подготовит макет документа, отобразит текст и только потом загрузит картинку. Заметим, что браузер сжимает или растягивает (масштабирует) изображение, встраивая его в прямоугольник указанного размера. Если задан только один из атрибутов, то второй из них рассчитывается браузером так, чтобы сохранить оригинальную пропорцию размеров изображения. Без указания атрибутов `width` и `height` загрузка графики происходит сразу же, как только браузер начинает интерпретировать тег `<img>` и, следовательно, вывод последующего текста и других элементов задерживается. При этом размеры изображения на экране оказываются равными оригинальным. Ниже приведены примеры указания размеров изображения:

```
<img src="picture.jpg" width=40 height=20>
<img src="picture.jpg" width="30%" height="25%">
<img src="picture.jpg" width="140">
```

Чтобы предусмотреть возможность показа документа в режиме отключения загрузки графики, а также в неграфических браузерах, используется атрибут `alt`. Этот атрибут так называемого альтернативного текста принимает в качестве значения текстовую строку, которая будет появляться вместо графического изображения. Обычно в этой строке указывают краткое название изображения и/или размеры графического файла. При наличии текстового описания пользователь сможет сам решить, стоит ли загружать тот или иной графический файл или лучше не тратить на это время. Например:

```
<img src="picture.jpg" alt="Мой портрет 12К">
```

Положение (точнее, выравнивание) изображения относительно других элементов документа можно определить с помощью атрибута выравнивания `align`, который может принимать следующие значения:

- `absbottom` — выравнивание нижней границы изображения по нижней границе текущей строки;
- `absmiddle` — выравнивание середины изображения по середине текущей строки;
- `baseline` — выравнивание нижней границы изображения по базовой линии текущей строки;
- `bottom` — то же, что и `baseline`;
- `top` — верхняя граница изображения выравнивается по самому высокому элементу текущей строки;
- `texttop` — верхняя граница изображения выравнивается по самому высокому текстовому элементу текущей строки;
- `left` — изображение располагается у левого края окна; текст и другие элементы обтекают изображение справа;
- `right` — изображение располагается у правого края окна; текст и другие элементы обтекает изображение слева.

Обычно атрибут `align` тега `<img>` используется для выравнивания изображения относительно текста. Значения `left` и `right` атрибута `align` задают особый тип выравнивания — обтекание изображения текстом.

Для прекращения выравнивания изображений, заданного с помощью `align=left` или `align=right`, относительно текста можно использовать тег `<br>` перевода строки с атрибутом `clear`. Этот атрибут может принимать три значения:

- `left` — для прекращения обтекания текстом элементов (изображений), выровненных по левому краю;
- `right` — для прекращения обтекания текстом элементов (изображений), выровненных по правому краю;
- `all` — для прекращения обтекания текстом элементов (изображений), выровненных либо по левому, либо по правому краю.

По умолчанию изображение встраивается почти вплотную с текстом или другими элементами документа, что может быть некрасиво. Чтобы этого избежать, можно задать пустые поля вокруг иллюстрации. Поля создаются с помощью атрибутов `vspace` для верхнего и нижнего полей и `hspace` для бо-

ковых полей в теге <img>. Значения этих атрибутов указываются в виде целых чисел, определяющих размеры полей в пикселах. Например:

```
<img src="picture.jpg" align=left hspace=20 vspace=10>
```

Для задания черной рамки вокруг изображения служит атрибут border, числовое значение которого определяет толщину рамки в пикселах. Если этот атрибут не указан, то рамка не отображается.

Примечание

Если тег <img>, задающий изображение, вставить в контейнерный тег ссылки <a>, то получится так называемая *графическая ссылка (баннер)*. Браузер по умолчанию выделяет ссылки рамками вокруг их графических изображений. Чтобы исключить отображение рамки, достаточно использовать в теге <img> атрибут border=0.

Фоновая графика

Фон клиентской области окна браузера можно заполнить изображением из графического файла. Фоновая графика задается в теге <body> с помощью атрибута background, значением которого является URL-адрес или имя графического файла. Например:

```
<body background="myfon.gif">
```

Для браузера Internet Explorer можно указать дополнительный атрибут bgproperties=fixed, запрещающий прокрутку фонового изображения. Например:

```
<body background="myfon.gif" bgproperties=fixed>
```

В качестве фонового изображения обычно используется небольшой и простой рисунок (так называемая *текстура*), который многократно выводится на экран, заполняя все окно. Этот рисунок может представлять собой небольшой прямоугольник или же длинную узкую полоску (например, с градиентной заливкой). Такие изображения, подобно листам обоев, покрывают все окно браузера, оставляя незаметными стыки. Кроме того, поскольку изображение небольшое (не более нескольких килобайтов), создание фона происходит быстро.

В качестве фона также можно использовать один экземпляр изображения, например фотографию. Поскольку размеры окна браузера и монитора могут быть самыми различными, то размеры фонового изображения должны соответствовать типичному или даже крайнему случаю. Например, если типичными являются разрешение монитора 1024×768 пикселов и полностью раскрытое окно браузера, то фоновое изображение должно иметь

размеры 1024×768 пикселов или больше. Однако при большем разрешении монитора в полностью открытом окне браузера будут видны 4 экземпляра этого же изображения, что обычно связано с проблемой их стыковки. Следует также иметь в виду, что файлы изображений с большим разрешением и сложные по содержанию имеют большой объем, что не вполне годится для использования в Web-приложениях.

Примечание

Создавать графический фон не только для документа, но и для отдельных его элементов (например, кнопок, таблиц, абзацев и др.) позволяют каскадные таблицы стилей.

Вставка Flash-документов

Flash-документ (мультфильм, ролик, клип) может включать разнородное содержимое: текст, растровую и векторную статическую и анимационную графику, звук, видео и элементы пользовательского интерфейса. Вы можете создать отдельную страницу или даже целый сайт, используя технологию Flash. Flash-документы разрабатываются с помощью пакета Macromedia Flash, выходные файлы которого имеют расширение swf. Просмотр swf-файлов производится с помощью специального Flash-проигрывателя. Чтобы просмотреть swf-файл в окне браузера MS Internet Explorer, требуется элемент управления ActiveX, реализующий функции Flash-проигрывателя, ссылка на который вставляется в HTML-документ с помощью тегов `<object>` и `<param>`:

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
        codebase=
"http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab
        #version=7,0,0,0"
        id="myflash">
  <param name="movie" value="URL-адрес swf-файла">
</object>
```

Здесь атрибут `classid` принимает в качестве значения уникальный регистрационный идентификатор объекта, соответствующего Flash-проигрывателю. Атрибут `codebase` содержит URL-адрес исходных файлов Flash-проигрывателя версии 7.0 на случай, если он не установлен на компьютере пользователя. Тег `<param>` служит для указания URL-адреса swf-файла, содержащего собственно Flash-документ, который должен быть воспроизведен Flash-проигрывателем. Атрибут `id` указан только для обеспечения возможности доступа к объекту с помощью сценария.

Чтобы Flash-проигрыватель был вставлен в Web-страницу браузерами, не воспринимающими теги <object> и <param> (например, Netscape Navigator), внутри контейнера <object> используют тег <embed>:

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
        codebase="http://download.macromedia.com/pub/shockwave/cabs/
flash/swflash.cab
        #version=7,0,0,0">
  <param name="movie" value="URL-адрес swf-файла">
  <embed src="URL-адрес swf-файла" name="myflash"
         type="application/x-shockwave-flash"
         pluginspage="http://www.macromedia.com/go/getflashplayer">
</object>
```

Здесь в теге <embed> атрибут name играет такую же роль, что и id в <object>, а атрибут pluginspage — такую же роль, что и codebase.

Кроме рассмотренных выше, можно использовать множество других параметров Flash-проигрывателя, среди которых особо отметим наиболее важные, задаваемые с помощью тегов <param>:

- прозрачность фона Flash-документа (важно при согласовании с фоном HTML-документа):
  - в теге <object> используется тег:

    ```
    <param name="wmode" value="transparent">
    ```

  - в теге <embed> используется атрибут:

    ```
    wmode="transparent"
    ```

- цвет фона Flash-документа:
  - в теге <object> используется тег:

    ```
    <param name="bgcolor" value="цвет">
    ```

  - в теге <embed> используется атрибут:

    ```
    bgcolor="цвет"
    ```

    Здесь *цвет* задается именем или шестнадцатеричным кодом, например, #f0f0f0;
- качество воспроизведения Flash-документа:
  - в теге <object> используется тег:

    ```
    <param name="quality" value="уровень_качества">
    ```

  - в теге <embed> используется атрибут:

    ```
    quality= "уровень_качества"
    ```

Здесь *уровень_качества* принимает значения: `low` (низкое), `medium` (среднее), `high` (высокое), `best` (самое лучшее), `autolow` (автонизкое), `autohigh` (автовысокое).

В пакете Macromedia Flash имеются средства автоматизации создания HTML-документа, содержащего заданный Flash-документ, которые позволяют легко установить все необходимые параметры.

Вставка звука и видео

Современные браузеры могут воспроизводить видео- и звуковые файлы различных форматов. Для этого они используют встроенные проигрыватели (plug-in, элементы управления ActiveX) или внешние программы-проигрыватели. Вставить звук или видео в HTML-документ можно с помощью различных тегов:

- `<bgsound src=`*адрес_файла*`>` — для вставки фонового звука;
- `<img dynsrc=`*адрес_файла*`>` — для вставки видео в формате AVI;
- `<a href=`*адрес_файла*`>` — для вставки звуковых и видеофайлов;
- `<embed src=`*адрес_файла*`>` — для вставки звуковых и видеофайлов.

При решении вставить звук и/или видео в HTML-документ следует учитывать, что соответствующие файлы имеют довольно большой объем. Наиболее популярными в Web сейчас являются файлы звуковых форматов MP3, WMA, AIFF, AU, RealAudio (с расширениями ra и ram), MP4, MIDI и видеоформатов MPEG, MOV. Звуковой формат WAV и видеоформат AVI в Интернете используются довольно редко.

Для операционной системы Windows стандартными звуковыми файлами являются WMA и WAV, а для Macintosh — AIFF. Общепринятым видеоформатом для Windows, Macintosh и UNIX является MPEG. Формат QuickTime (MOV) также широко распространен и может использоваться Windows и Macintosh. При выборе других форматов желательно предупреждать об этом пользователей.

Фоновый звук

Тег `<bgsound>` позволяет указать звуковой файл формата WMA, AU, MIDI или WAV, который будет проигрываться при загрузке страницы. При этом на экран не выводится панель управления воспроизведением.

Тег `<bgsound>` имеет следующие атрибуты:

- `src` — URL-адрес звукового файла;