



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

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	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Игорь Шишигин</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Татьяна Лапина</i>
Компьютерная верстка	<i>Натальи Смирновой</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

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

© Дунаев В. В., 2008
© Оформление, издательство "БХВ-Петербург", 2008

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 <i>alert</i>	132
2.2.2. Метод <i>confirm</i>	133

2.2.3. Метод <i>prompt</i>	134
2.2.4. Метод <i>write</i>	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 <i>String</i>	179
2.7.3. Методы объекта <i>String</i> обработки строк	180
2.7.4. Методы объекта <i>String</i> 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 <i>Array</i>	194
2.8.5. Методы объекта <i>Array</i>	195
2.8.6. Функции обработки числовых массивов	199
2.9. Числа	201
2.9.1. Создание объекта <i>Number</i>	204
2.9.2. Свойства объекта <i>Number</i>	204
2.9.3. Методы объекта <i>Number</i>	205
2.10. Математические вычисления.....	207
2.10.1. Свойства объекта <i>Math</i>	207
2.10.2. Методы объекта <i>Math</i>	207

2.11. Дата и время	209
2.11.1. Создание объекта <i>Date</i>	209
2.11.2. Методы объекта <i>Date</i>	211
2.11.3. Календарь	217
2.12. Объект <i>Boolean</i>	223
2.13. Объект <i>Function</i>	224
2.13.1. Создание объекта <i>Function</i>	224
2.13.2. Свойства объекта <i>Function</i>	225
2.13.3. Методы объекта <i>Function</i>	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 <i>write()</i>	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. Древоподобный 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 РНР	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 <i>header()</i>	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 <i>window</i>	531
П1.1.1. Свойства объекта <i>window</i>	531
П1.1.2. Методы объекта <i>window</i>	533
П1.1.3. События объекта <i>window</i>	534
П1.2. Объект <i>document</i>	535
П1.2.1. Коллекции объекта <i>document</i>	536
П1.2.2. Методы объекта <i>document</i>	537
П1.2.3. События объекта <i>document</i>	537
П1.3. Объект <i>location</i>	538
П1.3.1. Свойства объекта <i>location</i>	538
П1.3.2. Методы объекта <i>location</i>	539
П1.4. Объект <i>history</i>	539
П1.4.1. Свойство объекта <i>history</i>	540
П1.4.2. Методы объекта <i>history</i>	540

П1.5. Объект <i>navigator</i>	540
П1.5.1. Свойства объекта <i>navigator</i>	540
П1.5.2. Коллекции объекта <i>navigator</i>	541
П1.5.3. Методы объекта <i>navigator</i>	541
П1.6. Объект <i>event</i>	541
П1.7. Объект <i>screen</i>	543
П1.8. Объект <i>TextRange</i>	543
П1.8.1. Свойства объекта <i>TextRange</i>	543
П1.8.2. Методы объекта <i>TextRange</i>	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. Основным принципом, 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 Front-Page 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 (Hyper-Text 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 predetermined 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 ``, задающий жирный шрифт для всей заключенной в нем строки, а в контейнере `` 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.

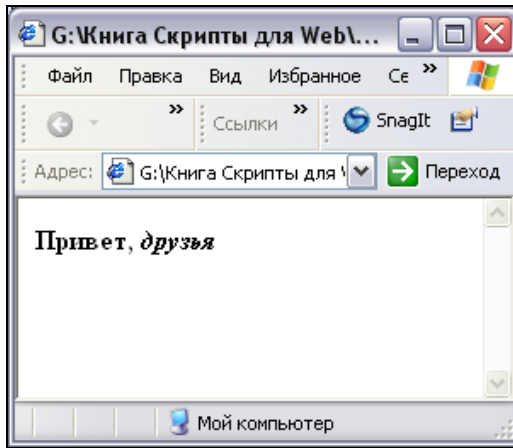


Рис. 1.1. Пример простого HTML-документа

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 `` с обязательным атрибутом `src="адрес_файла"`. В качестве значения атрибута `src` используется имя (при необходимости с указанием пути) или URL-адрес графического файла. Например,

```

```

Если не указывать необязательные атрибуты, то их значения будут равны принятым по умолчанию. Например, тег `` 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 `<! >`, 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 `
`. 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 nereкомендованные консорциумом W3C теги и атрибуты, такие как ``, `<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 nereкомендованные 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>
  
</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` (`bgproperties=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 predefinedного 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 <div> и

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